АтомМайнд

Инструкция пользователя

Содержание

1 Введение

Эта документация описывает платформу AtomMind и созданные на ней стандартные продукты для различных вертикальных рынков.

Документация находится в режиме непрерывного улучшения. AtomMind - это очень большая платформа, находящаяся в процессе постоянного развития и совершенствования. В данном руководстве команда разработчиков AtomMind пытается как бы "сыграть в догонялки" с постоянно растущим темпом развития AtomMind.

Таким образом, мы рекомендуем вам иметь в виду несколько моментов при чтении руководства:

- AtomMind - довольно сложная система, и ее освоение требует определенного времени. Используйте данное руководство скорее как справочник, а не учебник. Если вам встретился незнакомый термин, пожалуйста, вернитесь к более ранним темам или пройдите по ссылкам. Определения терминов, как правило, даны при их первом упоминании в тексте
- Один и тот же компонент системы может иметь несколько тем, посвящённых ему, в которых он будет рассмотрен соответственно уровню знакомства с системой. Вы всегда можете просмотреть более раннюю тему, связанную с этим компонентом, в ней будет более простое и доступное объяснение.
- Прежде всего, мы стремимся обеспечить простую для чтения и использования документацию, точную, но при этом понятную. Мы понимаем, что наши пользователи - это реальные люди, которые хотят получить необходимый им *результат* от системы, и мы надеемся, что данное руководство будет в этом хорошим помощником. Приятного чтения!

1.1 О АО "ТВЭЛ"

AtomMind разработан и поддерживается АО "ТВЭЛ" - российским производителем ядерного топлива, производственным холдингом, входящим в состав Топливного дивизиона госкорпорации «Росатом», г.Москва. AtomMind - платформа "Интернета вещей", позволяющая объединять данные со структурированных источников, производственного оборудования и немашиных данных с целью создания комплексных цифровых двойников производств. Продвинутые возможности системы в части математического обеспечения, в том числе с использованием самых продвинутых алгоритмов машинного обучения, поддержкой нейронных сетей, позволяют создавать самые передовые модели по улучшению качества изделий на производственных предприятиях, перейти от плановых ремонтов к предиктивному обслуживанию, обеспечить интеллектуальную поддержку принятия решений для руководителей как на уровне цехов, так и предприятия в целом.

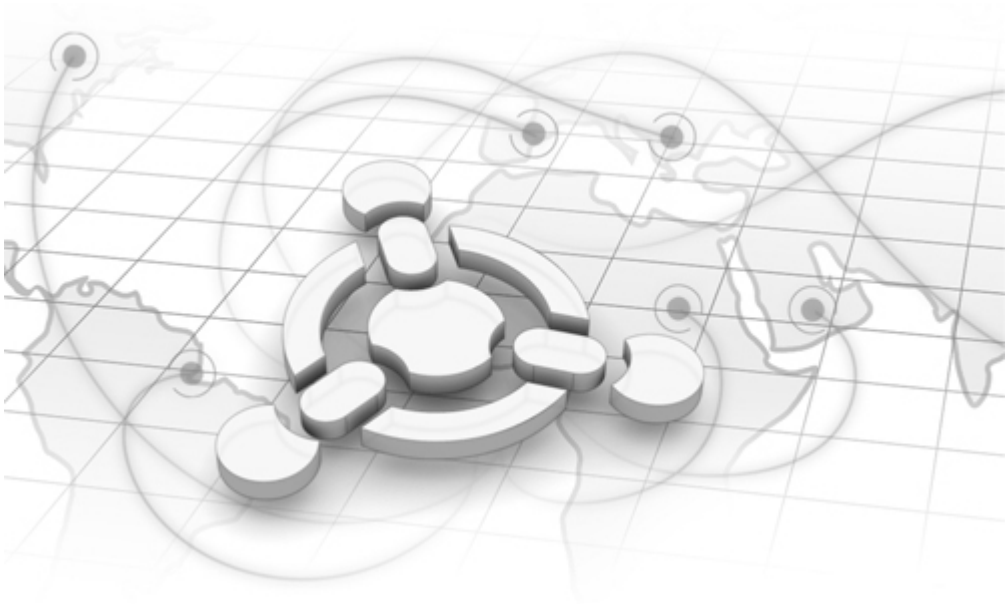
1.2 Об авторских правах

1.3 Обратная связь

Для облегчения наших усилий по созданию эффективной документации, ТВЭЛ просит вас отправлять нам любые комментарии, дополнения или исправления относительно документации. Ваши отзывы очень важны для нас и мы будем использовать их для улучшения нашей продукции и услуг.

Свои комментарии вы можете отправить на: info@tvel.ru. Большое спасибо за вашу помощь.

2 Обзор платформы



ТВЭЛ AtomMind – это low-code платформа для разработки приложений Интернета вещей, обеспечивающая цифровую трансформацию предприятий, начиная от подключения разнородных активов и вплоть до глубинной аналитики данных, позволяющей принимать стратегические решения по развитию бизнеса.

Платформа и созданные на её основе вертикальные решения помогают предприятиям, малому и среднему бизнесу, системным интеграторам, производителям оборудования, операторам связи и поставщикам управляемых услуг разрабатывать и внедрять собственные решения Интернета вещей, работающие как на периферийных устройствах, так и в облаке.

Основная идея AtomMind - сделать разработку IoT решений доступной для инженеров с минимальным опытом программирования, значительно уменьшая сроки вывода продуктов на рынок и связанные с этим затраты на НИОКР.

Базовые компоненты программы основаны на технологии Java и могут быть использованы на большинстве современных операционных систем и оборудования.

Подключение ваших устройства к платформе и ее интеграция с существующим IT ландшафтом происходит очень просто и малозатратно.

Экосистема AtomMind также включает множество отраслевых решений: Сетевое управление и мониторинг, SCADA и MES, Автоматизация зданий и Умные города, Контроль доступа и безопасность, Мониторинг транспорта, Управление инцидентами, Машинное обучение и аналитика, и многое другое.

2.1 Low-code разработка и DevOps

AtomMind - это *платформа low-code разработки*. Используется для создания и внедрения приложений, продуктов, решений, сервисов IoT и цифрового предприятия, а также для работы с ними. Все это осуществляется через графический пользовательский интерфейс, а не с помощью традиционного тексто-ориентированного программирования.

В то же время, AtomMind является *платформой комплексной разработки* безопасных, надежных, масштабируемых и высокопроизводительных приложений практически без написания кода на всех этапах, от синтаксического разбора протоколов устройств до пиксельной настройки интерфейса. Платформа разработана не только для создания прототипов или B2C продуктов, но также обеспечивает соблюдение корпоративных стандартов на всех стадиях реализации приложения!

Low-code и no-code разработка значительно уменьшает время вывода ваших IoT сервисов на рынок. Из-за того, что платформа позволяет даже инженерам с хорошим знанием индустрии, но маленьким опытом программирования

создавать и разворачивать как рыночно-ориентированные продукты, так и сервисы внутренней цифровизации, снижаются затраты на разработку и эксплуатацию. Более того, как показывает наш широкий опыт, командам с высокой бизнес-экспертизой нравится участвовать в процессе создания ПО.

То, как быстро low-code разработка завоевывает позиции, очень напоминает, как несколько десятилетий назад Java и .NET представили новый уровень абстракции и производительности поверх C/C++ разработки.

AtomMind обладает широчайшей индустриальной применимостью и набором блоков для создания решений для промышленного IoT, а также хорошо интегрируется с облачными платформами крупных поставщиков. Все это помогает вывести производительность вашей команды разработки на очень высокий уровень.

Будучи лидером в обработке специфичных для IoT данных, AtomMind также предлагает множество продвинутых инструментов для сбора, анализа и визуализации данных, созданных человеком. Это позволяет еще больше расширить спецификацию вашего low-code продукта, добавив модули и элементы, характерные для CRM, EAM, ERP, MES, ITSM, CMMS, BPM и подобных программных решений.

Кроме улучшения показателей времени и стоимости вашего процесса разработки, AtomMind повышает эффективность всех связанных активностей, таких как развертывание, защита и масштабирование веб-сервисов, пакетирование и распределение необлачных продуктов, интегрирование пользовательских решений на основе платформы в большие IT ландшафты, и т.д.

Большие предприятия еще больше выигрывают от использования AtomMind. По мере увеличения количества решений на базе платформы, они начинают действовать как единая цифровая платформа предприятия, по сути многократно используя существующие IT ресурсы и во много раз уменьшая количество точек интеграции. Это приводит к минимизации команды разработки, расходов на оборудование и поддержку ПО.

AtomMind также трансформирует бизнес-модель интеграторов и независимых поставщиков ПО, позволяя им реализовать low-code как сервис, известный как "сервис разработки облачных приложений".

Таким образом, AtomMind позволяет большинству департаментов и отделов вашего бизнеса включиться в цифровые цепочки создания ценности, а также укрепить ваши команды разработки с минимальными нормативными затратами.

2.2 Структура платформы

Каждое развертывание AtomMind состоит из одного или нескольких серверов 144, на которых размещаются компоненты одного или нескольких решений/продуктов/сервисов, созданных на базе платформы. Сервер представляет собой Java-приложение, работающее на операционных системах, поддерживающих технологию Java, установленных на физическом сервере, виртуальной машине или промышленном ПК. Каждый сервер может быть частью независимого отказоустойчивого кластера 1326, что обеспечивает его высокую доступность.

Ядром каждого сервера AtomMind является единая модель данных 41, представленная иерархическим контекстным деревом 41. Каждый контекст представляет определенное устройство, источник данных или ресурс системы в виде набора сущностей: переменных 61, функций 70, событий 73 и действий 87.

Сервер запускает различные плагины 207, отвечающие за развертывание 144 приложений, получение 494, хранение 692, обработку 735 и визуализацию 912 данных и пр. В зависимости от структуры системы, каждый сервер может использовать либо множество разных плагинов ("надежная" архитектура), либо только несколько плагинов (архитектура на основе микросервисов).

В крупной инсталляции разные серверы платформы могут иметь разные роли. Серверы связаны между собой с помощью распределенной архитектуры 1332, а также могут быть разделены на несколько шин (например, "edge" и "облако"). Распределенная архитектура позволяет серверам обмениваться частями единых моделей данных, фактически реализуя структуру распределенного приложения.

Серверы AtomMind взаимодействуют с устройствами 497, управляют ими, конфигурируют, а также собирают потоки данных и событий. Специфичные для устройств данные всегда нормализуются, т.е. преобразуются в формат единой модели данных AtomMind. Нормализация происходит либо на стороне устройства с помощью программного/аппаратного агента 684, либо на стороне сервера через драйвер устройства 518.

Драйверы также используются для доступа к структурированным источникам данных, таким как сторонние системы. Пакеты платформы включают множество драйверов для стандартных коммуникационных протоколов, такх как HTTP, MQTT, SNMP или Modbus.

Каждый сервер обеспечен различными хранилищами 692, которые хранят настройки, события, а также бинарные данные в NoSQL, реляционных и других базах данных.

В AtomMind, нет формальной разницы между серверами разработки, тестирования и производства. Любой сервер может выступать в роли интегрированной среды разработки и предлагать инструменты для low-code разработки, внедрения, отладки и DevOps. Их практическая доступность ограничена функциями безопасности и контроля доступа 477 платформы.

Платформа предоставляет две независимые системы проектирования UI (интерфейса пользователя): браузерный конструктор Web UI и кроссплатформенный десктопный конструктор приложений.

Оба виды интерфейсов могут использоваться для администрирования и настройки платформы, а также содержат различные визуальные редакторы для low-code разработки драйверов, процессов, инструментальных панелей и других компонентов приложений.

Монолит vs микросервисы

AtomMind имеет сервис-ориентированную архитектуру (SOA), спроектированную для поддержки разработки приложений на базе микросервисов.

У AtomMind Server очень компактное ядро, и в рамках подхода к созданию микросервисов каждый сервер запускает лишь единственный "первичный" плагин 207, которые предлагает отдельный сервис в рамках основного приложения. Множество серверов соединены между собой через распределенную архитектуру 1332 или классические методы, такие как HTTP/REST или брокер сообщений.

В то же время, решения малого и среднего масштаба могут использовать способность AtomMind Server запускать множество плагинов. Развертывание всех плагинов на одном сервере соответствует схеме монолитного приложения. Несмотря на очевидные недостатки такого подхода, он позволяет быстрее создавать приложения и выводить их на рынок.

Конечно, архитектура решения может подразумевать использование "смешанной" архитектуры, при которой одни серверы делят некоторое количество модулей, а на других запускается лишь один определенный сервис. Вообще, чем выше нагрузка на ваше решение, тем большее количество микросервисов вам следует учесть на стадии проектирования архитектуры.

2.3 Процесс разработки приложений

Создание нового решения, продукта или сервиса на базе платформы AtomMind - это простой и понятный процесс, включающий несколько шагов:

- Планирование архитектуры приложения и верхнеуровневый дизайн
- Выбор модулей платформы для использования при реализации, распределяя их функции между серверами платформы
- Создание прототипов компонентов UI и инструментальных панелей приложения
- Подготовка 152 экземпляров платформы для разработки, тестирования, промышленной эксплуатации и других сред (таких как кандидат на окончательную версию)
- Установка репозитория системы контроля версий (например, Git) для вашего приложения
- Настройка подключений устройств и сбора данных 494
- Настройка хранения 692 и агрегации 745 данных
- Реализация моделей 810 данных (цифровых двойников ваших активов и процессов)
- Реализация цепочек обработки данных через наборы правил 743, процессы 895 и т.д.
- Настройка глубокой аналитики, такой как машинное обучение 865, обработка сложных событий 774 и т.д.
- Создание инструментальных панелей 912 оператора и других частей пользовательского интерфейса приложения
- Настройка автозапуска 941, чтобы активировать нужные точки входа пользователя
- Определение безопасности приложения, схем аутентификации, пользовательских ролей и прав доступа 477
- Установка средств интеграции со сторонними системами
- Брендирование и интернационализация вашего приожения
- Упаковка ресурсов и настройка через модуль приложения 1318
- Настройка процессов развертывания и DevOps
- Развертывание приложения на промышленных серверах (для веб-сервисов), либо подготовка пакета установки от производителя оборудования (для необлачных продуктов)
- Настройка процессов проверки целостности и самоконтроля

3 Единая модель данных

Одной из важнейших и инновационных концепций AtomMind является *единая модель данных сервера*. Эта модель создана для объединения различных данных от устройств в единую систему путем преобразования этих данных в "нормализованный" вид. Нормализованные данные могут быть:

- Сохранены в центральной базе данных;
- Однородно обработаны и маршрутизированы инструментами анализа данных;
- Представлены, экспортированы и импортированы с использованием стандартных компонентов;
- Доступны для интегрированных визуальных редакторов;
- Совместно использованы серверами платформы;
- Доступны внешним системам через различные API.

Контексты

Сердцем модели данных AtomMind является *дерево контекстов*. Дерево контекстов - это иерархическая структура контейнеров данных, называемых **контекстами** 41.

Целью контекстов является обеспечение единого доступа к определенным системным ресурсам или устройствам.

Каждый контекст раскрывает несколько типов объектов 44: переменные 61, функции 70, события 73 и действия 87. Эти объекты позволяют взаимодействовать с контекстом стандартным образом, независимо от природы устройства или ресурса, представленных контекстом.

Таблицы данных

Другое важнейшее понятие единой модели данных AtomMind - стандартный элемент данных, называемый таблица данных 49 (Data Table). Это структура табличных данных, используемая для представления:

- Значений переменных
- Входящих и исходящих значений функций
- Данных, связанных с определенными событиями
- Любых других единиц данных, передаваемых внутри AtomMind

Таблицы данных очень схожи с таблицами базы данных, так как имеют нуль или более строк и несколько полей определенного формата 49. Ниже представлены несколько важных фактов о таблицах данных:

1. Для унификации, скалярные величины (числа, строки, булевые, даты) представлены в виде таблицы данных с одной строкой и одним полем
2. Списки разнотипных значений обычно представлены в виде таблицы с одной строкой и множеством полей различных форматов, в то время как массивы предпочтительнее представлять в виде таблицы с одним полем и множеством строк.
3. Если тип поля таблицы данных является также таблицей данных, то каждая клетка в этой области содержит вложенную таблицу данных. Допустим неограниченный уровень вложения таблиц.
4. Дескриптор формата 49 таблицы данных содержит исчерпывающую информацию о типах поля, их значениях, о проверке допустимости значений и правилах представления, о перекрестной зависимости ячеек и т.д.

3.1 Контексты

Одним из ключевых понятий платформы AtomMind является *контекст*. Контекст выступает в роли логического "контейнера" для данных, относящихся к какому-либо системному ресурсу, устройству или источнику данных внутри единой модели данных 41. Он также предоставляет методы для управления этими данными и контроля над состоянием объекта. Например, внутренняя работа AtomMind осуществляется с контекстами, которые соответствуют устройствам 49. Создается "Контекст устройства", и этот контекст используется для наблюдения за настройками устройства и их изменения, просмотра событий и т.д.

Дерево контекстов

Практически все данные, контролируемые AtomMind сервером, представлены контекстами. Контексты организованы в иерархическую структуру, которая называется *деревом контекстов*. Каждая установка AtomMind Server содержит одно дерево контекстов. Данное дерево динамично, поэтому контексты можно добавлять и удалять во время нормальной работы сервера.

Все контексты, доступные в исходном дереве AtomMind сервера, описаны в разделе Справочник по контекстам[1450] .

Помимо исходного дерева контекстов сервера, существуют деревья контекстов, используемые определенными модулями платформы. Например, модель данных веб-инструментальной панели[223] - это отдельное контекстное дерево, чьи контексты представляют UI-компоненты инструментальной панели.

Названия и пути контекстов

Каждый контекст имеет *имя*. Имя контекста - это уникальный идентификатор контекста среди дочерних компонентов его родительского контекста. Оно представлено в виде строки, которая может содержать только буквы, числа и символ подчеркивания ("_"). Например, контекст пользователя администратор по умолчанию[479] называется "admin".

Некоторые имена контекстов определены заранее и встроены в систему. В других случаях пользователь может задать имя контекста. Например, при создании новой учетной записи устройства, ее имя используется в качестве имени контекста.

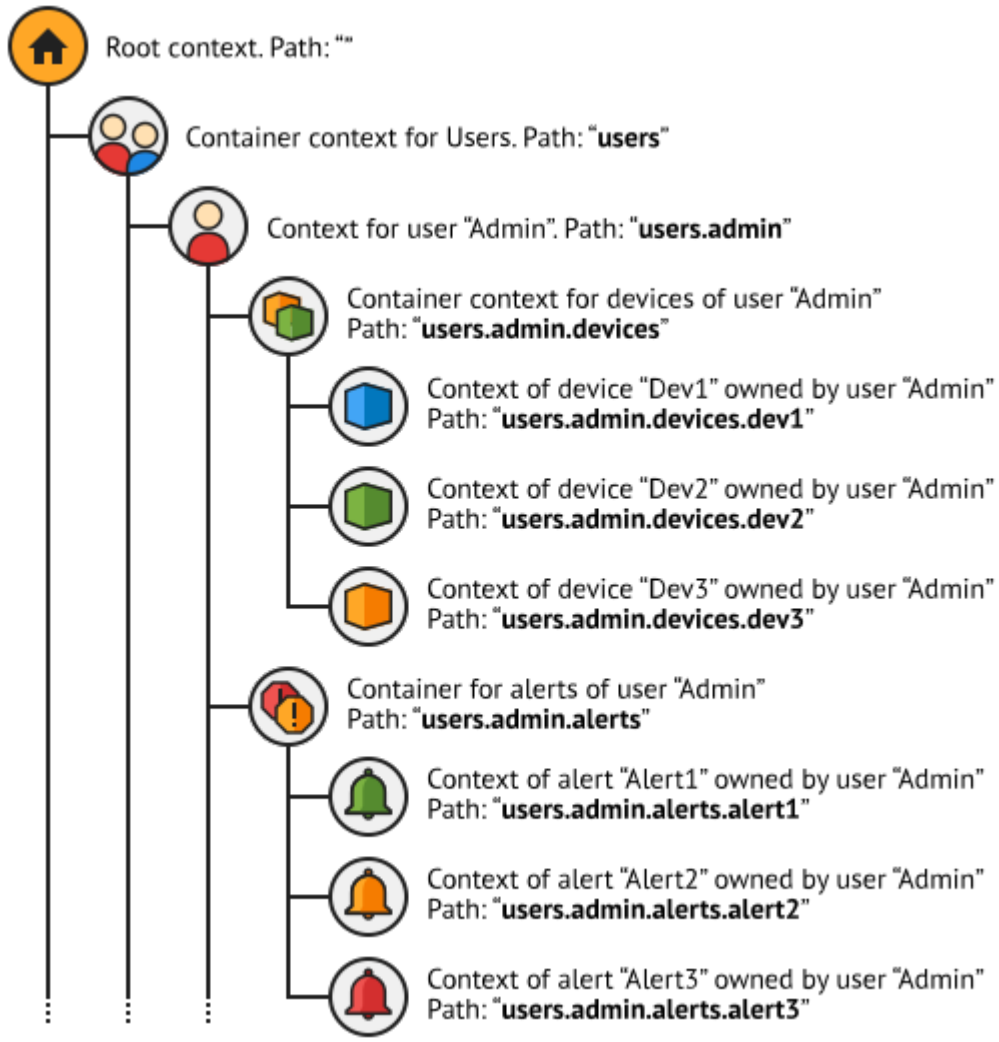
Как уже говорилось, дерево контекста имеет иерархическую структуру, и адресация к каждому контексту в дереве осуществляется через *контекстный путь.* Проще говоря, контекстный путь - это его уникальный идентификатор внутри всего дерева контекстов.

Контекстный путь можно также назвать *полным именем* контекста. Путь состоит из одного и более имен контекста, разделенных точками ("**.**").

Пример 1: контекст **users** соответствует всем пользователям в системе. Он включает в себя некоторые пользовательские данные и операции, например, операцию "Регистрация новой учетной записи пользователя" и т.д.

Пример 2: контекст **users.peter.devices.dev1** (это его полное имя) относится к устройству **dev1**, зарегистрированного под учетной записью **peter.** Данный контекст обеспечивает доступ к настройкам устройства и соответствующим операциям, например, "Конфигурировать устройство" и т.д.

Дерево контекстов "растет" из так называемого *корневого контекста*. Его особенность заключается в имени - "" (пустая строка).



Еще один термин, который имеет отношение к дереву контекстов, - *отношение родитель-потомок*. Каждый контекст, кроме корневого, имеет родительский контекст. Также каждый контекст может иметь нуль и более потомков. Например, **users.admin** является потомком контекста **users** (который, в свою очередь, является контекстом-родителем).

Типы путей контекста

Контекстные пути могут быть *абсолютными* (т.е. разрешаться из корня дерева контекстов) или *относительными* (разрешаться из какого-либо другого контекста). Относительные пути начинаются с точки ("**.**").

Описания контекстов

Описание представляет собой понятное для пользователя имя контекста. Оно используется для ссылки на данный контекст из любого места интерфейса пользователя. Например, описание для контекста **plugins** - "Drivers/Plugins Per-account Configuration".

Иногда пользователь может задать описание вручную, например, при создании новой модели 810. В этом случае, пользовательское описание будет использовано для ссылки на модель в пользовательском интерфейсе.

Использование отдельных свойств для имени и описания контекстов позволяет AtomMind поддерживать внутри системы одно имя (уникальный ID) контекста, и одновременно обеспечить максимально понятное название, используя простое описание. Помимо этого, изменение описания для объекта плавно изменит его внешний вид в интерфейсе (UI) без его "нарушения" (потому что внутри системы у него осталось прежнее имя).

Типы контекстов

Тип контекста помогает AtomMind решить, могут ли одни и те же операции применяться к разным контекстам, таким как свойства между ними. Например, вы можете скопировать свойства одного устройства на другое, однако, будет совершенно бессмысленно копировать свойства устройства в контекст Тревоги 779. Вот почему AtomMind позволяет копировать свойства контекстов только между самими контекстами (операция, известная также под названием репликация).

Тип контекста также используется для поиска контекстов такого же или похожего вида, как объекты.

Тип контекста представляет собой строку, которая может содержать только буквы, числа, символ нижнего подчеркивания ("_") и точку (".").

Объекты контекста

Каждый контекст включает в себя так называемые *объекты* (*entities).* Выделяют четыре типа объектов:

- Переменные 61 (также называемые *свойства* или *настройки*), которые позволяют просматривать или изменять данные, относящиеся к контексту
- Функции 70 (также называемые *операции* или *методы*), используемые для выполнения некоторых действий, связанных с контекстом
- События 73 (также называемые *сообщения* или *уведомления*), необходимые для наблюдения за тем, что происходит в контексте
- Действия 87, которые позволяют пользователям платформы взаимодействовать с контекстом в интерактивном режиме через серию *UI процедур*

Статусы контекста

Каждый контекст имеет *пиктограмму (иконку)*, которая используется для его визуального представления в пользовательских интерфейсах AtomMind Server (таких как Web UI 220).

Некоторые контексты могут иметь различные *статусы*. Например, контекст аппаратного устройства может быть в двух состояниях: "онлайн" и "офлайн". Каждый статус имеет свою пиктограмму и текстовое описание.

Права доступа к контексту

Когда пользователь 478 или ресурс, наследующий права доступа пользователя, пытаются получить доступ к контексту или некоторым его объектам, он делает это, используя свой текущий уровень прав доступа 486 для этого контекста. Последний определяется таблицей прав доступа 487 пользователя.

Уровень прав доступа определяет, что может делать пользователь с контекстом, например:

- Совсем не может получить доступ к контексту
- Может выполнять лишь операции чтения и записи определенных переменных
- Может выполнять определенные операции
- Может создавать определенные события или подписываться на них

3.1.1 Маски контекстов

Маска контекста является строкой, которая соответствует одному и более контекстам. Маска похожи на путь контекста 42, но ее *сегменты* (части, разделенные точками) могут включать *специальные символы* ("**") вместо имен контекстов. Они напоминают маски имен файлов (например, "**.txt**"), которые часто используются при работе с компьютерами. Маски контекстов могут *разрешаться* в несколько путей контекстов, которые *соответствуют* маске

Пример маски контекста, который следует читать как "все устройства всех пользователей": **users.*.devices.***

Группы 751 обеспечивают специальный синтаксис контекстных масок для доступа ко всем членам группы. Для более подробной информации обратитесь к разделу Маски контекстов членов группы 756.

Разрешение масок

Маски могут *разрешаться* в список путей контекста. Например, маска **users.*** может разрешиться в следующий список путей:

users.admin

users.user1

users.user2

и т.д.

А маска **users.*.devices.*** может разрешиться в:

users.admin.devices.thermometer1

users.admin.devices.thermometer2

users.admin.devices.thermometer3

users.user1.devices.gateway1

users.user1.devices.gateway2

users.user2.devices.plc

Процесс разрешения маски всегда включает проверку прав доступа 477. Например, маска **users.*** разрешается в список пользовательских контекстов 1608, доступных при правах доступа для пользователя 478, который запрашивает операцию. Она будет разрешаться в список всех пользователей в системе, только если пользователь, запускающий операцию, имеет права доступа ко всем учетным записям.

Совпадение с маской контекста

Путь контекста может *совпадать* с маской. Помимо этого, маска может *расширять* путь, или же путь может *расширять* маску.

Чтобы *совпадать* с маской контекста, путь должен содержать такое же число сегментов (частей, разделенных "."), что и маска. Если какой-либо сегмент не содержит специальный символ ("*"), он должен быть идентичным соответствующему сегменту в пути контекста.

Если путь *расширяет* маску, главные сегменты пути должны совпадать с маской, однако, путь может включать несколько дополнительных сегментов.

Аналогично, если маска *расширяет* путь, ее начало должно совпадать с путем, но она будет содержать в себе и дополнительные сегменты.

ПРИМЕРЫ СОВПАДЕНИЯ С МАСКОЙ

Путь **users.admin** совпадает с маской **users.***

Путь **users.admin** не совпадает с маской **external_device_servers.***

Путь **users.admin.devices.c1** удлиняет маску **users.***. Он длиннее маски, и поэтому он совпадает с началом маски и добавляет сегменты.

Путь **users.admin.devices.c1** не удлиняет маску **users.*.alerts**. Маска не совпадает с тремя главными сегментами пути.

Маска **users.*.devices.*** удлиняет путь **users.admin**, потому что она добавляет информацию к нему

Маска **users.*.devices.*** не удлиняет путь **reports.impacts_report**, потому что они не совпадают

Маска **devices.*** не совпадает с путем **users.user1.devices**. Маска содержит несуществующий ведущий сегмент **devices**. Например, маска **users.*.devices.*** может использоваться для совпадения с этим путем.

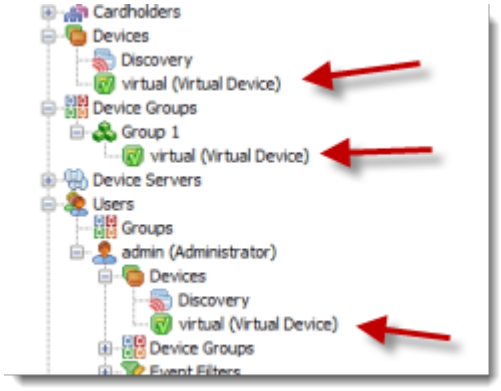
3.1.2 Видимое и действительное дерево контекстов

Данный раздел посвящен объяснению различия между видимым Деревом контекстов и *серверным* Деревом контекстов.

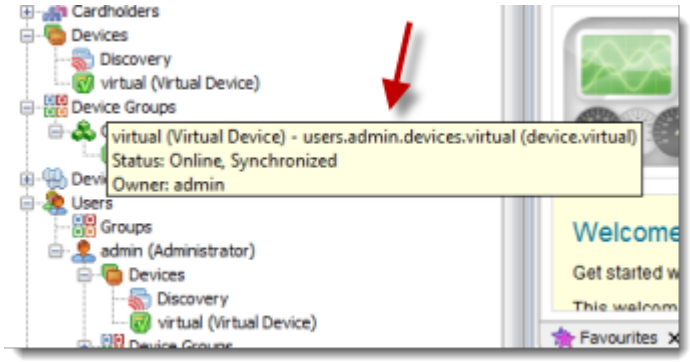
Необходимо знать, что структура дерева контекстов, видимого в Системном дереве 370 и подобных компонентах (таких как Селектор объектов 402) **не** совпадает полностью со структурой дерева контекстов 41 AtomMind Server. Для удобства пользования, узлы, которые относятся к одному серверному контексту, могут появиться в двух, трех и более местах Системного дерева.

Например, один контекст устройства 1494, представляющий аппаратное устройство, может появиться одновременно в:

- узле **Устройства**, отображая устройство активного пользователя
- узле **Все устройства**, включающем все устройства в системе
- одном и более узлах под названием **Группа устройств**



На приведенном выше рисунке, узлы, на которые указывают стрелки, относятся к одному серверному контексту. При наведении курсора мыши на узел Системного дерева, появится всплывающая подсказка, в которой указан его абсолютный серверный путь:



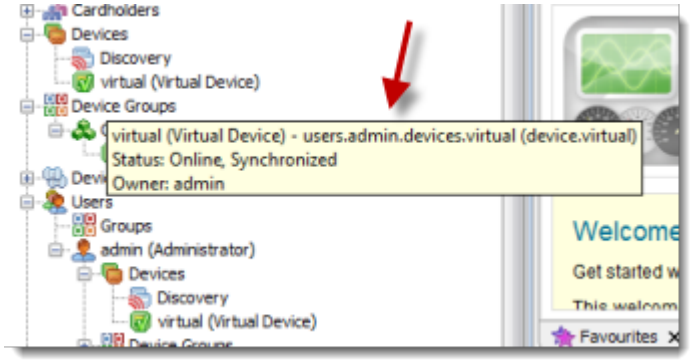
Данное различие может стать причиной неожиданных результатов при проверке путей контекстов, видимых в дереве. Например, дерево может включать в себя контекст **Устройство ABC,** который расположен в **Корень > Все устройства**. Однако, настоящий серверный путь к данному контексту является `users.Owner_Name.devices.deviceABC` где `Owner_Name` – пользователь 478 AtomMind Server, которому принадлежит устройство. Серверный контекст Все устройства не имеет действительных потомков, т.к. он является так называемым *отображенным контекстом.*

Пути контейнеров

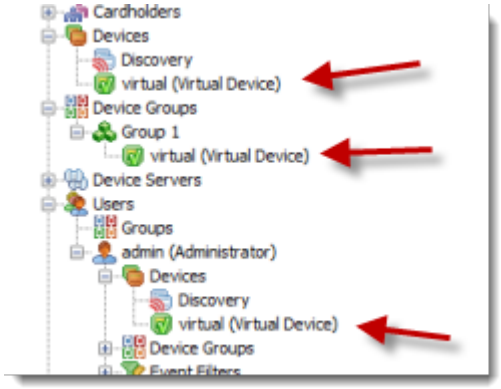
Неопытные пользователи AtomMind иногда путают "действительные" и "отображаемые" контексты контейнеров.

Большинство контекстов в корне системного дерева отображаемые. Например, если вы наведете мышку на узел **Устройства**, вы увидите, что его путь контекста - это `devices`:

Если вы развернете этот узел и проверите путь контекста устройства, расположенного прямо под ним, вы увидите, что его путь `users.admin.devices.virtual`:



Это означает, что действительный контейнер нашего устройства под названием `virtual` - это `users.admin.devices`. Он принадлежит пользователю 478 системы с именем `admin`. Мы можем найти этот действительный контейнер, пройдя путь `Users => admin (Administrator) => Devices` (заметьте третью стрелку на нижнем изображении):



Большинство контекстов контейнеров в корне видимого системного дерева **отображаемые**, т.е. узел `Queries` показывает доступные запросы всех пользователей и т.д.

Однако контексты групп, расположенные в контекстах этих контейнеров, не **отображаемые**, т.е. узел `Query Groups`, расположенный в корне узла `Queries`, показывает только группы, доступные для текущего пользователя. Например, чтобы посмотреть группы запросов, доступные другим пользователям, следуйте `Users > User > Queries > Query Groups`.

3.1.3 Репликация данных контекста

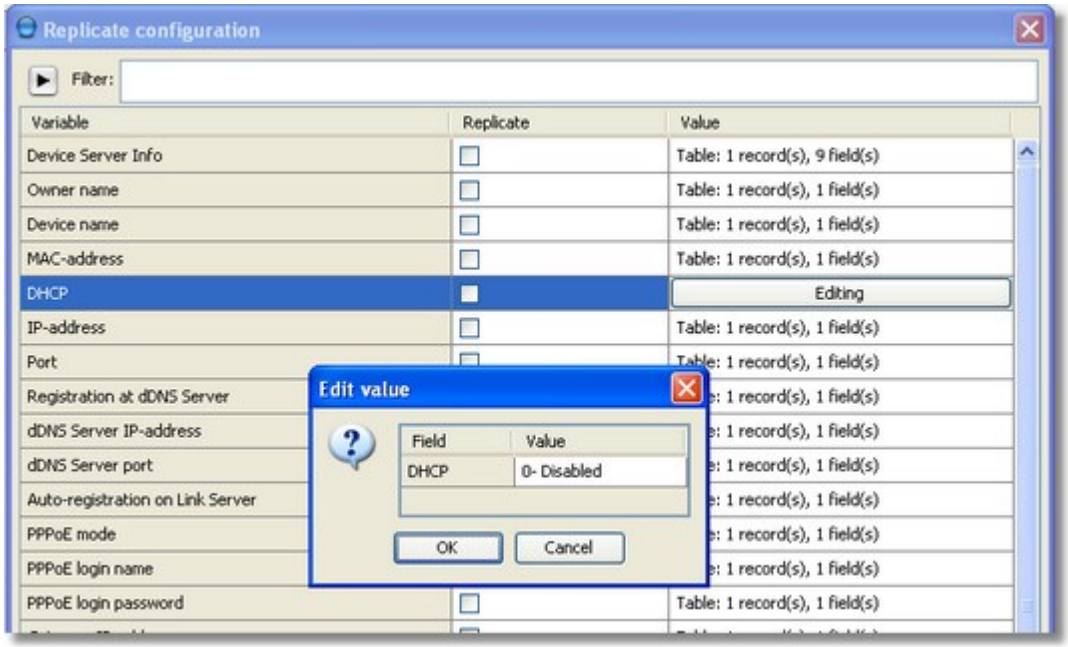
Операция *репликации данных контекста* используется для настройки одного контекста по аналогии с другим. Эту операцию можно запустить через пользовательский интерфейс AtomMind Client, например, путем перетаскивания одного узла системного дерева 370 в другой узел того же типа.

Внутри эта операция модифицирует значения переменных 61 (свойства) *целевого* контекста 41 (цель копирования) в соответствии со значениями в *источнике*. Данное значение копируется из контекста источника только если переменная, которая будет ее содержать (т.е., переменная с тем же именем), уже существует в целевом контексте. Операция репликации протекает следующим образом:

- Если переменная не читается в контексте источника, операция с этой переменной отменяется
- Если переменная не записывается в контексте источника, операция также отменяется
- Сервер читает значение переменной из контекста источника в таблице данных 49 **A**
- Затем он читает значение переменной из целевого контекста в таблице данных **B**
- Данные из таблицы данных **A** копируются в таблицу данных **B** при помощи операции интеллектуальное копирование таблиц данных 58
- Сервер переписывает значение переменной в целевой контекст таблицы данных **B**

Если на каком-либо этапе случается ошибка, операция с текущей переменной отменяется. Процесс копирования переходит к следующей переменной.

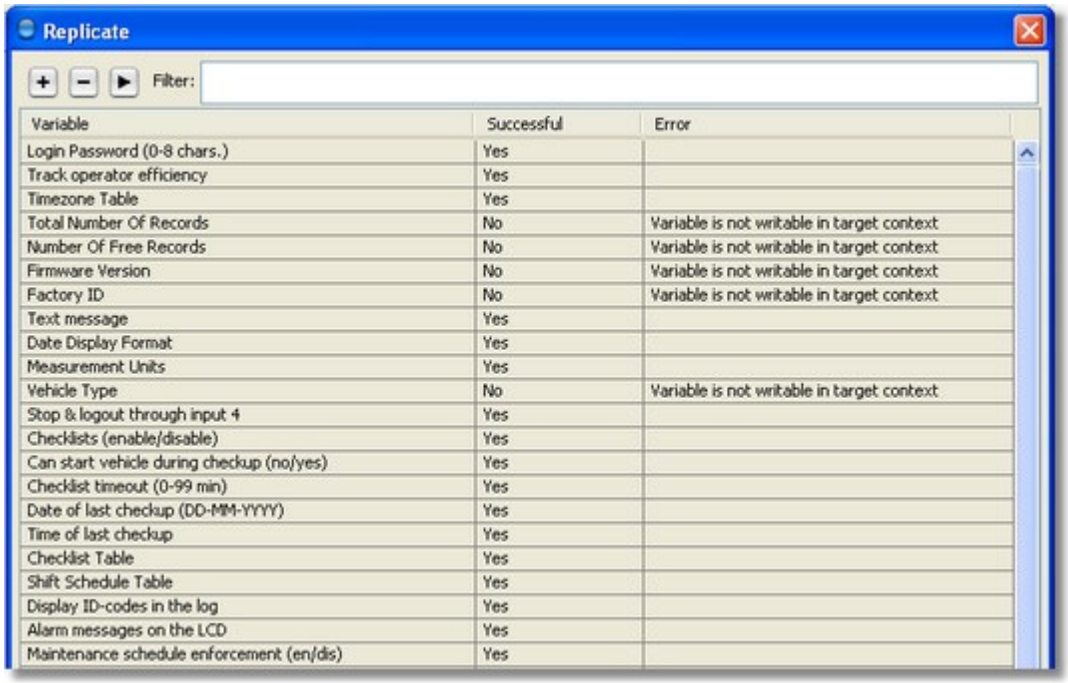
Сразу перед началом операции пользователи могут выбирать 90 **переменные** для копирования. Также возможно не разрешить репликацию определенных **полей.** Дополнительно можно скорректировать новое **значение** каждой переменной:



Операция копирования возвращает отчет в следующем формате:

Описание переменной	Копирование прошло успешно (да или нет)	Список ошибок, возникших во время копирования этой переменной

Отчет выглядит следующим образом:



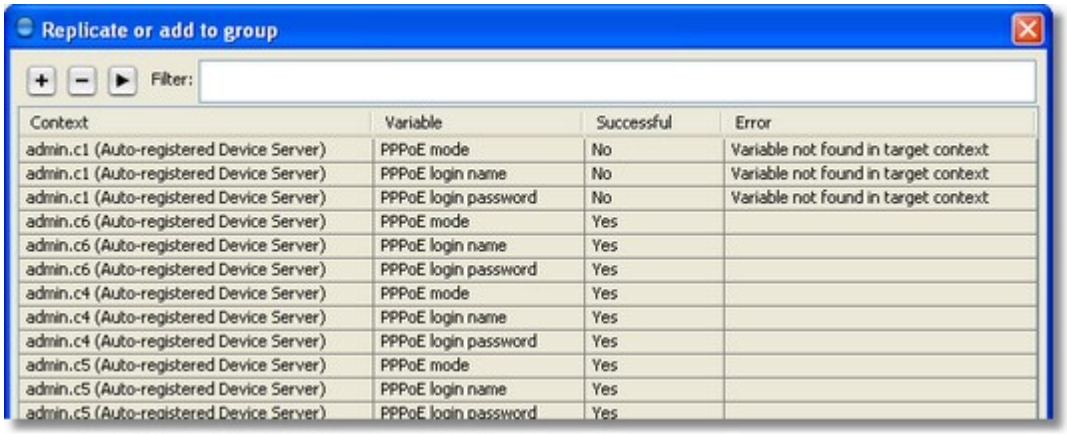
Настройка копирования в потомки

Операция *Копирование в потомки* подобна обычной операции копирования контекста с той разницей, что она копирует значения переменных из контекста источника в каждый потомок целевого контекста, а не в сам целевой контекст. Эту операцию можно начать, например, путем перетаскивания узла системного дерева[370], представляющего некий контекст, в узел, представляющий группу контекстов того же типа. В этом случае все контексты в группе будут сконфигурированы так же, как перемещенный объект.

Операция копирования в потомки выдает отчет в следующем формате:

Целевой контекст	Имя переменной	Копирование прошло успешно (да или нет)	Список ошибок, возникших во время копирования этой переменной

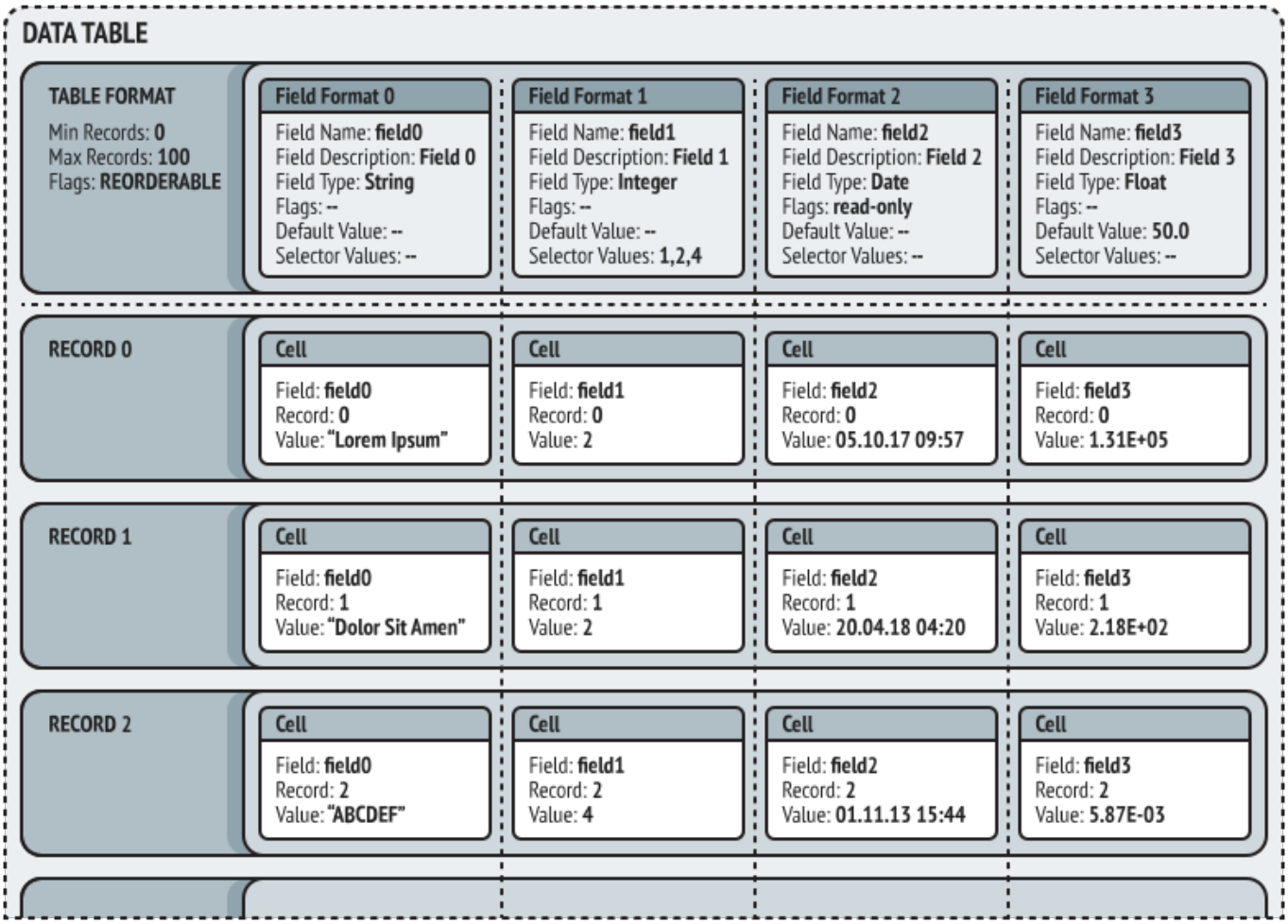
Пример отчета:



3.2 Таблицы данных

Таблица данных является основным типом данных в AtomMind.

Каждая таблица данных может содержать нуль или больше *записей* (рядов, строк) и нуль или больше *полей* (столбцов) и, следовательно, (количество записей * количество столбцов) *ячеек* с данными. Записи (строки) для определенной таблицы описываются Форматом таблицы 49. Все записи в Таблице данных всегда одного формата. Формат таблицы описывает каждое поле таблицы и содержит несколько других опций.



Таблицы с одним столбцом и несколькими строками используются для представления массивов. Таблицы с одной строкой и несколькими столбцами представляют структуры данных.

Даже скалярные типы данных, такие как одна строку или число, представлены Таблицами данных с одной записью и одним полем. Это может показаться странным, на первый взгляд, однако такая концепция значительно упрощает сложные операции.

Ячейки таблиц данных могут содержать вложенные таблицы. Это позволяет единственной таблице со вложенными таблицами представлять структуру данных любой сложности.

Применение Таблиц данных

Таблицы данных широко используются в AtomMind. Можно выделить три основных случая применения:

- **Значения переменных.** Значение каждой переменной[61] в каждом контексте[41] AtomMind Server представляется Таблицей данных. Ее формат[49] задается определением переменной[61].
- **Входные и возвращаемые параметры функции.** Значения входных параметров и возвращаемых значений для каждой функции[70] в контексте AtomMind Server представлены в Таблице данных. Их формат[49] задается определением переменной[61].
- **Данные события.** Каждое событие[73] контекста AtomMind Server имеет отдельную Таблицу данных. Эта таблица содержит данные, относящиеся к событию. Ее формат задан определением события[73].

3.2.1 Формат таблицы

Формат таблицы содержит полную информацию о таблице. Он может существовать отдельно от Таблицы данных. Вы можете создать различные Таблицы, используя один и тот же формат. Когда вы добавляете новую (пустую) запись в такую таблицу, каждое ее поле будет изначально содержать значение по умолчанию, как указано в Формате таблицы. Если поле определено как "пустое", значение также может быть NULL (т.е. значения не будет).

Формат таблицы обычно используется в следующих случаях:

- Для определения формата переменной[61] контекста,
- Для определения формата значений входных параметров и возвращаемых значений функций[70] контекста,
- Для определения формата Таблицы данных, относящейся к событию[73].

Формат таблицы определяет следующие свойства таблицы:

Свойство	Описание
Минимальное и максимальное количество записей	Количество записей должно быть ограничено пределом, иначе, таблица будет рассматриваться как *неправильная* (*invalid).*
Сортировка разрешена	Флажок, указывающий, что записи в таблице могут быть переставлены местами (перестроены) во время обработки таблицы
Допустимые значения записей	Правила проверки правильности каждой записи в таблице. Применяются в самой таблице и не могут быть изменены пользователем.
Допустимые значения таблицы	Правила проверки корректности таблицы целиком. Применяются в самой таблице и не могут быть изменены пользователем.
Привязки	Список привязок данных[741], определяющих как изменяются значения в таблице при изменении ячеек таблицы или переменных среды во время обработки таблицы.
Выражение наименования	Выражение[112], определяющее как будет представлена таблица для пользователей в интерфейсе.

Среда вычисления[114] выражения наименования таблицы данных:

Контекст по умолчанию[119]	Отсутствует.
Таблица данных по умолчанию[120]	Таблица вычисляемого выражения наименования.
Ряд по умолчанию[119]	0

Переменные среды 123	Только стандартные 123 переменные.

Формат поля

Формат поля состоит из нескольких дескрипторов *Формата поля*, которые описывают каждое поле. Большинство из них никогда не конфигурируются пользователем (или администратором). Они являются внутренними настройками, и упоминаются здесь для дополнительной информации.

Свойства поля:

Свойство	**Описание**
Name	Имя поля.
Description	Текстовое описание поля.
Help	Подробное описание поля.
Default Value	Значение поля по умолчанию. Вы можете это пропустить, если поле является пустым (см. далее).
Nullable	Флажок, указывающий, что поле может содержать значения *NULL ("Неопределенные")*
Non-Replicatable	Флажок, указывающий, что поле должно быть пропущено во время операции Умного копирования таблицы данных
Key Field	Флажок, обозначающий поле как ключевое. Ключевые поля приводят к тому, что операция Умное копирование таблицы данных использует специальный алгоритм копирования.
Selection Values	Список значений, которые могут быть выбраны для поля вместе с текстовыми описаниями.
Extendable Selection Values	Флажок, указывающий, что поле может быть установлено на любое значение, включая те, которых нет в списке в Значениях выборки.
Advanced	Поле, помеченное как Advanced, изначально будет скрыто для просмотра/редактирования таблицы в Редакторе таблиц данных 382, кроме случая, когда значение поля отличается от значения по умолчанию. Чтобы отобразить дополнительные поля, на панели Редактора таблиц данных есть специальная кнопка.
Validators	Список *валидаторов*, используемых для проверки, является ли значение подходящим для поля. AtomMind Server представляет несколько заранее сконфигурированных валидаторов, используемые внутренне системой для проверки элемента.
Editor/Renderer	Данный параметр определяет, как поле отображается и изменяется в пользовательском интерфейсе.
Options	Опции редактора/отрисовщика, позволяющие настраивать отрисовку значений поля в пользовательском интерфейсе. Синтаксис строк опций для различных типов редактора/отрисовщика описаны здесь 52.
Icon	ID строки иконки поля.
Group	Понятное для человека описание группы полей.
Inline	Этот флажок имеет особые значения для разных полей: • Если флажок Встраиваемый установлен на поле Блок Данных, значение поля будет сохранено в отдельной таблице базы данных 706. Значение не будет изначально загружено, если сама таблица загружается из базы данных и отправляется в AtomMind Client или внешнее приложение. Однако значение Блока Данных будет содержать идентификатор блока данных, который можно использоваться для запуска загрузки данных Блока Данных из базы данных сервера. Это подходит для больших бинарных значений, таких как изображение, звук и видеофайлы. • Если флажок Встраиваемый установлен на поле Цвет, значение поля будет использоваться для выделения записей таблицы, когда она просматривается/редактируется в Редакторе Таблиц Данных 382. Возможно совместить флажок Встраиваемый с флажком Скрытый, чтобы спрятать в таблице само поле Цвет. • Если флажок Встраиваемый установлен на поле Таблица данных, значение поля по умолчанию будет иметь тот же формат, что и таблица, которой принадлежат эти поля.

	Тем самым, это делает возможным создание таблиц с неограниченным числом вложений при помощи Редактора таблиц данных 382.
Encrypted	Активирует шифрование значения поля на уровне базы данных 692.

Типы полей

Существуют несколько предопределенных типов полей, которые могут появиться в Таблицах данных:

Тип поля	Описание
Integer field	Содержит 32-битное целое число со знаком.
Long field	Содержит 64-битное целое число со знаком. Поля типа Long часто используются для содержания временных периодов, выраженных в миллисекундах.
String field	Содержит строку (неограниченной длины).
Boolean field	"Флажок" с двумя возможными значениями: TRUE или FALSE.
Float field	Содержит 32-битное число с плавающей запятой.
Double field	Содержит 64-битное число с плавающей запятой. В большинстве случаев, поле типа Double не должно быть использовано преимущественно по отношению к полю типа Float.
Date field	Содержит временную метку с точностью до миллисекунд. Может рассматриваться как дата, время или временная метка (дата+время).
Data Table field	Содержит вложенную Таблицу данных.
Color field	Содержит определение цвета (RGBA).
Data Block field	Содержит двоичные данные, которые могут рассматриваться как типичный файл, изображение, звук и т.д.

3.2.2 Редакторы/Отрисовщики

В этом разделе перечисляются все редакторы и отрисовщики 49 полей таблицы данных, поддерживаемые AtomMind.

Подходящие типы полей	Код редактора/отрисовщика	Описание	Опции
Все типы полей	list	**Редактор списка**. Отрисовывает поля, у которых возможные значения представлены в виде списка зависимых кнопок вместо обычного комбинированного списка. Этот редактор не будет работать для полей, у которых включен флаг Расширяемые допустимые значения.	Не разрешены.
Date	date	**Редактор даты.** Позволяет задать только дату для полей типа Date. Установка времени невозможна.	Временная зона для визуализации данных. Пользовательские временные зоны могут быть указаны в форме строки, например: • GMT-8 • GMT-08:00 • Америка/Лос_Анджелес

Date	time	**Редактор времени.** Позволяет задать только время для полей типа Дата. Установка даты невозможна.  Редактор времени не сохраняет значения года, месяца, дня и милисекунды любой редактируемой временной метки. Эти части временной метки могут быть установлены на любые незаданные значения после редактирования.	Временная зона для визуализации данных. Пользовательские временные зоны могут быть указаны в форме строки, например: • GMT-8 • GMT-08:00 • Америка/Лос_Анджелес
Integer	spinner	**Счетчик.** Отрисовывает счетчик рядом с числовым текстовым полем, позволяющий увеличивать/уменьшать значения при помощи мыши.	Не разрешены.
Integer, Long, Float, String	bar	**Отрисовщик индикатора выполнения.** Отображает индикатор выполнения, указывающий текущее значение поля. Самое большое значение индикатора определяется Опциями редактора.	Строка опций будет интерпретироваться как числовое целое максимальное значение для индикатора, т.е. его верхний предел. Нижний предел всегда нулевой. Если опции не определены, максимальное значение нужно принимать как 100. Пример: 10000. Эта строка опций заставит индикатор отрисовывать значения в размере от 0 до 10000.
Integer, Long, Float, String	bytes	**Отрисовщик байтов**. Интерпретирует значение поля как количество байтов или скорость передачи байтов и отрисовывает его как: • количество байтов, килобайтов, мегабайтов, гигабайтов ИЛИ • количество байтов, килобайтов, мегабайтов или гигабайтов в секунду	Строка опций будет интерпретирована как Целое значение и использована в следующих случаях: • 0 - значения поля будут отрисовываться как байты, килобайты и т.д. • 1 - значения поля будут отрисовываться как байты, килобайты и т.д. • 2 - значения поля будут умножаться на 8 и отрисовываться как байты, килобайты и т.д.
Long	period	**Редактор временного периода.** Позволяет задать временной период в виде комбинации Единицы времени (от миллисекунд до года) и количества Единиц времени, вводимых в виде числа. Конечным значением поля будет количество миллисекунд.	Строка опций должна иметь следующую форму: MIN_UNIT MAX_UNIT. MIN_UNIT и MAX_UNIT - минимальные и максимальные типы целых единиц, используемые для отрисовки или редактирования периода. Доступные единицы описаны здесь 2160, единицы Неделя и Квартал не поддерживаются. Пример: 1 3. Эта строка опций позволяет использовать только секунды, минуты и часы для отрисовки/выбора периода. Милисекунды будут опущены. Значения больше одного часа будут отрисовываться в часах.
Long, String	foreignInstance	**Экземпляр внешнего класса**. Позволяет хранить ссылку на конкретный экземпляр класса 885	• **Контекст хранилища**. Контекст хранилища 1447 для просмотра объектов/классов.

		в поле. Предоставляет метод выбора экземпляра с помощью просмотра, поиска, сортировки и фильтрации списка экземпляров выбранного класса. При использовании данного редактора, значение поля содержит идентификатор экземпляра класса, заданный в опциях редактора.	• **Просмотр**. Определяет, какой вид просмотра 888 будет использован для определения показываемого списка колонок, правил фильтрации и порядка сортировки. Доступно только для классов 885. Если выбран Пользовательский просмотр, это возможно - определить опции **Класс**, **Колонки**, **Фильтр** и **Сортировка**. • **Таблица**. Тип/имя объекта, чьи экземпляры могут быть выбраны. Большинство контекстов хранилища предоставляют доступ к одному объекту, в этом случае в этом поле будет доступен только один параметр. • **Поле ссылки**. Определят, какая колонка (поле) будет использоваться как внешний ключ. • **Колонки**. Определяет, какие колонки (поля) будут показаны в диалоге выбора экземпляров. • **Фильтр**. Определяет фильтр 889 экземпляра. В выражениях фильтра можно использовать переменную окружения '{env/sourceInstance}', представляющую экземпляр исходного класса, из которого мы ссылаемся. • **Сортировка**. Определяет порядок сортировки. • **Инструментальная панель**.Определяет, какая инструментальная панель 912 должна быть открыта, если сделан клик по экземпляру выбранного класса. Выбранная инструментальная панель должна быть настроен для визуализации экземпляров **Класса**. • **Иконка**. Определяет идентификатор иконки, которая должна быть показана после экземпляра выбранного класса.
Integer, Float, Double, String	instance	**Экземпляр класса**. Позволяет хранить ссылку на конкретный экземпляр класса 885 в поле. Предоставляет метод выбора экземпляра с помощью просмотра, поиска, сортировки и фильтрации списка экземпляров выбранного класса. При использовании данного редактора, значение поля содержит идентификатор экземпляра класса, заданный в параметрах класса.	• **Контекст хранилища**. Контекст хранилища 1447 для просмотра объектов/классов. • **Таблица**. Тип/имя объекта этих экземпляров можно выбирать. Большинство контекстов хранилища предоставляют доступ к отельному объекту, в этом случае только одна функция будет доступна в этом поле. • **Инструментальная панель**.Определяет, какая инструментальная панель 912 должна быть открыта, если сделан клик по экземпляру выбранного класса. Выбранная инструментальная панель должна быть настроен для визуализации экземпляров **Класса**.

			• **Иконка**. Определяет идентификатор иконки,которая должна быть показана после экземпляра выбранного класса.
String	expression	**Редактор выражения.** Позволяет ввести AtomMind выражение при помощи Редактора выражения.	• **Контекст по умолчанию**. Контекст по умолчанию, который будет использован для отладки выражения. • **Таблица по умолчанию**. Таблица по умолчанию, которая будет использована для отладки выражения. • **Ссылки**. Ссылки, которые будут доступны для вставки с помощью одного клика во время редактирования выражения. • **Ожидаемый результат**. Текстовое объяснение того, что должно возвращать выражение. • **Описание контекста по умолчанию**. Текстовое объяснение того, что должно будет контекстом по умолчанию во время оценки выражения. • **Описание таблицы по умолчанию**. Текстовое объяснение того, что будет таблицей по умолчанию во время оценки выражения.
String	password	**Редактор пароля.** Редактор пароля представляет собой текстовое поле, которое заменяет все символы на "*".	Не разрешены.
String	text	**Текстовый редактор.** Использует полнофункциональный редактор текстов с поддержкой выделения синтаксиса для редактирования текстовых строк.	Строка опций будет интерпретироваться как режим выделения синтаксиса, одна из следующих: `aggregate` - выражение AtomMind `html` - разметка HTML `java` - код Java `shellscript` - сценарий оболочки Unix `smi-mib` - синтаксис файла SNMP MIB `sql` - запрос SQL `xml` - разметка XML
String	html	**Отрисовщик HTML.** Отрисовывает текст ячеек как разметку HTML.	Строка опций будет интерпретироваться как целое число символов для показа в основной таблице (т.е. до тех пор, пока текст не откроется в отдельном диалоге). По умолчанию отображается 30 символов. Пример: `100`. Эта строка опций покажет в таблице первые 100 символов строки.
String	textarea	**Редактор текстовой области**. Открывает текстовую область в отдельном диалоге для редактирования текстовых строк. Дает больше пространства для	Строка опций будет интерпретироваться как некое целое числовое количество символов для отображения в главной таблице (т.е. пока текст открывается в отдельном

		редактирования длинных значений и имеет возможность прокрутки.	диалоге). По умолчанию, показывается 30 символов. Пример: 100. Эта строка опций покажет в таблице первые 100 символов строки.
String	etextarea	**Отрисовщик встроенной текстовой области**. Использует компонент текстовой области, появляющийся внутри ячейки таблицы, для отрисовки текстовых значений в несколько строк.	Строка опций будет интерпретироваться как целое числовое максимальное количество символов для отображения в каждой строке текстовой области. Если текст длиннее, к текстовой области будет добавлено больше строк. По умолчанию их 20. Пример: 50. Эта строка опций вызовет отрисовку текстовой области с достаточной шириной для отображения 50 символов и достаточной высотой для вмещения всего текста.
String	context	**Редактор контекста.** Позволяет пользователю задать путь контекста.	• **Корневой элемент**. Выбор контекстов будет разрешен только из контекстов, находящихся ниже этого корневого контекста. • **Типы контекстов**. Если хотя бы один тип указан, редактор контекста позволит выбрать только контексты определенных типов. • **Маски контекстов**. Если хотя бы одна маска указана, редактор контекста позволит выбрать только контексты, соответствующие любой их казанных масок, и их родительские контексты.
String	contextmask	**Редактор маски контекста.** Позволяет пользователю задать маску контекста.	Те же, что выше.
String	reference	**Отрисовщик ссылки.** Позволяет пользователю начать операцию, кликнув по ссылке. Выполняемая операция и ее параметры определены в параметрах редактора.	• **Внешний вид**. Стиль ссылки: Ссылка или Кнопка. • **Тип ссылки**. Тип операции: Запись переменной, Вызов функции, Формирование события или Выполнение действия. В большинстве случаев, требуемый тип - это Действие. • **Тип контекста**. Определяет, каким образом система выбирает контекст для вызова операции при клике по ссылке. Может быть Статическим, когда определен особый **Контекст** во время конфигурации ссылки, либо Динамическим, когда путь контекста узнается при помощи вычисления **Выражения контекста**. • **Контекст**. Путь контекста, из которого вызывается операция. Доступно, если **Тип контекста** Статический. • **Выражение контекста**. Выражение, возвращающее путь контекста, из которого происходит вызов операции. Доступно, если **Тип контекста** Динамический.

			Среда вычисления 114 контекста выражения: Контекст по умолчанию 119 — Контекст по умолчанию текущего редактора. Таблица данных по умолчанию 120 — Просматриваемая/изменяемая в редакторе таблица данных. Строка по умолчанию 119 — Строка, содержащая выбранную ссылку. Переменные среды 123 — Только стандартные 123 переменные. • **Тип объекта**. Определяет, каким образом система выбирает объект (переменная/функция/событие/действие), который будет изменен, если был клик по ссылке. Может быть Статическим, когда определен особый **Объект** во время конфигурации ссылки, либо Динамическим, когда имя объекта вычислено при помощи оценки **Выражения объекта**. • **Объект**. Имя объекта для использовании в операции, напр. имя действия, которое нужно выполнить. Доступно, если **Тип объекта** Статический. • **Выражение объекта**. Выражение, которое возвращает имя объекта для использования в операции, напр. имя действия, которое нужно выполнить. Доступно, если **Тип объекта** Динамический. Среда вычисления 114 выражения объекта: Контекст по умолчанию 119 — Контекст по умолчанию текущего редактора. Таблица данных по умолчанию 120 — Просматриваемая/изменяемая в редакторе таблица данных. Ряд по умолчанию 119 — Ряд, содержащий выбранную ссылку. Переменные среды 123 — Только стандартные 123 переменные. • **Параметры объекта**. Определяет, какие параметры будут переданы операции. Например, если Тип ссылки - Действие или Функция, то будет использован список параметров объекта для заполнения таблицы параметров ввода действия или функции.

			• **Иконка**. Определяет идентификатор иконки, которая должна быть показана после текста ссылки.
String	font	**Редактор шрифта**. Позволяет выбрать шрифт из списка шрифтов, установленных на машине, где запущен AtomMind Client.	Не разрешены.
String	ip	**Редактор IP-адреса**. Позволяет ввести IP-адрес. Имена хостов не разрешены.	Не разрешены.
Color	box	**Отрисовщик прямоугольников**. Обычно цветовые значения отрисовываются как маленькие цветные прямоугольники вместе с числовыми RGB-значениями цвета. Отрисовщик прямоугольников не показывает числовые RGB-значения и вместо этого отображает большой закрашенный прямоугольник.	Не разрешены.
Data Block	dtext	**Текстовый редактор блока данных.** Тот же редактор, что и указанный выше, однако, используемый для редактирования поля типа блок данных в качестве текста.	Оставлены для внутреннего пользования.
Data Block	image	**Редактор изображения.** Позволяет вставлять и просматривать изображения в полях типа Data Block.	Оставлены для внутреннего пользования.
Data Block	sound	**Редактор звука.** позволяет вставлять и прослушивать звуки в полях типа блок данных.	Оставлены для внутреннего пользования.
Data Block	hex	**Редактор шестнадцатеричных чисел**. Позволяет просматривать/редактировать отдельные байты блоков данных в шестнадцатеричном формате. При редактировании блока данных также возможно поменять размер блока путем ввода нового размера (в байтах) внутри текстового поля.	Не разрешены.

3.2.3 Интеллектуальное копирование таблиц

Операция *Интеллектуальное копирование таблиц данных* используется для слияния данных из одной таблицы данных 49 (*источника*) с данными в другой таблице данных (*цели*). Оно называется "интеллектуальным" не просто так. Допустим, у вас имеется свойство "Клиенты разрешены" в каком-то контексте (например, вымышленный контекст "AccessControl"). Для данного конкретного контекста переменная относится к бинарному типу - может быть либо Да (разрешить клиенты), либо Нет (не разрешать).

А теперь представим, что у вас есть другой контекст, например, (вымышленный) контекст "OfficePrinter", у которого также есть свойство с названием ClientsAllowed. Только в этот раз это целое число, представляющее *количество* клиентов, которые могут подключаться к этому принтеру или опрашивать его. Это не бинарное значение. Но оно имеет такое же название! (Что часто случается в разнотипной, открытой системе)

Итак, если серверная операция копирования по умолчанию взяла свойства и просто скопировала их поверх базового имени, вскоре возникнет проблема. Нужно было воспользоваться "умным" копированием значений свойств, не нарушая систему, и поэтому мы разработали Интеллектуальное Копирование Таблиц Данных.

Вообще говоря, операция Копирование Таблиц Данных копирует как можно большее количество данных из исходной таблицы в целевую, но всегда контролирует, чтобы целевая таблица сохраняла свой оригинальный формат.

Например, если определение формата целевой таблицы указывает, что максимальное количество записей 5, в таблице будет 5 записей после операции копирования, даже если в исходной таблице было 100 записей.

Другой пример: если исходная и целевая таблицы содержат поле под названием "значение", но в исходной таблице тип значения Строка, а в целевой таблице тип значения Целое, операция копирования не изменит тип поля в целевой таблице. Вместо этого операция попытается конвертировать все строки из исходной таблицы в целые и записать их в целевую таблицу.

Операция копирования не отменяется, если появляются ошибки при копировании некой записи или поля. Вместо этого операция продолжается, стараясь скопировать как можно больше данных. Отчет об ошибках приходит после завершения операции.

Процесс копирования выбирает один из следующих способов:

- Копировать с ключевыми полями 59
- Копировать без ключевых полей 59

Ключевое поле - это поле или набор полей в таблице, которые вместе составляют уникальный идентификатор для записи (элемента таблицы). Совокупность этих полей обычно просто называется "ключом".

Первый способ выбирается, если формат 50 целевой Таблицы Данных определяет ненулевое количество ключевых полей. Если ключевого поля нет, используется второй способ.

При любом способе система контролирует, что количество записей в целевой таблице не превышает минимальные и максимальные значения (определенные форматом определение переменной 61).

Копировать с ключевыми полями

Если в целевой таблице есть Ключевые Поля, которые не определены как Ключевые Поля в исходной таблице, операция переходит в режим копировать без ключевых полей 59.

Операция копирования делится на три этапа. На каждом этапе происходит работа с записями, имеющими одинаковые значения во всех Ключевых Полях в обеих таблицах (совокупное значение всех ключевых полей называется "ключом").

1. Все записи в целевой таблице, содержащие ключи, которых нет в исходной таблице, удаляются.

2. Данные из каждой записи исходной таблицы копируются 59 в запись целевой таблицы с тем же ключом, если он есть.

3. Для каждой оставшейся записи исходной таблицы создается новая запись в целевой таблице, и данные из исходной записи копируются 59 в новую запись.

Последовательность записей в целевой таблице не меняется в процессе копирования с ключевыми полями. Все записи с ключами, которых нет в целевой таблице, будут добавляться в конец целевой таблицы, независимо от их положения в таблице-источнике.

Копировать без ключевых полей

Этот способ копирования также происходит в три этапа.

1. Если количество записей в целевой таблице больше, чем в исходной таблице, последние записи в целевой таблице удаляются, чтобы сравнять количество записей.

2. Каждая запись исходной таблицы копируется 59 в запись целевой таблицы с тем же номером.

3. Если количество записей в исходной таблице больше, чем в целевой таблице, в целевой таблице создаются новые записи, чтобы сравнять количество записей. Данные из исходной записи копируются 59 в новую запись с тем же номером.

Алгоритм копирования записи данных

При копировании исходной записи (одна строка в таблице) в запись назначения, каждое поле в строке тестируется, чтобы понять, может ли оно быть скопировано:

1. Если поле заявлено как "только для чтения" в формате целевой записи, оно пропускается.

2. Если поле заявлено как нереплицируемое в формате источника целевой записи, оно пропускается.

3. Если поле имеет тот же тип в исходных и целевых записях, его значение копируется напрямую.

4. Если у поля различные типы в исходных и целевых записях, AtomMind Server старается конвертировать значение. Например, все типы значений могут конвертироваться в значение Строка, значения Строка могут иногда конвертироваться в Целое, и т.д.

3.2.4 Построение таблиц из списка параметров

Бывает необходимо создать таблицу данных из списка параметров, закодированных в строки и разделенных запятой, т.е. `"param1", 'param2', null`. Например, этот метод используется для вызова функции из ссылки 118 или запроса 829.

AtomMind использует общий алгоритм для посторения таблицы данных из списка параметров, закодированных в строки. Этот алгоритм использует типы полей, определенных в формате 50, для конвертирования параметров строки в определенные значения поля и различные ячейки в таблице данных. Этот формат может быть, например, форматом ввода функции, для которой создается вводная таблица данных.

Проще говоря, система знает формат таблицы, которая создается, берет значения строк из исходного списка по одному, конвертирует их в определенные типы и заполняет ячейки таблицы.

Параметры в списке должны разделяться запятыми (","**,**") и могут содержать следующие типы значений:

Значение	Пример	Описание
Строка без кавычек, строка с одинарными кавычками	`'{nested_reference} + 1'` `{nested_reference} + 1`	Кавычки извлекаются из строки, если она заключена в кавычки. Результирующее значение обрабатывается как выражение 112, которое оценивается, и результат оценки используется в качестве значения ячейки в таблице входных параметров функции. Если вы не уверены в том, каково *выражение*, можно представить его как формулу динамической таблицы (1+1, например). Глава язык выражений AtomMind 112 объясняет все аспекты.
Строка с двойными кавычками	`"123"`	Кавычки извлекаются из строки. Результирующее значение не обрабатывается как выражение -- наоборот, оно конвертируется в любой тип данных, который функция ожидает для этого аргумента. Конверсия производится по принципу "наилучшие усилия" -- вы не всегда сможете конвертировать любой тип данных в любой другой тип данных.

Правила кодирования/декодирования значений различных типов в строки/из строк можно найти здесь 2125.

Каждая ячейка в этой таблице получает свое значение из соответствующего аргумента в списке аргументов (т.е. значение первой ячейки - это значение первого аргумента в списке).

Если создаваемая таблица данных имеет *динамический (неопределенный) формат*, система выстраивает таблицу данных в одну строку со всеми строковыми полями. Каждый исходный параметр помещается в отдельное поле.

Очень редко функция может ожидать больше одного ряда полей (т.е., больше одной "записи") в качестве ввода. Если формат ввода функции определяет N полей, первые N аргументов используются для заполнения первой строки, вторые N аргументов для второй и т.д.

Пример: Допустим, у вас есть функция **func1** (произвольное имя), которая ожидает следующую таблицу данных ввода с двумя полями и двумя рядами:

string_field	integer_field
"abc"	123
"xxx"	456

Впоследствии вы можете вызывать ее из ссылки 118 или запроса 829 следующим образом: **func1("abc", "123", "xxx", "456")**. Поэтому, поскольку у вас есть два параметра на каждую строку, первые два параметра будут использоваться для заполнения первого ряда. Следующие два параметра будут использоваться для заполнения второго ряда и т.д.

Отметим, что даже при том, что целые значения выше имеют двойные кавычки, **func1("abc", 123, "xxx", 456)** выдаст тот же результат. Разница в том, что целые значения будут обрабатываться как выражения (которые состоят из простых целых литерал), а сама обработка может происходить чуть медленнее.

3.3 Переменные

Переменные содержат данные, относящиеся к контексту 41. В AtomMind Server, значение каждой переменной представляет собой *Таблицу данных* 49. Даже простые скалярные (например, числовые и строковые) значения представлены в виде таблиц данных, состоящих из одной ячейки. Это помогает использовать общий подход при обработке значений переменных. Все переменные можно обрабатывать при помощи стандартных операций с таблицами (таких как интеллектуальное копирование таблицы данных).

Переменные контекста иногда называются *свойствами.* Эти два слова взаимозаменяемы в языке AtomMind. Они означают одно и то же.

Примеры

В контексте Пользователь 1608 находится переменная **childInfo** ("Информация о пользователе"). Ее значение всегда содержит одну запись (как обусловлено Форматом переменной) с несколькими полями, в которых содержится информация об имени, фамилии пользователя, его номере телефона и т.д.

В контексте Фильтр событий 1509 находится переменная **rules** ("Правила фильтра"). Ее значение может включать несколько записей ("строк"), каждая из которых описывает тип события, которое должно отображаться во время активации фильтра событий 762.

Определение переменной

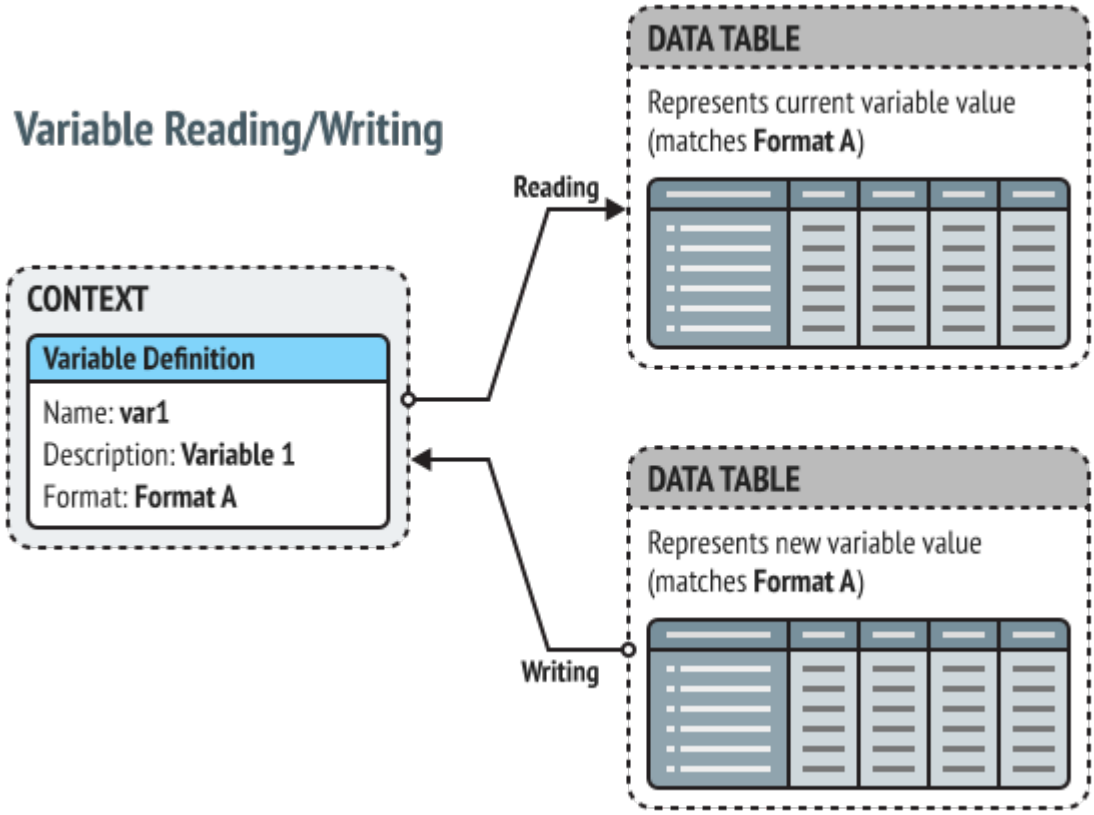
Каждая переменная задается при помощи *Определения переменной*, которое содержит несколько опций:

- **Имя переменной.** Имя - это уникальный ID переменной внутри контекста, в котором переменная определена. Представляет собой строку, в состав которой могут входить только английские буквы, числа и символ нижнего подчеркивания ("_").
- **Формат переменной.** Формат 50 таблицы данных 49 для переменной. Определяет, как может выглядеть переменная - минимальное и максимальное количество рядов, типы полей, возможные значения полей, правила пригодности и т.д. Некоторые переменные имеют *динамический* формат, т.е. их формат может быть представлен любой Таблицей данных.
- **Доступна для чтения.** Флажок, указывающий на то, что можно получить значение переменной из контекста. В По факту, абсолютное большинство переменных читаемы.
- **Доступна для записи.** Флажок, указывающий, что значение переменной можно заменить на новое.
- **Описание.** Удобное для чтения описание переменной.
- **Справка.** Подробное описание переменной (опционально).
- **Группа.** Указывает на то, что переменная принадлежит *группе переменных*. Группа может быть не определена, т.е. в неё могут входить переменные, которые не принадлежат ни к одной группе. Группы помогают выбирать количество переменных для обработки. Например, контекст Устройства может включать в себя действие 87 Конфигурировать для настройки Учетной записи Device, и другое действие для настройки аппаратного устройства. Каждое действие будет оперировать всеми переменными отдельной группы. Имя группы может быть задано заранее на AtomMind Server или получено от какого-либо аппаратного устройства, однако, изменить его нельзя.
- **Разрешения на чтение.** Необходимый уровень 486 прав доступа для получения значения данной переменной.
- **Разрешения на запись.** Необходимый уровень 486 прав доступа для изменения значения данной переменной.

Чтение и запись

Успешная операция чтения переменной возвращает текущее значение переменной в форме таблицы данных. Эта таблица всегда соответствует формату переменной, если он не динамический.

Операция записи переменной принимает новое значение переменной в форме таблицы данных с тем же форматом, или в форме любой таблицы данных, если формат переменной динамический. Если полученная таблица данных не полностью соответсвует формату переменной, система делает все возможное, чтобы перевести полученную таблицу в нужный формат, сохраняя как можно больше данных.



AtomMind Server обрабатывает переменные различными способами. Если вы передаете таблицу данных как значение переменной, вы необязательно получите обратно ту же самую таблицу, когда попытаетесь извлечь значение переменной.

Внутренне переменные разделены на несколько типов:

- Структуры в памяти, которые не сохраняются при остановке %LS%>.
- Долговременные настройки, которые сохраняются в базе данных 49.
- Настройки устройств, которые читаются/записываются с удаленного устройства и кэшируются сервером.
- "Виртуальные" переменные. Значение виртуальных переменных формируется на лету и не сохраняется постоянно.

Если при установке или получении значений переменных возникли проблемы, может появиться сигнал *исключения*. В случае выполнения операции "получить переменную", никакое значение не возвращается. Операция "Задать переменную" может вызвать исключение по какой-то другой причине, даже если значение переменной действительно было задано. Исключение содержит текстовое описание проблемы.

Работа с переменными

В большинстве случаев, значения переменных можно просматривать и редактировать при помощи компонента UI Редактор свойств 256. Редактор свойств использует компонент Редактор таблиц данных 282 для отрисовки и редактирования значения каждой отдельной переменной.

События обновления переменной

Каждый раз при обновлении переменной, генерируется особое событие Обновление 84. Данные события содержат новое значение переменной и время обновления. В некоторых случаях в AtomMind Server также генерируется событие Изменение 84.

Статус переменной

Переменные контекстов в AtomMind могут также иметь свой *статус*. Статус переменной представляет собой комбинацию кода статуса и его описания, удобного для чтения.

Статусы переменных отображаются в левом столбце Редактора свойств 377. Однако, существуют и другие способы увидеть статус переменной. Например, статусы переменных настроек устройства можно посмотреть в диалоговом окне Статус устройства 111.

3.3.1 Общие переменные

Этот раздел описывает переменные, общие для большинства контекстов[41] AtomMind.

3.3.1.1 info (Информация о контексте)

Данная переменная содержит основную информацию о контексте.

Имя переменной: info

Записи: 1

Права доступа: Доступна для чтения/записи на уровне[486] "не определен" (none).

Формат[49] записи:

Имя поля	Описание поля	Тип поля	Примечания
description	Описание контекста[43], т.е. текстовая информация о его назначении.	Строка	
type	Тип[43] контекста.	Строка	
group	Группа контекста. В AtomMind[684] должна быть NULL.	Строка	Может быть NULL
icon	ID иконки контекста. Используется для визуального представления контекста в пользовательском интерфейсе. Должно быть NULL в AtomMind[684].	Строка	Может быть NULL
localRoot	Используется в распределенной архитектуре[1332]. Путь корня точки монтирования в локальном (сервере-потребителе) дереве контекстов.	Строка	
remoteRoot	Используется в распределенной архитектуре[1332]. Путь корня точки монтирования в удаленном (сервере-поставщике) дереве контекстов. Если значение null, это контекст является локальным (т.е. не прокси из любого удаленного сервера-поставщика).	Строка	Может быть NULL
remotePath	Используется в распределенной архитектуре[1332]. Путь контекста в удаленном (сервере-поставщике) дереве контекстов.	Строка	
mapped	Возвращает значение true, если отображаются дочерние компоненты контекста (т.е. для контекстов групп и агрегирования).	Булевое	

3.3.1.2 children (Потомки контекста)

Данная переменная содержит список потомков контекста, одну запись на каждый контекст-потомок. При получении данной переменной, вы видите только те потомки, которые доступны при вашем уровне прав доступа[486], необязательно, что будут отображены все потомки.

Имя переменной: children

Записи: 0... не ограничено

Права доступа : Доступна для чтения/записи на уровне[486] "Не определен" (none).

Формат 49 записи:

Имя поля	Описание поля	Тип поля	Примечание
name	Имя контекста-потомка.	Строка	

3.3.1.3 variables (Переменные котекста)

Данная переменная содержит основную информацию о переменных, доступных в контексте. При получении этой переменной, вы видите только определения переменных, доступных для вашего уровня прав доступа 486, а не все существующие переменные. Более подробную информацию о полях определения переменных см. в разделе Переменные 61.

Имя переменной: variables

Записи: 0... не ограничено

Права доступа: Доступна для чтения/записи на уровне 486 "Не определен" (none).

Формат 49 записи:

Имя поля	Описание поля	Тип поля	Примечания
name	Имя переменной.	Строка	
format	Формат переменной, закодированный 2124 в строку. Если данное поле является NULL, формат переменной динамичен - может вернуть данные любого формата.	Строка	Может быть null
description	Описание переменной.	Строка	
readable	Флажок, указывающий, что переменная доступна для чтения при вашем уровне прав доступа.	Булевое	
writable	Флажок, указывающий, что переменная доступна для записи при вашем уровне прав доступа.	Булевое	
help	Подробное описание переменной.	Строка	Может быть null
group	Группа переменной или NULL, если переменная не принадлежит ни к какой группе.	Строка	Может быть null
iconId	ID иконки, используемой для отображения переменной (для внутреннего использования).	Строка	Может быть null
helpId	ID статьи документации с описанием переменной (для внутреннего использования).	Строка	Может быть null

3.3.1.4 functions (Функции контекста)

Данная переменная содержит основную информацию о функциях, доступных в контексте. При получении этой переменной, вы видите только определения функций, доступных для вашего уровня прав доступа 486, а не все существующие функции. Более подробную информацию о полях определения функций см. в разделе Функции 70.

Имя переменной: functions

Записи: 0... не ограничено

Права доступа: Доступна для чтения/записи на уровне 486 "Не определен" (none).

Формат 49 записи:

Имя поля	Описание поля	Тип поля	Примечания
name	Имя переменной.	Строка	
inputformat	Формат входа функции, закодированный 2124 в строку. Если данное поле является NULL, формат входа функции динамичен - он может принимать данные любого формата.	Строка	Может быть null
outputformat	Формат выхода функции, закодированный 2124 в строку. Если данное поле является NULL, формат выхода функции динамичен - он может возвращать данные любого формата.	Строка	Может быть null
description	Описание функции.	Строка	
help	Подробное описание переменной.	Строка	
group	Группа 61 переменной или NULL, если переменная не принадлежит ни к какой группе.	Строка	Может быть null

3.3.1.5 events (События контекста)

Данная переменная содержит основную информацию о событиях, доступных в контексте. При получении этой переменной, вы видите только определения событий, доступных для вашего уровня прав доступа 486, а не все существующие события. Более подробную информацию о полях определения функций см. в разделе События 73.

Имя переменной: events

Записи: 0... не ограничено

Права доступа: Доступна для чтения/записи на уровне 486 "Не определен" (none).

Формат 49 записи:

Имя поля	Описание поля	Тип поля	Примечания
name	Имя события.	Строка	
format	Формат данных события, закодированных 2124 в строку. Если данное поле является NULL, формат события динамичен - оно может содержать данные любого формата.	Строка	Может быть null
description	Описание события.	Строка	
help	Подробное описание события.	Строка	
level	Уровень события.	Строка	
group	Группа 61 события или NULL, если оно не принадлежит ни к какой группе.	Строка	Может быть null

3.3.1.6 actions (Действия контекста)

Эта переменная содержит основную информацию о действиях 87, доступных в контексте. Когда вы получаете эту переменную, вы видите лишь определения действий, которые доступны вам с нынешним уровнем прав доступа 486, необязательно все ваши существующие действия.

Имя переменной: actions

Записи: 0... не ограничено

Права доступа: Доступно для чтения/записи на на уровне 486 "Не определен" (none).

Формат 50 записи:

Имя поля	Описание поля	Тип поля	Примечания
name	Имя действия.	Строка	
description	Описание действия.	Строка	Может быть null
help	Подробное описание действия.	Строка	Может быть null
accelerator	Клавиша быстрого доступа к дествию.	Строка	Может быть null
dropSources	Правила активации операции перетаскивания.	Таблица данных	Может быть null
hidden	Флажок видимости действия.	Булевое	
enabled	Флажок доступности действия.	Булевое	
iconId	ID иконки действия.	Строка	Может быть null
group	Группа действия.	Строка	Может быть null
executionGroup	Группа выполнения действия.	Строка	Может быть null
default	Флажок, определяющий, является ли это действие действием по умолчанию в данном контексте.	Булевое	

3.3.1.7 contextStatus (Статус контекста)

Эта переменная содержит информацию о статусе контекста. Для отслеживания изменения статуса контекста, подпишитесь на событие contextStatusChanged 83.

Имя переменной: contextStatus

Записи: 1

Права доступа: Доступно для чтения/записи на на уровне 486 "Не определен" (none).

Формат 50 записи:

Имя поля	Описание поля	Тип поля	Примечания

status	Числовой код статуса, специфичный для контекста.	Целое	Может быть null
comment	Текстовое описание статуса контекста.	Строка	Может быть null

3.3.1.8 activeAlerts (Активные тревоги)

Эта переменная предоставляет информацию об экземплярах тревог[779], которые в настоящее время активны для контекста.

Имя переменной: activeAlerts

Записи: 0... не ограничено

Права доступа: Доступно для чтения/записи на уровне[486] *Нет прав.*

Формат[50] записи:

Имя поля	Описание поля	Тип поля	Примечания
location	Определяет местоположение тревоги в распределенной инсталляции[1332]. **Локальная** значит, что активная для данного контекста тревога определяется на сервере-потребителе, в то время как **Удаленная** указывает, что тревога определяется на удаленном сервере-поставщике.	Integer	
event	ID события тревоги[790].	Long	Скрытое поле.
alert	Путь контекста тревоги[1454].	String	
type	Тип экземпляра: активный[804] и ожидающий подтверждения[804].	Integer	
time	Время возникновения тревоги.	Date	
level	Уровень тревоги.	Integer	
message	Сообщение тревоги.	String	
trigger	Сообщение триггера тревоги.	String	
cause	Причина тревоги.	String	
data	Данные о тревоге.	Data Table	
acknowledgements	Подтверждения тревоги.	Data Table	
enrichments	Обогащения тревоги.	Data Table	

3.3.1.9 groupMembership (Членство в группах)

Данная переменная предоставляет информацию о группах[751], в которых участвует текущий контекст.

Имя переменной: groupMembership

Записи: 0... не ограниченно

Права доступа: Доступна для чтения/записи на уровне[486] "Не определен" (none).

Формат 49 записи:

Имя поля	Описание поля	Тип поля	Примечания
location	Определяет расположение группы в распределенной инсталляции 1332. **Локальный** значит, что этот контекст является членом группы, определенной на сервере-потребителе, в то время как **Удаленный** указывает, что этот контекст добавляется к группе, расположенной на удаленном сервере-поставщике.	Целое	
group	Полный путь контекста группы 1529, в состав которого входит текущий контекст.	Строка	

3.3.1.10 validity (Пригодность)

Эта переменная приводит список контекстов, для которых пригоден 749 текущий контекст.

Имя переменной: validity

Записи: 0... не ограничено

Права доступа: Доступно для чтения на на уровне 486 прав доступа *нет прав* (none).

Формат 50 записи:

Имя поля	Описание поля	Тип поля	Примечания
context	Полный путь контекста 1529, для которого пригоден текущий контекст.	Строка	

3.3.1.11 templates (Шаблоны)

Эта переменная предоставляет информацию о шаблоне контекста и переменных, значения которых будут использоваться для текущего контекста.

Имя переменной: templates

Записи: 0... не ограничено

Права доступа: Доступна для чтения/записи на уровне 486 "Не определен" (none).

Record Format 50:

Имя поля	Описание поля	Тип поля	Примечания
variable	Имя переменной, значение которой будет определено шаблоном.	Строка	
context	Контекст 1529 шаблона.	Строка	
expression	Выражение для расчета значения текущей переменной.	Строка	Может быть null

	В случае для пустого выражения в качестве значения текущей переменной будет взято значение переменной с таким же именем у контекста шаблона. Иначе, значением переменной будет результатом вычисления выражения. Контекстом по умолчанию для вычисляемого выражения будет контекст шаблона.		
ignoreFields	Список полей у текущей переменной, которые будут игнорироваться при установке нового значения из переменной контекста шаблона.  Поля **name** и **description** для переменных **genericProperties** и **childInfo** игнорируются по умолчанию.	Таблица данных	Может быть null

3.3.2 История переменных

Значения переменных контекста часто обновляются. Значения могут вручную меняться оператором или меняются автономными компонентами системы (такими как запланированные задачи 823). При самом простом раскладе новые значения получаются из аппаратных устройств во время синхронизации 514.

Когда значение переменной контекста меняется, AtomMind Server может опционально сохранять его старое значение, чтобы позволить отследить исторические изменения переменной.

Исторические значения могут позже использоваться для:

- Отображения исторических трендов на графиках 105 (таких как график температуры за последний день)
- Построение отчетов 928 (таких как отчет о ежемесячном потреблении топлива)
- Эскпортирование их в сторонние системы

 Подробно о том, как работать с историей переменных, см. в разделе Работа с историческими событиями/значениями 745.

Необработанная история и статистика

AtomMind Server обеспечивает два способа сохранения исторических значений:

- Сохранение необработанной истории в "классической" NoSQL или реляционной базе данных (см. база данных сервера 692)
- Сохранение собранной истории в кольцевой базе данных (см. модуль статистики 718)

Каждый из этих методов имеет свои "за" и "против".

НЕОБРАБОТАННАЯ ИСТОРИЯ

В режиме хранения "необработанной истории" каждое историческое значение переменной сохраняется как отдельная сущность запись в базе данных сервера 692.

Плюсы этого метода:

- Точность старых значений, независимо от частоты дискретизации

Минусы этого метода:

- Большой объем исторических данных
- Медленное сохранение
- Медленное извлечение исторических значений, особенно при загрузке образцов, собираемых на протяжении долгого периода времени

СТАТИСТИКА

Когда история переменной хранится в циклической базе данных статистики 718, ее старые значения не сохраняются "как есть". Вместо этого сервер считает числовой "агрегированный" результат каждого значения и

использует его для расчета средних, минимальных и максимальных значений каждого часа/дня/недели/месяца/года.

Плюсы этого метода:

- Компактное хранение исторических данных
- Размер сохраняемых исторических значений не увеличивается со временем
- Быстрое сохранение
- Необычайно быстрый доступ к иторическим значениям, независимо от длительности периода

Минусы этого метода:

- Доступны только агрегированные исторические значения, исходные "необработанные" значения не сохраняются
- Точность рассчитанных агрегированных результатов уменьшается со временем

3.3.3 Производительность переменных

Производительность операций чтения и записи переменной значительно зависит от "природы" переменной и базовых алгоритмов. Вот некоторые примеры:

- Извлечение значения кэшированной 502 переменной настройки устройства очень быстрое, даже миллионы операций чтения в секунду не будут иметь значительного влияния на загрузку процессора. То же самое относится к записи нового значения переменной, поскольку действительный ввод/вывод устройства будет отложен, а вызов вернется асинхронно.
- Извлечение свойств компонентов виджета 955 также очень быстрое.
- Чтение переменной дата 1548 из контекста Запрос вызывает выполнение запроса, включая соответствующую загрузку процессора и памяти 842.
- Если было запрошено значение любой *удаленной* переменной (например, AtomMind Client запросил его у AtomMind Server или сервер-потребитель запросил его у сервера-поставщика в распределенной архитектуре 1332), потребуется дополнительное время для отправки соответствующего запроса удаленному серверу и получения ответа. Это время в целом равно кольцевому времени сети, полученному при помощи команды `ping`.

3.4 Функции

Функции представляют собой операции, которые могут выполняться в контексте 41. Каждая функция имеет *входное* и *выходное* значение. Оба значения по сути являются Таблицами данных 49. Каждая из этих Таблиц данных может включать в себя несколько строк и полей и даже иметь вложенные Таблицы данных внутри ячеек, поэтому количество возможных параметров входа и вывода функции практически не ограничено.

Определение функции

Каждая функция определяется при помощи *Определения* функции. Определение содержит несколько опций:

- **Имя функции.** Имя является уникальным в контексте, в котором определена функция. Представляет собой строку, которая может содержать только английские буквы, цифры и символ нижнего подчеркивания ("_").
- **Входной формат.** Формат 50 Таблицы данных 49, используемый для определения значения входных параметров функции.
- **Выходной формат.** Формат таблицы данных, используемый для определения значения выходных параметров функции.
- **Описание**. Удобное для восприятия описание функции.
- **Помощь**. Подробное описание функции (опционально).
- **Права доступа**. Уровень 486 прав, необходимый для вызова функции.
- **Группа**. Показывает, что функции принадлежащие к *группе функций*. Группа помогает объединять похожие функции во время выполнения различных операций.

Примеры

Контекст **Пользователи** 1604 имеет функцию под названием **list** ("Список пользователей"), которая возвращает список всех Учетных записей 478 на сервере, доступных для пользователя, который вызывает функцию.

Контекст **Запланированные задачи** 1585 имеет функцию под названием **create** ("Запланировать новую задачу"), которая создает новую задачу 823 с определенными параметрами.

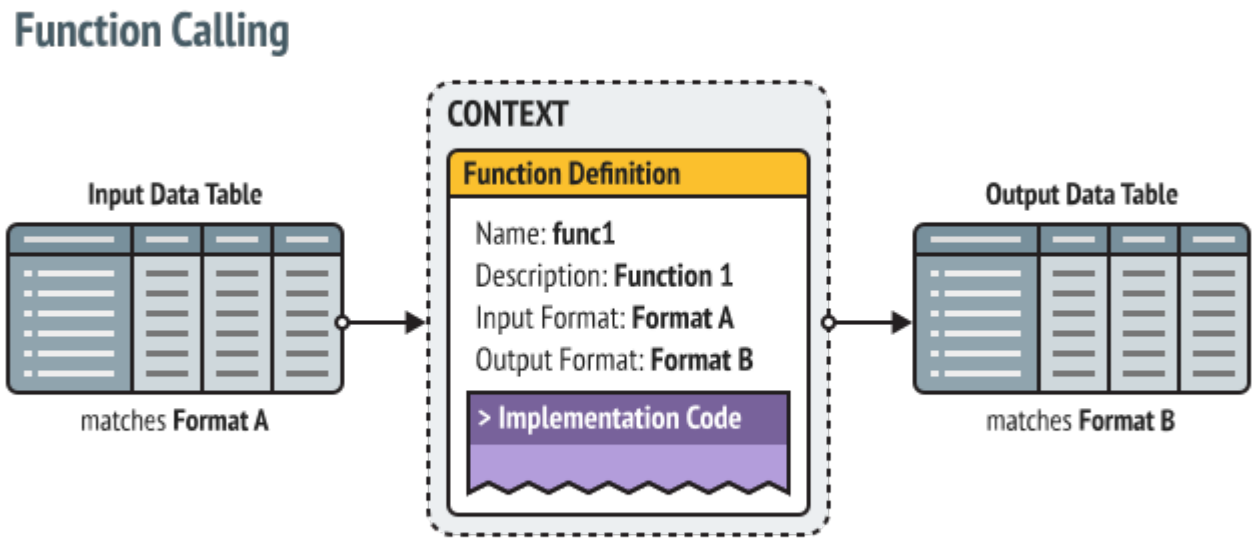
Контекст **users.admin.devices.dev1** может содержать функция **reboot** ("Перезагрузить"), которая перезагружает Устройство **dev1**.

Обратите внимание, что функции не интерактивны сами по себе. Они являются внутренними операциями сервера - их выполнение происходит "за кулисами". Чтобы предоставить параметры входа для функции (или посмотреть возвращаемые параметры), используются интерактивные Действия 87 с их UI процедурами 88.

Реализация функции

Разные функции реализуются по-разному:

- Некоторые функции ведут себя согласно логике, применяемой в ядре платформы или в плагине 207
- Некоторые функции делегируют свою логику внешнему устройству
- Некоторые функции делегируют свою логику малокодовому алгоритму, такому как набор правил



Если функция не может завершить работу, она выдаст *исключение*, содержащее текстовое описание проблемы.

3.4.1 Общие функции

Этот раздел описывает функции, которые являются общими для большинства контекстов 41 AtomMind.

3.4.1.1 makeCopy (Копировать)

Данная функция используется для копирования системного объекта, например, Тревогу 779. Она создает копию определенного контекста под текущим контекстом, т.е. тем, где она была определена.

Имя функции: makeCopy

Права доступа: Доступна на уровне 486 прав доступа *Менеджер*

Записи входа: 1

Формат входа: 49

Имя	Тип	Описание
context	Строка	Путь копии создаваемого контекста под текущим контекстом.

Записи выхода: 0... не ограничено

Формат выхода: 49

Имя	Тип	Описание
variable	Строка	Имя переменной, чье значение было реплицировано.
successful	Булевое	Статус репликации, успешно или нет.
errors	Строка	Список ошибок репликации.

3.4.1.2 delete (Удалить)

Данная функция используется для безвозвратного удаления потомка текущего контекста.

Имя функции: delete

Права доступа: Доступна на уровне 486 прав доступа *Менеджер*

Записи входа: 1

Формат входа: 49

Имя	Тип	Описание
name	Строка	Имя удаляемого потомка.

Записи выхода: 0

Формат выхода: 49 не задан

3.4.1.3 updateVariable (Обновить переменную)

Данная функция используется для операции обновления значения переменной контекста.

Имя функции: updateVariable

Права доступа: Доступна на уровне 486 прав доступа *Менеджер*

Записи входа: 1

Формат входа: 49

Имя	Тип	Описание

variable	Строка	Имя переменной
expression	Строка	Вычисляет аргумент ее строки как выражение AtomMind 112 и результат в *переменную*. Значение *переменной* используется как *табица по умолчанию* во время вычисления выражения.

Записи выхода: 0

Формат 49 выхода: не задан

Функция возвращает новое значение переменной.

3.4.2 Производительность функций

Поизводительность вызова функции значительно зависит от "природы" функции и базового алгоритма. Вот несколько примеров:

- Функция синхронизации 1495 контекста устройства очень быстрая, посколько она просто планирует синхронизацию 514 и возвращается.
- Функция выполнить запрос 1559 будет в целом иметь значительное влияние на производительность, что будет зависеть от сложности запроса и размера пакета данных источника.
- Вызов функции, соответствующей работе устройства, занимает время, требующееся для ввода/вывода устройства, но не вызывает загрузку процессора со стороны сервера.
- Если вызывается *удаленная* функция (т.е. AtomMind Client вызвал ее из AtomMind Server или сервер-потребитель вызвал ее из сервера-поставщика в распределенной среде 1332), потребуется дополнительное время для отправки соответствующего запроса на удаленный сервер и получения ответа. Это время в целом равно кольцевому времени сети, полученному при помощи команды `ping`.

3.5 События

Событие возникает в определенном контексте 41, если что-либо происходит в данном контексте. Например, событие в контексте устройства 497 может произойти, если было получено оповещение от аппаратного устройства. Каждое событие в AtomMind Server имеет специальную связанную с ним структуру под названием *Таблица данных* 49. Эта таблица содержит данные, относящиеся к событию. Например, событие **login** может включать в себя одну запись с полем **username**, в котором содержится имя пользователя, выполнившего вход.

Определение события

Каждое событие определяется в своем контекста при помощи *Определения события*. Определение содержит несколько опций:

- **Имя события.** Уникальное имя в контексте, где определяется событие. Является строкой, которая может включать в себя только английские буквы, цифры и символ нижнего подчеркивания ("_").
- **Формат события**. Формат 50 Таблицы данных 73, представляющей данные о событии.
- **Описание**. Удобное для чтения описание события.
- **Помощь**. Подробное описание события (опционально).
- **Срок хранения**. Определяет время, в течение которого сохраняемое событие хранится в истории 73.
- **Уровень**. Критичность события. Данный уровень является уровнем по умолчанию для события, однако, появившееся в контексте событие может иметь другой уровень критичности. Список доступных уровней критичности приведен в раздедле Уровни событий 75.
- **Права доступа**. Уровень 486 прав доступа, необходимый для получения событий данного типа.
- **Группа**. Указывает на то, что событие принадлежит к *группе событий*. Группа помогает объединить схожие экземпляры событий во время выполнения различных операций.

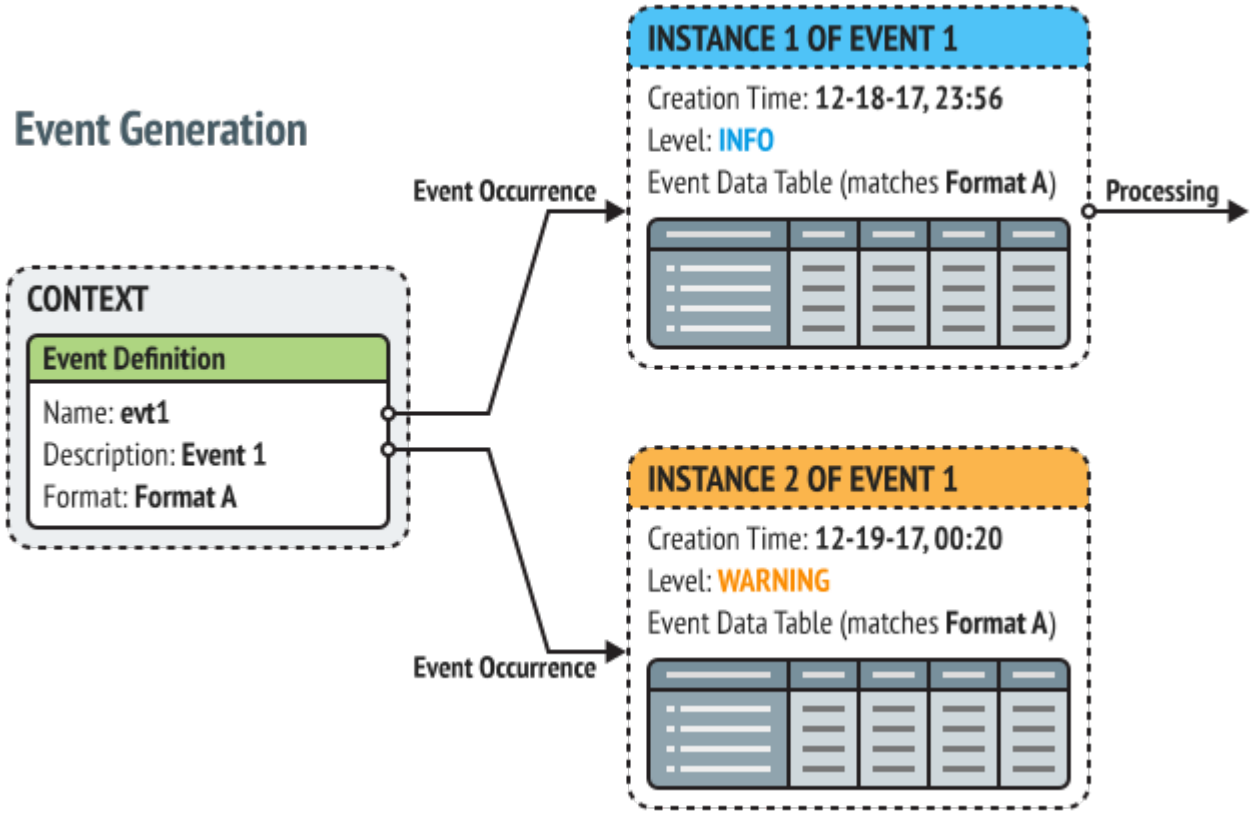
Примеры

Контекст Тревога 1454 имеет событие **alert** ("Тревога"), которое активируется при возникновении тревоги 779.

Корневой контекст[1559] AtomMind Server имеет событие **login** ("Вход пользователя в систему"), которое активируется при входе пользователя[478] на сервер.

Экземпляры событий

Каждый раз, когда в контексте происходит событие определенного типа, создается отдельный *экземпляр события*. Этот экземпляр проходит через жизненный цикл[77] события.



3.5.1 Параметры экземпляров событий

Каждое событие имеет следующие параметры:

ID	Уникальный ID события
Время создания	Время, когда произошло событие. Если событие пришло из внешнего источника, время его создания может устаналиваться этим источником. Таким образом, время создания может оказаться в будущем относительно местного времени AtomMind Server, если часы источника события или сервера неправильные.
Время истечения	Время, когда событие будет удалено из постоянного хранилища[692] Сервера (только для постоянных событий).
Контекст	Контекст[41], в котором сработало событие
Уровень	Уровень[75] события.
Данные	Специфичные для события данные в форме таблицы данных[49] (см. ниже Таблица данных события).
Подтверждения	Подтверждения[76] событий (только для постоянных событий).
Enrichments	Обогащения[76] событий (только для постоянных событий).

Таблица данных события

Каждое событие имеет прикрепленную Таблицу данных. Эта таблица различается для каждого типа событий и содержит информацию, относящуюся к данному типу события. Можно сказать, что она "примечания" или "характеристики" конкретного типа событий.

Например, событие **login** имеет поле **username**, указывающее, какой пользователь выполнил вход. Данное поле используется только для события **login** -- нет смысла применять его, к примеру, в событии **alert**. Таким образом, оно находится в Таблице данных для события **login** (а в таблице данных события **alert** такого поля не существует).

Если в таблице данных события имеются привязки 741, эти привязки обрабатываются до сохранения события в базе данных 692 сервера.

3.5.2 Уровни событий

Уровни определяют важность события. Когда происходит событие, его уровень может быть присвоен либо компонентом, запускающим событие, либо выставлен на некое значение по умолчанию, указанное в Определении События. Существует несколько заранее определенных уровней события:

- **Не определен** (неопределенная важность)
- **Уведомление** (наименее важное)
- **Информационный**
- **Предупреждение**
- **Ошибка**
- **Критическая ошибка** (наиболее важное)

Числовые значения уровней события (могут использоваться разными выражениями 112, т.е. во время фильтрации событий 765):

Уровень	Значение
Не определен	0
Уведомление	1
Информационный	2
Предупреждение	3
Ошибка	4
Критическая ошибка	5

3.5.3 Слушатели событий

Если компонент AtomMind Server "хочет" узнать, что происходит в определенном контексте, он *подписывается* на события в контексте. Для этого, необходимо зарегистрировать специальный приемник *событий контекста*. Данный приемник получает *уведомление*, когда в контексте появляется выбранное событие.

Например, компонент Журнал событий 398 в AtomMind Client регистрирует приемники событий в контекстах, определяемых активным Фильтром событий 762. При возникновении данных событий, Журнал получает уведомления, которые он использует для отображения новых событий.

Приемники событий могут быть динамично добавлены и удалены во время работы сервера.

3.5.4 История событий

Постоянные события сохраняются в *истории событий*. История событий является таблицей базы данных 692, которая содержит все постоянные события, созданные различными компонентами AtomMind Server. Другие компоненты получают доступ к истории событий через контекст **events**. Например, компонент Журнал событий в AtomMind Client использует данный контекст для отображения истории событий.

Каждое событие, хранимое в истории, имеет свой *срок действия*. Когда срок действия события истекает, оно автоматически удаляется из истории. Срок действия по умолчанию - 10 дней, но его можно изменить, редактируя таблицу Сроки действия событий 196 в опциях Глобальной конфигурации AtomMind Server.

Подробно о том, как работать с историей событий, см. в разделе Работа с историческими данными/значениями 745.

3.5.5 Подтверждение событий

Каждое долговременное событие может быть подтверждено один и более раз. Подтверждение представляет собой текстовую строку, которая вводится и просматривается через компонент Журнал событий 398 в AtomMind Client. Пользователь также может ввести текст подтверждения во время всплывания сообщения о тревоге 779 в AtomMind Client. Если тревога требует подтверждения, в всплывающем окне для пользователя будет предложено ввести текст подтверждения. Подтверждение состоит из временной метки, имени пользователя (того, который подтвердил тревогу) и текста подтверждения.

Пример: Событие *Сбой сервера* может быть подтверждено системным оператором, добавившим сообщение "Заменена дефектная флешка".

Подтверждения обычно создаются пользователями. Они полезны для ведения учета - вы всегда можете быть уверены в том, что оператор видел событие, а также определить, какой именно был оператор, когда был просмотр события и какие комментарии введены для него.

Таблица подтверждений событий

Перейти на таблицу подтверждений событий можно из различных выражений 112 фильтров событий. Она имеет следующий формат:

Имя поля	Тип поля	Примечания
author	Строка	Имя создателя подтверждения.
time	Дата	Временная метка подтверждения.
text	Строка	Текст подтверждения.

3.5.6 Обогащение событий

Обогащение события - это процесс добавления важных параметров, полученных из внутренних ресурсов AtomMind или внешних систем, к экземплярам события. Каждый экземпляр события может иметь неограниченное количество *обогащений*, каждое из которых описано в строке *имя* и строке *значение*.

В отличие от подтверждений событий, обогащения обычно генерируются компонентами AtomMind и внешними системами, не пользователями.

Пример: AtomMind часто интегрируется с системами Service Desk, поэтому при появлении тревоги 779 AtomMind в Service Desk создается заявка. Внешняя система Service Desk может обогащать экземпляры событий тревоги 790 номерами этих заявок, чтобы операторы AtomMind могли быстро идентифицировать заявку, ассоциирующуюся с тревогой.

Обогащение события ассоциируется с определенными экземпляром события, не его типом. Каждый экземпляр события может обогащаться сразу же после его создания или в любое время позже.

Обогащение можно найти в Журнале событий 398:

- Включите столбец **Обогащения** в настройках фильтра событий 762 для просмотра обогащений всех событий.
- Кликните правой кнопкой на событии и выберите Просмотреть Обогащения для просмотра подробной информации об обогащениях определенного события.

Заново созданные экземпляры события могут автоматически обогащаться AtomMind Serverом. Для получения более подробной информации см. раздел Обогащения 196 в статье **Правила обработки событий**.

Таблица обогащений события

На таблицу обогащений события можно перейти из различных выражений 112 фильтрации событий. У нее следующий формат:

Имя поля	Тип поля	Примечания
name	Строка	Имя обогащения. Имя может содержать только английские буквы, цифры и нижние подчеркивания.
value	Строка	Содержание обогащения.
date	Дата	Временная метка обогащения.
author	Строка	Имя создателя обогащения.

3.5.7 Жизненный цикл событий

Эта статья описывает жизненный цикл события сервера. События, сгенерированные в Agent и AtomMind Client 359, имеют более простые жизненные циклы с меньшим количеством этапов.

1. Некоторые компоненты системы хотят запустить событие во время работы. Например, компонент, ответственный за коммуникацию с определенным Device (который называется *драйвером Device*) генерирует событие **info** в контексте этого Device, когда с ним теряется связь.
2. Проверяются права доступа 477 генератора событий. Если эти права доступа не позволяют сгенерировать событие этого типа в данном конкретном контексте, генерация события прекращается.
3. Если определяется выражение префильтра 196, оно оценивается. Если оно возвращается со значением false, генерация события не происходит.
4. Если определяется выражение идентификатора дедупликации 196, оно оценивается. Затем система пытается найти другое событие с тем же идентификатором дедупликации, и если оно находится, текущее событие опускается, но счетчик событий предыдущего увеличивается.
5. Событие обогащается 76 дополнительными данными, определенными пользователем.
6. Если событие определяется как постоянное, оно сохраняется в истории событий. Вы не можете сделать событие постоянным, если оно уже не определено подобным образом (большинство событий постоянные). Однако вы можете поменять "время жизни" события (его срок действия). Это делается путем изменения таблицы времен истечения события 196 в глобальной конфигурации AtomMind Server (см. раздел история событий 75).
7. Каждый зарегистрированный слушатель получает уведомление.

Диспетчеризация события

Последний шаг жизненного цикла событий - это диспетчеризация. На этой стадии каждый подписчик обрабатывает экземпляры события.

Пока все другие стадии обработки события появляются в потоке, который сгенерировал событие, диспетчеризация события работает в отдельном потоке, который называется *диспетчер события*. Исходный поток только добавляет событие в *очередь события* диспетчера.

Использование очередь и отдельный поток обрабатываемого события означает, что случатели могут получить доступ к событию позже, чем оно появляется. В зависимости от фактора загрузки сервера этот период отложения может достигать до сотен мс или даже секунд, если очередь дспетчера событий растет.

Если у сервера плохая работа, важно оценить длину очереди диспетчера события. Длина изначальной очереди события AtomMind Server's показывается в поле длина очереди события 1567 таблицы **статус сервера**.

3.5.8 Общие события

Этот раздел описывает события, которые являются общими для большинства контекстов 41 AtomMind.

3.5.8.1 info (Информация)

Данное событие возникает, когда что-либо происходит в контексте, и пользователю следует с этим ознакомиться. к этому относится, например, проблема синхронизации настроек между аппаратными устройствами и базой данных AtomMind Server (для контекста Терминал данных 1494). Формат 49 данного события представляет собой одно строковое поле "info", в котором содержится понятное пользователю описание события.

Событие **info** определено во всех контекстах AtomMind Server. Оно является долговременным, т.е. сохраняется в истории событий 73.

Имя события info

Права доступа: Доступно на уровне 486 прав доступа контекста.

Период дейтсвия: 1 неделя

Записи: 1

Формат 49 записи:

Имя поля	Тип поля	Примечания
info	Строка	Текст информационного сообщения.

Событие **info** имеет дедупликацию 762, включенную по умолчанию. Все последние экземпляры с тем же сообщением конвертируются в единый экземпляр, сохраняемый базе данных 692 сервера.

3.5.8.2 childAdded (Добавлен контекст-потомок)

Это событие возникает, когда к контексту добавляется новый контекст-потомок.

Это событие определяется во всех контекстах AtomMind Server. Оно непостоянное, т.е. не хранится в истории событий 75.

Имя события: childAdded

Права доступа: Доступно на уровне 486 прав доступа контекста

Период действия: Непостоянный

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
child	Строка	Имя контекста-потомка.

3.5.8.3 childRemoved (Удален контекст-потомок)

Это событие возникает, когда контекст-потомок удаляется из контекста.

Это событие определяется во всех контекстах AtomMind Server. Оно непостоянное, т.е. не хранится в истории событий 75.

Имя события: childRemoved

Права доступа: Доступно на уровне 486 прав доступа контекста

Период действия: Непостоянный

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
child	Строка	Имя потомка.

3.5.8.4 visibleChildAdded (Добавлен видимый контекст-потомок)

Это событие возникает, когда новый контекст стал новым видимым контекстом-потомком отображенного 45 или группового 1529 контекста.

Разница между событиями *childAdded* 78 и *visibleChildAdded* в том, что первый возникает в родительском контексте (где контекст-потомок физически создается и к которому принадлежит), а второй возникает в отображенном или групповом контексте (где контекст-потомок отобразился, но к которому не принадлежит).

Это событие определяется во всех контекстах AtomMind Server. Оно непостоянное, т.е. не хранится в истории событий 75.

Имя события: visibleChildAdded

Права доступа: Доступно на уровне 486 прав доступа контекста

Период действия: Непостоянный

Записи: 1

Формат 50 записи

Имя поля	Тип поля	Примечания
path	Строка	Путь контекста-потомка

3.5.8.5 visibleChildRemoved (Удален видимый контекст-потомок)

Это событие возникает, когда контекст удаляется из списка видимых контекстов-потомков отображенного 45 или группового 1529 контекста.

Разница между событиями *childRemoved* 78 и *visibleChildRemoved* в том, что первое возникает в родительском контексте (к которому физически относится контекст-потомок и из которого он удаляется), а второе возникает в отображенном или групповом контексте (к которому контекст-потомок не принадлежит, и в котором он стал невидимым).

Это событие определяется во всех контекстах AtomMind Server. Оно непостоянное, т.е. не хранится в истории событий 75.

Имя события: visibleChildRemoved

Права доступа: Доступно на уровне 486 прав доступа контекста

Период действия: Непостоянный

Записи: 1

Формат 50 записи

Имя поля	Тип поля	Примечания
path	Строка	Путь контекста-потомка

3.5.8.6 variableAdded (Добавлена переменная)

Это событие возникает, когда новое определение переменной добавляется к контексту.

Это событие определяется во всех контекстах AtomMind Server. Оно непостоянное, т.е. не хранится в истории событий 75.

Имя события: variableAdded

Права доступа: Доступно на уровне 486 прав доступа контекста

Период действия: Непостоянный

Записи: 1

Формат 50 записи: соответствует формату переменной переменные 64.

3.5.8.7 variableRemoved (Удалена переменная)

Это событие возникает, когда определение переменной удаляется из контекста.

Это событие определяется во всех контекстах AtomMind Server. Оно непостоянное, т.е. не хранится в истории событий 75.

Имя события: variableRemoved

Права доступа: Доступно на уровне 486 прав доступа контекста

Период действия: Непостоянный

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
name	Строка	Имя переменной.

3.5.8.8 functionAdded (Добавлена функция)

Это событие возникает, когда определение функции добавляется к контексту.

Это событие определяется во всех контекстах AtomMind Server. Оно непостоянное, т.е. не хранится в истории событий 75.

Имя события: functionAdded

Права доступа: Доступно на уровне 486 прав доступа контекста

Период действия: Непостоянный

Записи: 1

Формат 50 записи: соответствует формату переменной функции 64.

3.5.8.9 functionRemoved (Удалена функция)

Это событие возникает, когда определение функции удаляется из контекста.

Это событие определяется во всех контекстах AtomMind Server. Оно непостоянное, т.е. не хранится в истории событий 75.

Имя события: functionRemoved

Права доступа: Доступно на уровне 486 прав доступа контекста

Период действия: Непостоянный

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
name	Строка	Имя функции.

3.5.8.10 eventAdded (Добавлено событие)

Это событие возникает, когда новое определение события добавляется к контексту.

Это событие определяется во всех контекстах AtomMind Server. Оно непостоянное, т.е. не хранится в истории событий 75.

Имя события: eventAdded

Права доступа: Доступно на уровне 486 прав доступа контекста

Период действия: Непостоянный

Записи: 1

Формат 50 записи: соответствует формату переменной события 65.

3.5.8.11 eventRemoved (Удалено событие)

Это событие возникает, когда определение события удаляется из контекста.

Это событие определяется во всех контекстах AtomMind Server. Оно непостоянное, т.е. не хранится в истории событий 75.

Имя события: eventRemoved

Права доступа: Доступно на уровне 486 прав доступа контекста

Период действия: Непостоянный

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
name	Строка	Имя события.

3.5.8.12 actionAdded (Добавлено действие)

Это событие возникает, когда новое определение действия добавляется к контексту.

Это событие определяется во всех контекстах AtomMind Server. Оно непостоянное, т.е. не хранится в истории событий 75.

Имя события: actionAdded

Права доступа: Доступно на уровне 486 прав доступа контекста

Период действия: Непостоянный

Записи: 1

Формат 50 записи: соответствует формату переменной действия 66.

3.5.8.13 actionRemoved (Удалено действие)

Это событие возникает, когда определение действия удаляется из контекста.

Это событие определяется во всех контекстах AtomMind Server. Оно непостоянное, т.е. не хранится в истории событий 75.

Имя события: actionRemoved

Права доступа: Доступно на уровне 486 прав доступа контекста

Период действия: Непостоянный

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
name	Строка	Имя действия.

3.5.8.14 actionStateChanged (Изменено состояние действия)

Это событие возникает, когда состояние действия контекста меняется.

Это событие определяется во всех контекстах AtomMind Server. Оно непостоянное, т.е. не хранится в истории событий 75.

Имя события: actionStateChanged

Права доступа: Доступно на уровне 486 прав доступа контекста

Период действия: Непостоянный

Записи: 1

Формат 50 записи: соответствует формату переменной действия 66.

3.5.8.15 infoChanged (Изменена информация)

Это событие возникает, когда меняется информация основного контекста.

Это событие определяется во всех контекстах AtomMind Server. Оно непостоянное, т.е. не хранится в истории событий 75.

Имя события: infoChanged

Права доступа: Доступно на уровне 486 прав доступа контекста

Период действия: Непостоянный

Записи: 1

Формат 50 записи: соответствует формату переменной информация 63.

3.5.8.16 statusChanged (Изменен статус)

Данное событие возникает в контексте при смене его статуса (иными словами состояния 44). Например, в контексте Устройство 1492 данное событие появляется, когда устройство подключено, и его контекст меняет свой статус с "**Отключено**" на "**Подключено**".

Формат 49 данного события предполагает два поля:

- **status:** Целое число, указывающее новое состояния контекста.
- **comment:** Строка, содержащая удобное для чтения описание нового состояния.

Данное событие определено во всех контекстах AtomMind Server, которые имеют различные состояния 44. Оно является долговременным, т.е. сохраняется в истории событий 73.

Имя события: contextStatusChanged

Права доступа: Доступно на уровне 486 прав доступа контекста.

Период действия: 1 неделя для контекста групп 1529

1 неделя для контекста тревог 1454

1 день для контекста датчиков 1596

Непостоянный для всех других случаев

Записи: 1

Формат 49 записи:

Имя поля	Тип поля	Примечания
status	Целое	Новый статус контекста.

comment	Строка	Описание текущего статуса контекста.
oldStatus	Целое	Предыдущий статус контекста.

3.5.8.17 updated (Обновление переменной)

Данное событие появляется, когда изменяется значение какой-либо переменной[61], определенной в данном контексте.

Это событие сгенерировано на обновлении большинства переменных. Однако есть определенные переменные, чье значение сгенерировано на ходу в течение операции чтения. Каждое чтение такой переменной может повлиять на другое значение, но нет четких "данных обновления". Поэтому **обновленное** событие не будет сгенерировано для таких переменных.

Имя события updated

Права доступа: Доступно на уровне[486] прав доступа *Наблюдатель*.

Период действия: Непостоянный (см. событие изменить[84] для сохранения обновления переменной)

Записи: 1

Формат[49] записи:

Имя поля	Тип поля	Примечания
variable	String	Имя измененной переменной.
value	Data Table	Новое значение переменной.
valueOld	Data Table	Предыдущее значение переменной, либо null, если включение предыдущих значений в Обновленные события не активировано. Существует несколько настроек, определяющих, включено ли предыдущее значение в Обновленные события, например: • В устройствах[497] это параметр Синхронизации настроек Добавить предыдущее значение в событие обновления переменной[502] • В моделях[810] это параметр Переменных модели Добавить предыдущее значение в событие обновления переменной[817]
user	String	Имя пользователя[478], который запустил обновление переменной. Null, если обновление было инициировано неаутентифицированным модулем системы.
updateOriginator	Integer	Содержит информацию об инициаторе обновления переменной. Может быть: **0** - неопределенный инициатор **1** - процессор привязок[735] (т.е. переменная была изменена привязкой)

3.5.8.18 change (Изменение переменной)

Данное событие является аналогом "на уровне системы" обновленного[84] события. Событие "изменение" появляется при изменении значения какой-либо переменной[61], определенной в данном контексте. В отличие от события "обновление", которое предназначено для распространения изменений свойств другим модулям системы, событие "изменение" служит для

постоянного хранения значений свойств. История данного события внутренне используется различными компонентами системы для получения истории свойств.

Это событие сгенерировано на обновлении большинства переменных. Однако есть определенные переменные, чье значение сгенерировано на ходу в течение операции чтения. Каждое чтение такой переменной может повлиять на другое значение, но нет четких "данных обновления". Поэтому **обновленное** событие не будет сгенерировано для таких переменных.

Имя события: change

Права доступа: Доступно на уровне 486 прав доступа *Наблюдатель*.

Период истечения: Динамический, зависит от типа переменной значений, которые были обновлены

Записи: 1

Формат 49 записи:

Имя поля	Тип поля	Примечания
variable	Строка	Имя измененной переменной.
value	Таблица данных	Новое значение переменной.
data	Строка	Значение переменной, закодированное в строку без формата. Данное поле заполняется, только когда поле "value" является NULL.

3.5.8.19 evaluation (Вычисление)

Данное событие появляется, когда выражение вычисляется внутри контекста.

Событие определено во всех контекстах AtomMind Server. Оно непостоянное, т.е. не сохраняется в истории событий 75.

Имя события evaluation

Права доступа: Доступно на уровне 486 прав доступа контекста.

Период действия: Непостоянный

Записи: 1

Формат 49 записи:

Имя поля	Тип поля	Примечания
expression	String	Текст выражения.
holder	String	Путь элемента, содержащего выражение, внутри контекста.
result	Data Table	Результат выражения, инкапсулированный в таблицу с одной ячейкой, чтобы сохранить его тип.
details	Data Table	Детали вычисления, например, иерархическая таблица, содержащая результаты вычисления для каждой части синтаксического дерева выражения.

defaultContext	String	Контекст по умолчанию 119, используемый во время вычисления.
defaultTable	Data Table	Таблица по умолчанию 120, используемая во время вычисления.
defaultRow	Integer	Ряд по умолчанию 119, используемый во время вычисления.

3.5.8.20 evaluationError (Ошибка вычисления)

Данное событие появляется в случае ошибки внутри контекста при вычислении выражения.

Событие определено во всех контекстах AtomMind Server. Оно непостоянное, т.е. не сохраняется в истории событий 75.

Имя события evaluationError

Права доступа: Доступно на уровне 486 прав доступа контекста.

Период действия: Непостоянный

Записи: 1

Формат 49 записи:

Имя поля	Тип поля	Примечания
expression	String	Текст выражения.
holder	String	Путь элемента, содержащего выражение, внутри контекста.
message	String	Текс сообщения об ошибке при вычислении.
details	Data Table	Детали вычисления, например, иерархическая таблица, содержащая результаты вычисления для каждой части синтаксического дерева выражения.
defaultContext	String	Контекст по умолчанию 119, используемый во время вычисления.
defaultTable	Data Table	Таблица по умолчанию 120, используемая во время вычисления.
defaultRow	Integer	Ряд по умолчанию 119, используемый во время вычисления.

3.5.9 Производительность событий

AtomMind Server способен обрабатывать миллионы событий в секунду. Однако все события разные, и, если одно событие возникает без каких-либо последующих действий, другие сохраняются в базе данных 692 сервера и доставляются для удаления слушателей, вызывающих загрузку процессора и ввод/вывод сети. Также локальные слушатели могут производить сложную обработку событий, вызывая дополнительную загрузку процессора и памяти.

Вот несколько заметок по производительности событий:

- Если событие непостоянное (т.е. хранение базы данных для его типа отключено) и не прослушивается какими-либо системными компонентами (такими как модель 814 или привязки виджета 1295), его возникновение не вызовет заметной утилизации процессора или памяти

- Сохранение события в базе данных вызывает загрузку диска и процессора на сервере базы данных. Максимальный уровень хранения базы данных практически ограничен несколькими тысячами событий в секунду. Это уровень, обеспечиваемый лежащей в основе базой данных.
- События, вызывающие операции сервера (такие как переоценка пригодности 749 или обработка привязок модели 814), дополнительно влияют на производительность, определяемую этими операциями. См. соответствующие статьи о производительности для получения более подробной информации.
- События, доставляемые удаленным слушателям (такие как привязки 1295 виджетов, запущенные в AtomMind Client), могут вызывать значительную утилизацию пропускной способности сети, если происходят слишком часто.

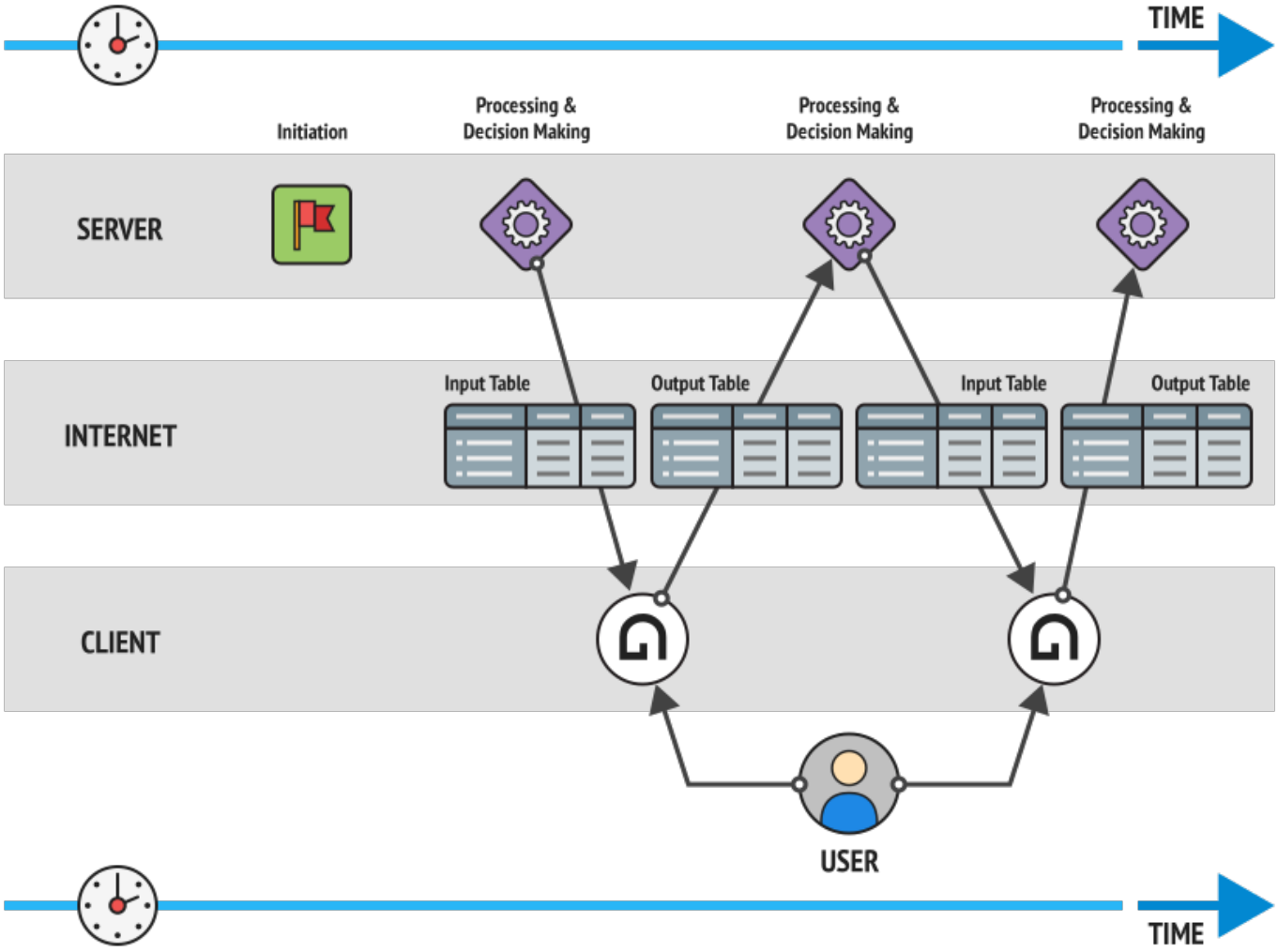
3.6 Действия

Действия представляют собой операции, выполняемые человеком и относящиеся к контексту 41. Список доступных действий может появиться, например, в контекстном меню узла Системного дерева 370.

Существует ключевое различие между действиями контекста и его функциями 70: в отличие от функций, действия могут "общаться" с пользователем, возвращая какие-либо значения или запрашивая входные параметры. Не все действия являются интерактивными, однако, большинство из них включают в себя несколько шагов, требующих внимания пользователя.

Действия общаются с пользователем при помощи UI Процедур 88. Каждое действие представляет собой комбинацию UI процедур и операций с серверными данными, такими как вызовы функций 70, изменение или анализ значений переменных 61 и т.д. Для более подробной информации обратитесь к разделу UI Процедуры 88.

Последовательность операций обработки серверных данных и UI процедур, выполняемых действием, называется *потоком действий*.



Для сравнения приведем примеры Действий:

- Контекст Терминал сбора данных 1494 имеет действие **Управление устройством,** которое позволяет пользователю изменять настройки устройства Терминал 497.

- Контекст Тревоги 1452 имеет действие **Создать новую тревогу**, которое запрашивает базовые настройки новой Тревоги 779, создает ее (вызвав *функцию* "Создать новую тревогу"), а затем снова обращается к пользователю, позволяя ему конфигурировать только что созданную тревогу.

Теперь рассмотрим функции:

- Корневой контекст 1559 AtomMind Server имеет функцию **Остановка сервера** ("stop"), которая останавливает сервер.
- Контекст Пользователи 1604 имеет функцию **Удалить учетную запись пользователя** ("delete"), которая удаляет выбранную учетную запись.

Как показывает пример, *функции* являются операциями, выполняемыми оборудованием, а *действия* требуют вмешательства и ввода пользователя.

Действие по умолчанию

Большинство контекстов имеют так называемое действие *по умолчанию*. Оно выполняется, когда, например, пользователь, совершает двойной щелчок мышью по контексту в Системном дереве 370 AtomMind Client.

Приведем два примера действий по умолчанию:

- Действием по умолчанию контекста Фильтры событий 1507 является "Создать новый фильтр".
- Действием по умолчанию контекста Запрос 1548 является "Выполнить запрос", которое выполняет запрос 829 по нажатию и отображает его выход пользователю.

Отслеживание выполнения действия

Каждый раз при выполнении действия оператором или системным компонентом создается событие действие 1450 и сохраняется в контексте Администрирование. Сохранение истории выполнения действия важно для сохранения отслеживания журнала аудита и обеспечения безопасности системы.

Дистрибутив AtomMind включает фильтр событий 762 **Действия**, позволяющий просматривать историю событий и отслеживать выполнение действий в реальном времени.

3.6.1 UI процедуры

Когда AtomMind Server взаимодействует с пользователем, он может выдать всплывающее сообщение, показать таблицу данных 49 или список с флаговыми кнопками, чтобы пользователь мог выбрать одну или несколько опций. Если вдуматься, достаточно всего нескольких типов интеракций, чтобы полноценно взаимодействовать с пользователем. Нет необходимости создавать специальный пользовательский диалог всякий раз, когда пользователю необходимо взаимодействовать с программой.

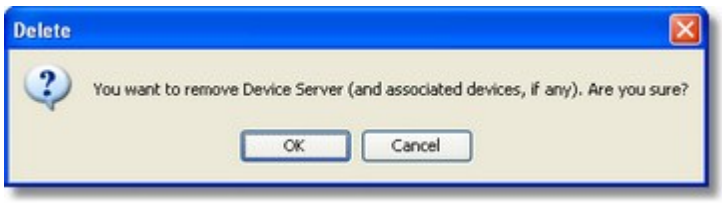
Стандартизированное взаимодействия с пользователем позволяет быстро освоить систему. Процесс обучения не такой трудоемкий, потому что вам достаточно понять данные в диалоговом окна, а не сам диалог.

*Процедуры пользовательского интерфейса (*UI процедуры) являются основными структурными элементами действий 87. Эти UI процедуры используются для отображения тревог, подтверждения действий, редактирования свойств и т.д. Каждое действие - это комбинацию серверных операций и UI процедур. В зависимости от логики действия и среды выполнения, действие может инициировать UI процедуры в различных комбинациях. Например, Действие Вызов функции 103 позволяет пользователю редактировать параметры входа вызываемой функции при помощи UI процедуры Редактировать данные 90, затем вызвать функцию и определить, какую UI процедуру начать: Показать ошибку 95, если выполнение функции произошло с ошибкой, Редактировать данные 90 в режиме "только чтение", если функция вернула какой-либо выход, или просто Показать сообщение 97 о том, что функция была выполнена успешно.

Другое ключевое преимущество использования UI процедур становится очевидным, если рассматривать AtomMind как мультиплатформенную систему. При использовании веб интерфейса 220 или десктопного клиента 359 для взаимодействия с системой AtomMind, очевидно, что средства контроля через существующий пользовательский интерфейс (UI) будут немного отличаться. Однако, средства управления UI всегда берут начало в UI процедурах, т.е. даже на смартфоне, можно просмотреть сообщение об ошибке или разрешить пользователю выполнить выбор множества объектов. Это также значительно сокращается процесс изучения и улучшает его систематичность.

3.6.1.1 Подтвердить

Данная простая UI процедура 88 используется для подтверждения пользователем различных операций. В AtomMind Client она выдает всплывающее диалоговое окно с кнопками "OK" и "Отмена":



параметры

- **Сообщение**. Строка с текстом подтверждения.
- **Тип опций.** Кнопки буду отображены в диалоговом окне. Допустимые целочисленные значения: -1 - По умолчанию, 0 - Да/Нет, 1 - Да/Нет/Отмена, 2 - OK/Отмена.
- **Тип сообщения.** Тип диалога подтверждения. Допустимые целочисленные значения: -1 - Простое сообщение, 0 - Ошибка, 1 - Информация, 2 - Предупреждение, 3 - Вопрос.

Выход

Это действие возвращает значение выбранной опции: 0 - Да или OK, 1 - Нет, 2 - Отмена.

Поддержка группировки

Данная UI процедура поддерживает группировку действий 101. При группировке, пользователь может выбрать опцию "Применить ко всем", которая передаст положительный или отрицательный выбор всем действиям в группе.

В AtomMind Client, флаговая кнопка **Применить ко всем** отображена в окне подтверждения:

3.6.1.2 Открыть в браузере

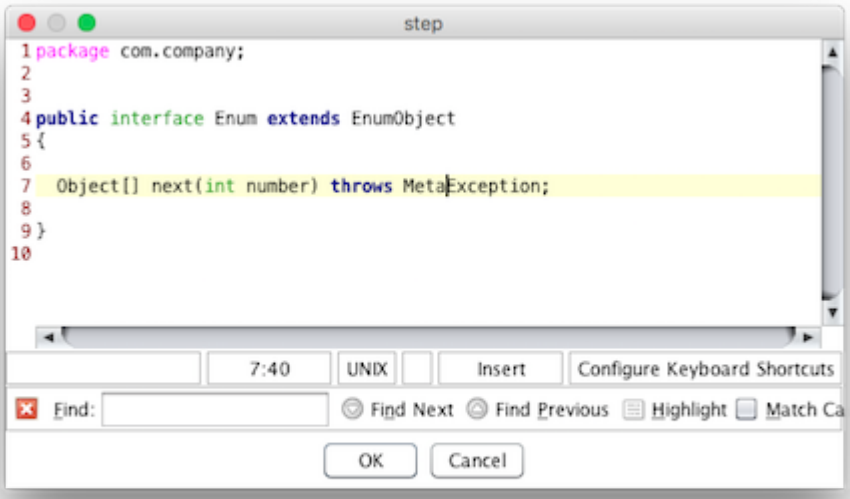
Данная UI процедура 88 открывает URL веб-страницы в браузере.

параметры

- **URI**. Полный URL-адрес веб-страницы (включая часть с указанием протокола, например,'http://...').

3.6.1.3 Редактировать код

Эта UI процедура 88 подсказывает пользователю, как редактировать код редакторе кодов 409 с подсветкой синтаксиса и широким набором опций, анализаторов кода, клавишами быстрого доступа и т.д.



параметры

- **Код**. Текст кода для редактирования.
- **Режим**. Режим подсветки текста в редакторе. Может быть одним из: *aggregate*, *java*, *XML*, *sql*, *shellscript*, *smi-mib*.

выход

Действие возвращает отредактированный код в виде строки.

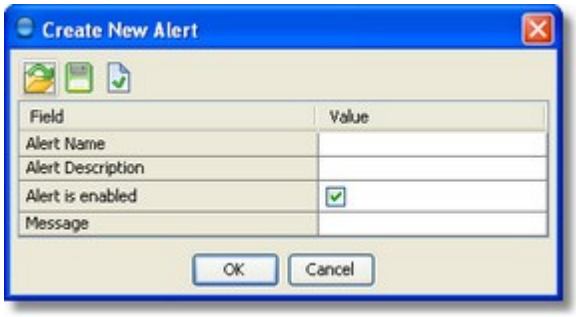
3.6.1.4 Редактировать инструментальную панель

Эта UI процедура 88 открывает веб инструментальную панель 918 для редактирования.

3.6.1.5 Редактировать данные

Данная UI процедура 88 позволяет редактировать или просматривать содержание одной Таблицы данных 49. Она может использоваться, к примеру, для определения входных параметров функции 70 или просмотра возвращаемого значения.

Данные могут отображаться в режиме редактирования или как данные, доступные только для чтения. Прокрутка самого диалогового окна может быть включена или отключена, оно также может отображаться в отдельном окне или как часть главного окна UI. Все эти параметры настраиваются для данной UI процедуры. В AtomMind Client редактирование выполняется при помощи компонента Редактор таблицы данных 382:



параметры

- **Данные.** Статические данные для отображения/редактирования.
- **Выражение получения данных**. Выражение, вычисляемое для получения исходных данных для редактирования/отображения. Если выражение указано, свойство **Данные** будет проигнорировано. Среда вычисления выражения описана здесь 898.
- **Период обновления**. Определяет, с какой частотой будет перевычисляться **Выражение получения данных** и обновляться данные, отображаемые данной UI процедурой.
- **Использовать стыкуемое окно**. Определяет, что будет использоваться - стыкуемое окно или модальный диалог.
- **Только чтение**. Флаг, который включает/выключает редактирование отображенной Таблицы данных 49.
- **Включить контекстное меню.** Определяет, разрешен ли доступ к меню Редактора таблицы данных 382.
- **Контекст по умолчанию**. Контекст по умолчанию для использования Редактором таблицы данных 382.
- **Справка**. Удобочитаемое описание или комментарии к отображаемым данным.
- **Показать панель инструментов**. Определяет видимость панели инструментов Редактора таблицы данных 382.
- **Показать номера строк**. Определяет видимость номеров строк Редактора таблицы данных 382.
- **Горизонтальная прокрутка**. Включает горизонтальную прокрутку данных.

выход

Данное действие возвращает отредактированные данные в виде Таблицы данных 49.

Поддержка группировки

Данная UI Процедура поддерживает группировку действий 101.

В AtomMind Client Редактор таблицы данных в таком случае будет содержать дополнительную кнопку **Все** (кнопка посередине снизу, между **OK** и **Отмена**). Она используется, чтобы применить измененные данные ко всем действиям, выполняемым в группе:

3.6.1.6 Редактировать выражение

Данная UI процедура 88 открывает редактор выражений 404, позволяя тестировать некоторые выражения 112.

3.6.1.7 Редактировать свойства

Данная UI Процедура 88 используется для изменения или просмотра одной и более переменных 61 контекста. Она применяется, к примеру, действием 103 Конфигурировать 105 во время настройки Устройств, Тревог и других объектов. Данная процедура позволяет перезагружать или сохранять свойства во внешних файлах. С ее помощью также может осуществляться экспорт и импорт значений свойств из оригинального формата AtomMind в сторонние форматы.

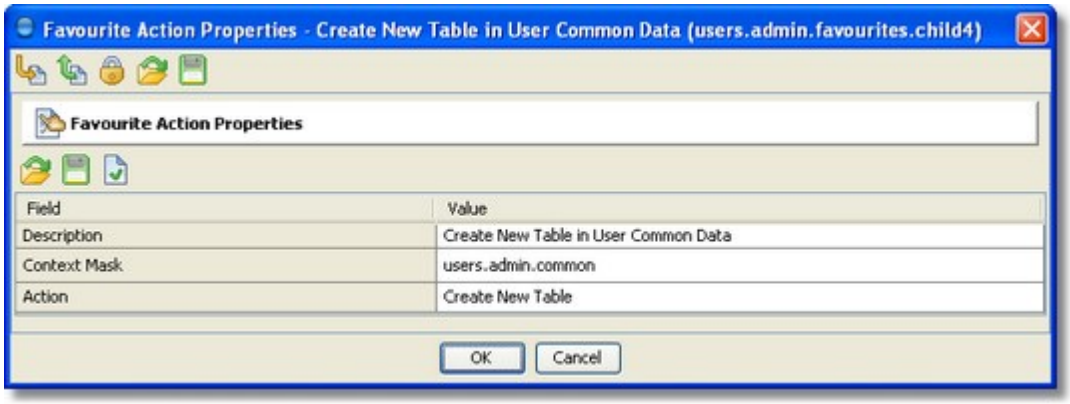
В некоторых случаях свойства доступны только для чтения.

На первый взгляд, эта UI процедура может показаться похожей на UI процедуру Редактировать данные 90. Однако, основное различие заключается в том, что процедура Редактировать данные позволяет изменять любые данные в таблице (например, при введении параметров для функции), функция Редактировать свойства

используется только для изменения свойств (или "переменных") различных контекстов. Изменение свойств включает в себя больше опций, нежели чем изменение общих данных. Например, используя данную процедуру, вы можете применить изменения и сохранить их (т.е. не нужно закрывать диалоговое окно для сохранения изменений). UI процедура Редактировать свойства предлагает больше вариантов, чем Редактировать данные. Они могут казаться похожими в некоторых клиентах, но на самом деле они абсолютно разные.

Чтобы понять различие между ними, обратимся к примеру диалогового окна Печать любого аппаратного средства (которое просит пользователя ввести количество копий, какие страницы печатать и т.п.) и Редактора Свойств в Visual BASIC (который просит пользователя редактировать свойства определенного компонента пользовательского интерфейса (UI)). Первое (диалоговое окно Печать) похоже на UI процедуру Редактировать данные - вы вводите данные, но они не применяются как свойства для объекта. Эти данные используются как аргументы, которые приводят к выполнению операций системой. В противоположность этому, Редактор свойств в VB похож на UI процедуру Редактировать свойства - систематически проходит по всем свойствам объекта, изменяя их.

В AtomMind Client, редактирование выполняется при помощи компонента Редактор свойств 377:



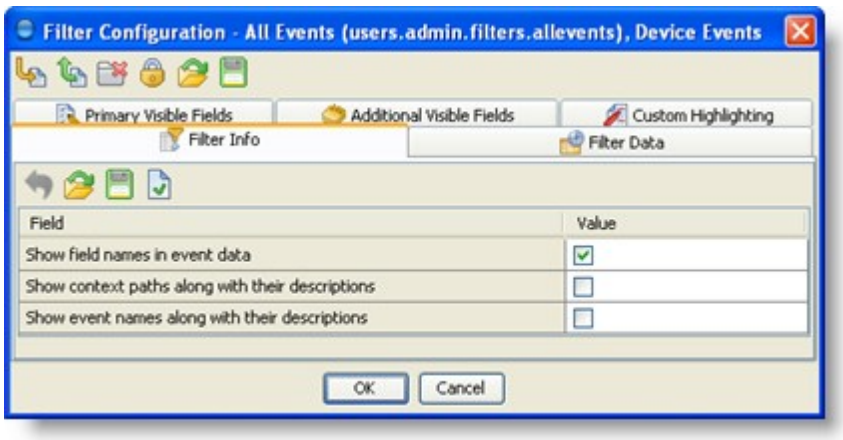
Редактировать свойства

- **Контекст**. Контекст для показа переменных.
- **Группа**. Группа свойств для показа.
- **Свойства**. Список свойств для показа. Включается, если **Группа** пустая.
- **Упрощенный режим**. Когда включено, скрывает инструментальную панель Редактора свойств 377.
- **Использовать стыкуемое окно**. Определяет, что будет использоваться - стыкуемое окно или модальный диалог.
- **Открывать в режиме чтения**. Контролирует включение режима только чтение Редактора свойств 377. Название содержит слово "открывать", т.к. этот режим можно включить вручную кнопкой **Включить/выключить редактирование** ( ) на панели после открытия диалогового окна.

Поддержка группировки

Данная UI процедура поддерживает группировку действий 101.

В AtomMind Client, Редактор свойств будет содержать только свойства и поля, общие для всех контекстов, чьи свойства редактируются. Список этих контекстов отображается в заголовке окна Редактора свойств:

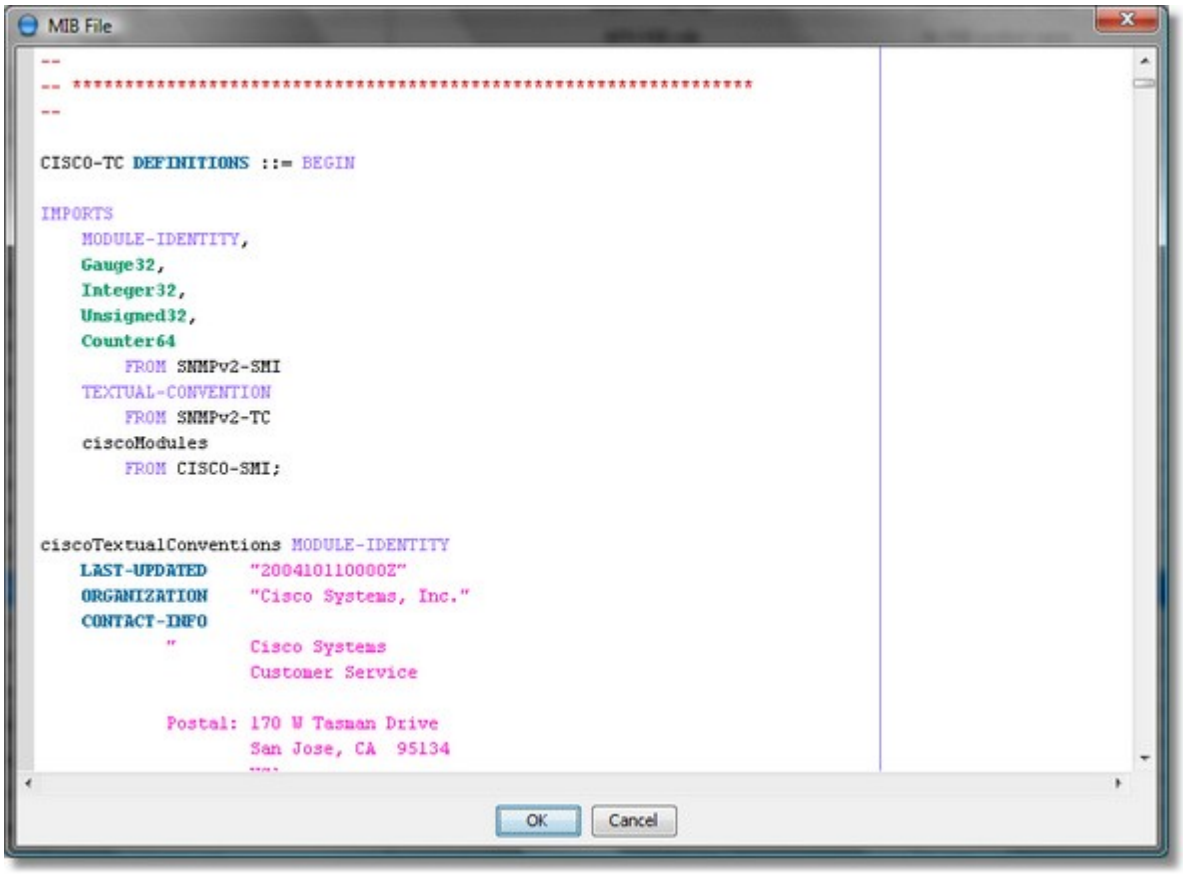


Имейте в виду, что отображаемые в Редакторе свойств данных, по сути, считываются из первого контекста в списке (так называемого *Мастер-контекста*). Однако, все изменения, внесенные в Редакторе свойств, будут применяться к Мастер-контексту и всем другим редактируемым контекстам (*подчиненным контекстам*).

3.6.1.8 Редактировать текст

Данная UI процедура 88 помогает пользователю редактировать какой-либо текст в текстовом редакторе с подсветкой синтаксиса.

В AtomMind Client данная UI процедура отображает компонент Текстовый редактор 408:



параметры

- **Текст**. Текст для редактирования.
- **Режим**. Режим подсветки синтаксиса в редакторе. Может быть одним из: *aggregate*, *java*, *XML*, *sql*, *shellscript*, *smi-mib*.

выход

Действие возвращает отредактированный текст в виде строки.

3.6.1.9 Запустить виджет

Данная UI процедура 88 открывает виджет 943 в окне AtomMind Client.

ПАРАМЕТРЫ

- **Виджет**. Путь контекста запускаемого Виджета 943.
- **Контекст по умолчанию**. Контекст по умолчанию 946 виджета.
- **Шаблон**. Шаблон виджета.

3.6.1.10 Выбрать объекты

Данная UI процедура 88 позволяет пользователю выбрать контексты 41, переменные 61, функции 70, события 73 и их поля при помощи удобного древовидного графического компонента. Например, она применяется действиями перетаскивания (Drag and Drop) 101, которые запускаются без самого перетаскивания - она позволяет пользователю выбрать контекст, который будет использоваться в качестве "перетаскиваемого".

В AtomMind Client выбор осуществляется через компонент Селектор объектов 402:

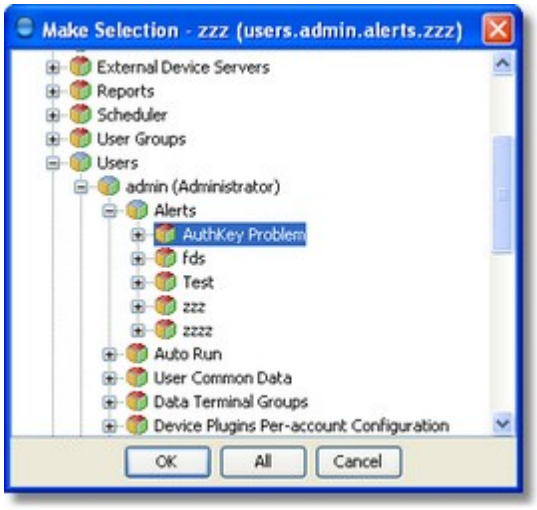
параметры

- **Типы**. Список типов контекстов 43, доступных для выбора.
- **Корневой элемент**. путь к контексту, который будет корневым для отображаемого дерева.
- **По умолчанию**. Контекст дерева по умолчанию. Выбранные объекты ниже этого узла будут иметь относительный путь 43.
- **Развернутый**. Контекст, который будет развернут после открытия Селектора объектов 402.
- **Показывать дочерние контексты**. Флаг определяет видимость дочерних контекстов.
- **Разрешить маски**. Разрешить видимость и возможность выбора масок. При таком режиме результат выбора может быть и путем, и маской.
- **Показывать переменные**. Опция включает/выключает видимость переменных контекста 61.
- **Показывать функции**. Опция включает/выключает видимость функций контекста 70.
- **Показывать события**. Опция включает/выключает видимость событий контекста 73.
- **Показывать поля**. Опция включает/выключает видимость полей объектов 51.
- **Выбор одного элемента**. Переключает режимы выбора одного/множества элементов.
- **Использовать чекбоксы**. Управляет использованием чекбоксов для выбора.

Поддержка группировки

Данная UI процедура поддерживает группировку действий 101. При группировании действий возможно применять один и тот же выбор объекта ко всем действиям в группе.

В AtomMind Client диалоговое окно Селектор Объектов содержит дополнительную кнопку **Все**:

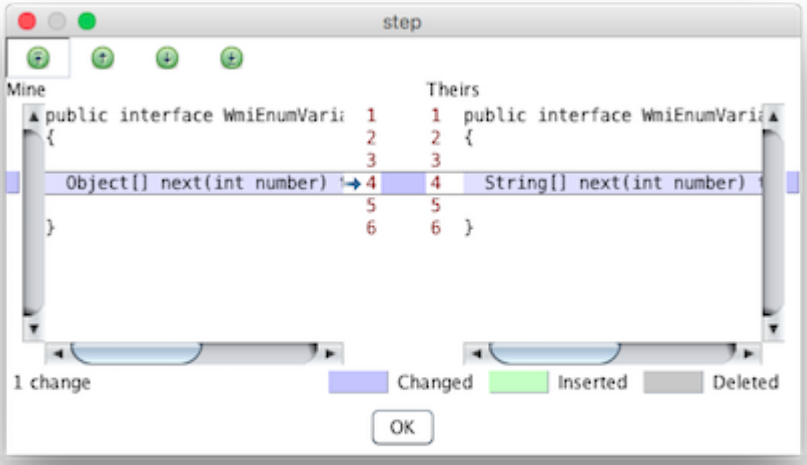


3.6.1.11 Показать инструментальную панель

Данная UI процедура 88 открывает инструментальную панель 912.

3.6.1.12 Показать отличия

Данная UI процедура 88 открывает Окно просмотра отличий, что позволяет визуально сравнить два текстовых источника.



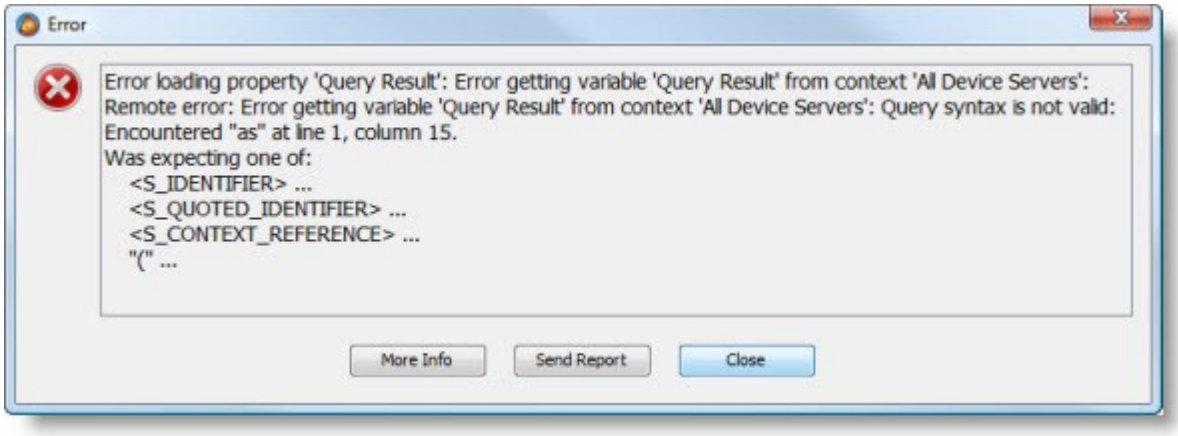
параметры

- **Заголовок первого файла**. Заголовок левой части редактора.
- **Первый файл**. Текстовое содержимое левой части редактора.
- **Заголовок второго файла**. Заголовок правой части редактора.
- **Второй файл**. Текстовое содержимое правой части редактора.

3.6.1.13 Показать ошибку

Данная UI процедура 88 активируется, когда во время выполнения действия 103 возникают условия ошибки. Процедура выдает сообщение и информацию об ошибке и позволяет пользователю отправить отчет.

В AtomMind Client данная UI процедура отображает Диалоговое окно сообщения об ошибке 407:



параметры

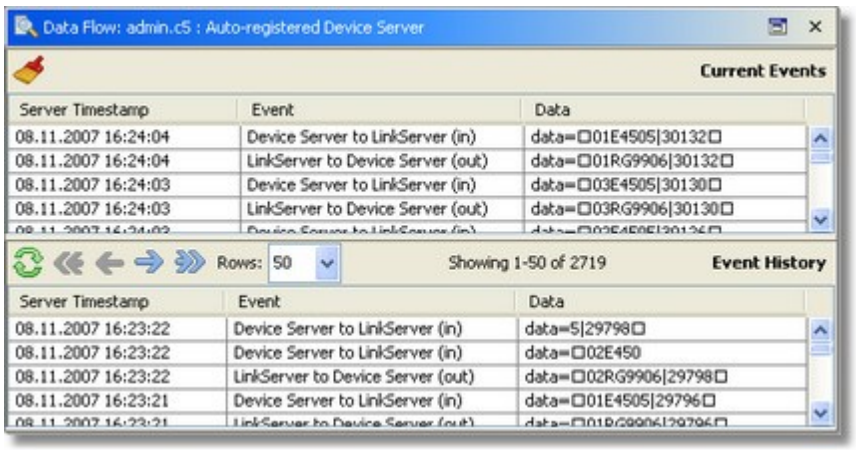
- **Уровень**. Уровень 75 ошибки.
- **Сообщение**. Сообщение об ошибке.
- **Исключение**. Текст исключения, приведшего к ошибке, либо просто подробное описание ошибки.

3.6.1.14 Показать журнал событий

Данная UI процедура 88 используется для просмотра событий 73 или доступа к их истории. Она может применяться для отображения потока событий в реальном времени, по мере их появления, или для просмотра событий, которые появились ранее (т.е. истории 73).

Данная операция также определяет, какие столбцы будут отображены в списке событий. В некоторых случаях, показываются только временные метки и имена событий. В других случаях, видимы другие свойства события, такие как контекст 41, в котором оно произошло, подтверждения 73 оператором, уровень 73 (критичность) события и т.д.

В AtomMind Client, эта UI процедура запускает компонент Журнал событий 398:



параметры

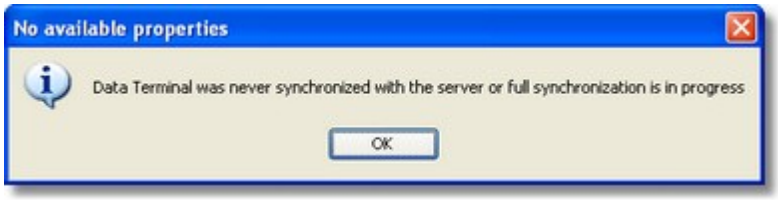
- **Фильтр событий**. Будут отображаться события, отвечающие указанному фильтру.
- **События**. Если не указан Фильтр событий, для фильтрации событий будет использоваться данный список пар Контекст-Событие. Будут отображаться только определенные события контекстов.
- **Событие по умолчанию**. Событие такого типа можно создать при помощи кнопки Создать событие (⊕), которая появится на инструментальной панели Журнала событий.
- **Текущие события**. Определяет, отображать ли секцию текущие события 399.
- **История событий**. Определяет, отображать ли секцию история событий 400.
- **Автоматически загружать историю событий**. Определяет, будет ли автоматически загружаться первая страница истории событий 400, или она будет пустой.

- **Показывать имена контекстов**. Если параметр включен, то контекст, в котором происходит событие 73, будет доступен как поле в обеих секциях.
- **Показывать имена событий**. Если параметр включен, имя события 73 будет доступно как поле в обеих секциях.
- **Показывать уровни событий**. Если параметр включен, уровень события 75 будет доступен как поле в обеих секциях.
- **Показывать данные событий**. Если параметр включен, данные события 73 будут доступны как поле в обеих секциях.
- **Показывать подтверждения событий**. Если параметр включен, подтверждения события 76 будут доступны как поле в обеих секциях.
- **Показывать обогащения события**. Если параметр включен, обогащения событий 76 будут доступны как поле в обеих секциях.
- **Параметры**. Таблица данных 49 с параметрами фильтра событий 772.

3.6.1.15 Показать сообщение

Данная простая UI процедура 88 показывает сообщение пользователю и ждет ответа.

В AtomMind Client эта UI процедура отображает всплывающее диалоговое окно с одной кнопкой "Ок":



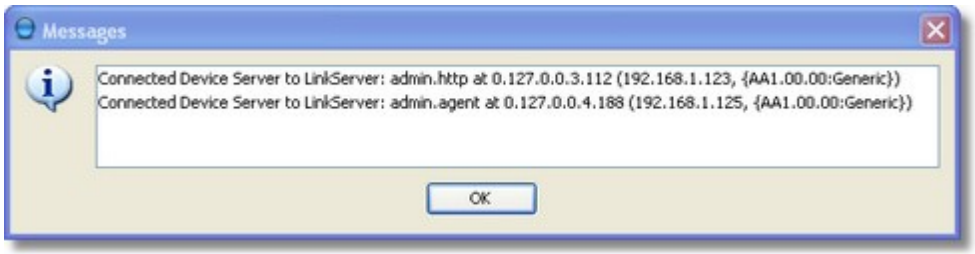
параметры

- **Сообщение**. Строка сообщения.
- **Уровень.** Определяет критичность обмена сообщениями. Допустимые целочисленные значения: 0 - Простое сообщение; 1,2 - Информация; 3 - Предупреждение; 4,5 - Ошибка.

Поддержка группировки

Данная UI процедура поддерживает группировки действий 101. При группировки действий сообщения, передаваемые в результате выполнения UI процедуры различными действиями в группе, отображаются все вместе.

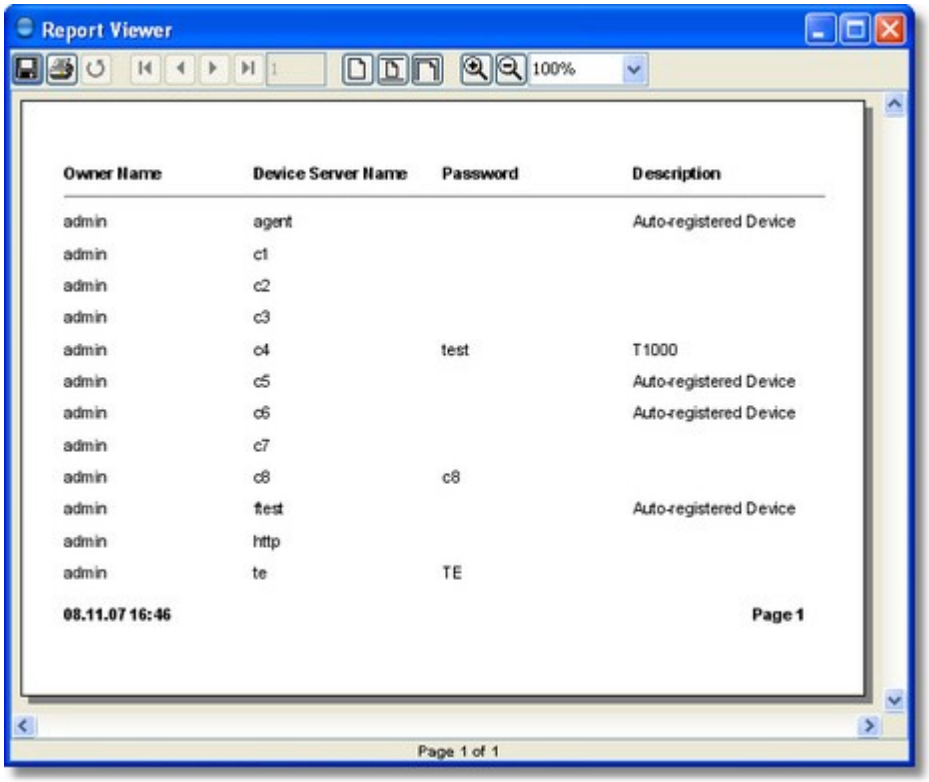
В AtomMind Client для такого случая используется одно всплывающее диалоговое окно:



3.6.1.16 Показать отчет

Данная UI процедура 88 позволяет пользователям просматривать, печатать или экспортировать отчеты 928 в различные форматы.

В AtomMind Client данная UI процедура показывает отчет в компоненте Средство просмотра отчетов 415:



параметры

- **Данные для отчета**. Двоичные данные для отчета.
- **Формат отчета**. Формат, используемый при экспортировании отчета.
- **Имя файла**. Имя файла, используемое для экспортирования отчета.

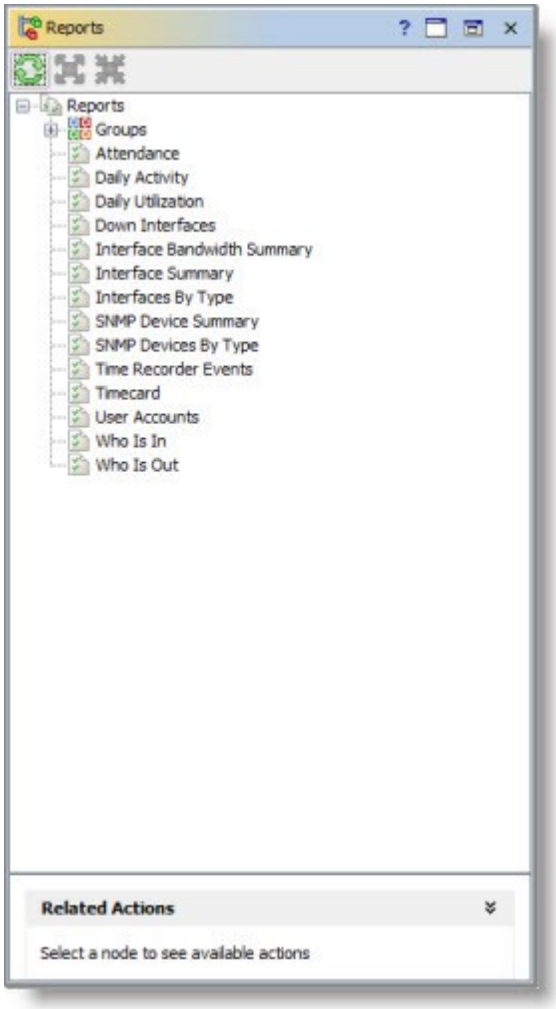
3.6.1.17 Показать системное дерево

Данная UI процедура используется для отображения дерева контекстов 41 сервера и позволяет пользователю управлять контекстами в этом субдереве. Процедура отображает компонент Системное дерево в отдельном окне.

Более подробно об этом см.:

- Системное дерево 370 в AtomMind Client
- Системное дерево 275 в Web UI

Системное дерево в AtomMind Client выглядит следующим образом:



параметры

- **Корневой элемент**. Контекст, который будет корневым. Все его родительские контексты будут недоступны.
- **Корневые элементы**. Таблица с множеством корневых элементов. Игнорируется, если определен один **Корневой элемент**.
- **Действия**. Показывает/скрывает окно соответствующие действия 372.
- **Контекстное меню**. Включает/отключает всплывающее меню.
- **Инструментальная панель**. Включает/отключает инструментальную панель.

3.6.1.18 Запустить интерактивное руководство

Данная UI процедура 88 запускает интерактивное руководство, которое учит пользователей, как пользоваться различными компонентами AtomMind.

Для более подробной информации обратитесь к главе Интерактивный самоучитель 476.

3.6.2 Режимы выполнения действий

Действия выполняются в двух режимах:

- Интерактивный 100 режим
- Автономный 100 (неинтерактивный) режим

3.6.2.1 Интерактивные действия

Интерактивный режим является обычным режимом выполнения, поддерживаемым каждым действием. В данном режиме, каждая GUI процедура, выполняемая во время потока действий, проводит какую-либо интеракцию пользователя при помощи пользовательского интерфейса AtomMind Server. Например, она может отображать сообщения или попросить пользователя изменить некоторые данные.

3.6.2.2 Неинтерактивные действия

Неинтерактивный режим поддерживается только действиями, которые могут быть выполнены "офлайн", без взаимодействия с пользователем. Выполнение действий в неинтерактивном режиме поддерживается Тревогами 793 и Запланированными задачами 823.

В таком режиме, результат каждой GUI процедуры определен заранее, когда действие конфигурируется для неинтерактивного выполнения. Эти заранее определенные результаты настраиваются во время установки выполнения автономных действий, т.е. при конфигурировании тревоги или задачи планировщика.

Все выходы действий перенаправляются в журнал 166 сервера и/или конвертируются в события контекста 73.

Привязки параметров ввода

При редактировании заранее определенных параметров неинтерактивного действия в Редакторе Таблиц Данных возможно настроить 397 привязки 741 таблицы. Когда сервер выполняет неинтерактивное действие, он оценивает все привязки, содержащиеся в его заранее определяемых параметрах, заполняя их актуальной или контекстно зависимой информацией.

Пример: Возможно запланировать действие Экспорт отчета в файл сервера 1554 для отчета **Недавние тревоги** и использовать привязки для генерировния другого имени файла при каждом экспортировании отчета. В этом случае к **Вводу** будет добавляться следующая привязка: таблица **Экспорт в файл сервера**:

- Цель: `File Path (file), String`
- Выражение: `"alerts_report_" + year(now()) + "_" + month(now()) + "_" + day(now()) + ".pdf"`

Пример: Можно выполнить внешнее приложение при возникновении тревоги путем добавления действия Выполнить приложение 1551 к списку корректирующих действий 793 тревоги. Однако часто бывает необходимо передать сообщение тревоги и другие параметры тревоги этому приложению. В этом случае к **Вводу** нужно добавить следующие или подобные привязки: таблица **Выполнить приложение или команду операционной системы**:

- Цель: `Command (command), String`
- Выражение: `"\"" + cell({env/parameters}, "message") + "\""`

Это выражение ссылается на `parameters` переменной среды 123, чьи значения Таблица данных ассоциирует с событием тревоги 790.

Пример: Для отправки e-mail сообщения по тревоге с текстом, описывающим причину тревоги, добавьте к тревоге автоматическое корректирующее действие **Отправить E-mail**, откройте его параметры ввода, зайдите внутрь **Ввода**: таблица **Отправить E-Mail**:

- Цель: `Message (message), String`
- Выражение: `"An alert was raised, it is caused by value of custom field: " + cell(cell({env/parameters},'data'),'customField')`

Это выражение ссылается на `parameters` переменной среды 123, чьи значения Таблица данных ассоциирует с событием тревоги 790. Затем оно использует первую (внутреннюю) функцию `cell()` для извлечения таблицы данных, относящейся к значению переменной (или данным события), вызвавшей тревогу. Другая (внешняя) функция `cell()` извлекает ползовательское поле из этой таблицы данных. Значение поля добавляется к тексту e-mail сообщения.

ОЦЕНКА ПРИВЯЗОК ПАРАМЕТРОВ ВВОДА

Среда выполнения привязок параметров действия позволяет получить доступ к:

- Данным из любого контекста[41] сервера
- Начальным параметрам выполняемого действия

Среда вычисления[114] привязок параметров ввода:

Контекст по умолчанию[119]	Контекст, откуда выполняется действие.
Таблица данных по умолчанию[120]	Текущая таблица, т.е. таблица, по которой оцениваются привязки.
Строка по умолчанию[119]	Текущая строка, если привязка не ссылается на отдельную строку и оценивается для каждого ряда таблицы отдельно. В ином случае равно нулю.

Переменные среды[123]

Имя переменной	Тип значения	Описание
parameters	Таблица Данных	Начальные параметры действия: • Таблица данных события тревоги[790], если действие выполняется при появлении тревоги • Не определены, если действие выполняется планировщиком
context	Строка	Доступно только при выполнении действия по тревоге. Представляет путь контекста тревоги[1454].
event	Строка	Доступно только при выполнении действия по тревоге. Представляет имя события тревоги (т.е. `alert`).
level	Целое	Доступно только при выполнении действия по тревоге. Представляет уровень[75] события тревоги.
time	Дата	Доступно только при выполнении действия по тревоге. Представляет временную метку события тревоги.
acknowledgements	Таблица данных	Доступно только при выполнении действия по тревоге. Представляет таблицу подтверждений[76] события тревоги.
enrichments	Таблица данных	Доступно только при выполнении действия по тревоге. Представляет таблицу обогащений[76] события тревоги.

3.6.3 Группировка действий

AtomMind поддерживает группировку действий[87], которое представляет собой объединенное выполнение одного и того же действия из нескольких контекстов[41].

Некоторые типы действий кардинально меняют свое поведение при запуске в режиме группы. Например, когда действие Конфигурировать[105] группируется для нескольких контекстов, свойства читаются из первого контекста в группе, однако, измененные настройки записываются во все контексты, позволяя тем самым конфигурировать множество объектов одновременно.

Другие действия поддерживают группирование на уровне GUI процедур[88]. Например, когда инициируется действие Вызов функции[103] для группы контекстов, оно позволяет пользователю применять одно подтверждение для всех действий в группе (т.е. подтверждать нужно только один раз). Параметры входа вызываемой функции можно задать только один раз для всех действий. Таким образом, вместо отдельного всплывающего окна для каждого действия появляется одно, в котором сдержится информация о выполнении сразу всех действий.

3.6.4 Действия перетаскивания

Некоторые действия активируется через их перетаскивание мышью в Пользовательском интерфейсе AtomMind Server. Например, в AtomMind Client вы можете перетащить один узел Системного дерева[370] в другой, что приведет к запуску действия. Контекст, который был перенесен в другой узел, называется "принятым контекстом".

Действия перетаскивания могут быть также активированы без участия мыши. Например, вы можете запустить их из списка Относительных действий[372], появляющегося под Системным деревом в AtomMind Client, или из контекстного меню узла в Системном дереве. В данном случае, пользователю будет предложено выбрать "принимаемый" контекст при помощи GUI процедуры Выбор объектов[94].

Каждое действие перетаскивания может принимать контексты одно и более типов 43. Некоторые действия могут принимать любой контекст, другие же принимают контексты только определенного типа.

3.6.5 Действия, относящиеся к переменным

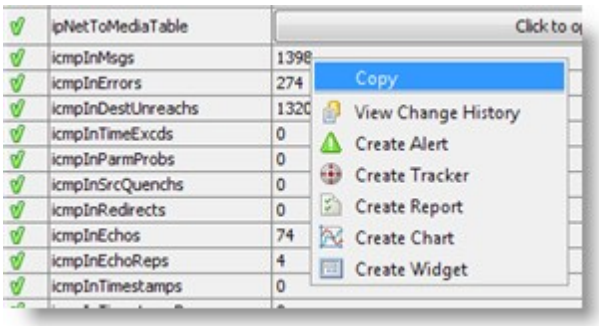
Некоторые действия активируются выбором переменной контекста 61 в Пользовательском Интерфейсе AtomMind Server и запуском в последствии действия *для* данной переменной. Такие действия получают имя выбранной переменной и путь к контексту 41, в котором она определена при запуске. Их рабочий поток тесно связан с данной переменной, поэтому они называются *Действиями, относящимися к переменной.*

Например, мы можем выбрать переменную, указывающую текущую температуру, определенную в Контексте Устройство 1494 измерительного прибора и запустить действие **Создать таблицу** 1624 для данной переменной. Данное действие создаст виджет 943, содержащий компонент график 1051 для отображения истории изменения температуры.

Действия, относящиеся к переменным похожи на Действия, относящиеся к событиям 102, которые на входе ссылаются на событие 73 контекста.

В AtomMind Client 359 Действия, относящиеся к переменным, запускаются правым нажатием клавиши по переменной в Редакторе свойств 377 и выбором действия для запуска из контекстного меню.

Доступ к таким действиям для переменной настройки аппаратного устройства осуществляется следующим образом (скриншот сделан в AtomMind Client):



3.6.6 Действия, относящиеся к событиям

Некоторые действия активируются выбором события контекста 73 в Пользовательском Интерфейсе AtomMind Server и запуском в последствии действия *для* данного события. Такие действия получают имя выбранного события и путь к контексту 41, в котором оно определено при запуске. Их рабочий поток тесно связан с данным событием, поэтому они называются *Действиями, относящимися к событию.*

К примеру, мы можем выбрать событие, указывающее, что определенное устройство 1494 было отключено от сервера и запустить действие **Создать тревогу** для данного события. Действие создаст тревогу 779, которая активируется при следующем отключении данного устройства.

Действия, относящиеся к событиям похожи на Действия, относящиеся к переменной 102, которые на входе ссылаются на переменную 61 контекста.

В AtomMind Client 359 действия, относящиеся к событиям запускаются правым нажатием клавиши по событию в Журнале событий 398 и выбором действия из меню Относительные действия.

3.6.7 Параметры выполнения

Многие действия имеют параметры выполнения, позволяющие предварительно настроить поток действия. Например, действие Вызвать функцию 103 включает в себя параметр **Положение окна отчета**, определяющий где будет отображено окно с результатами выполнения.

Когда пользователь инициирует "обычное" выполнение (например, используя контекстное меню Системного дерева 370 в AtomMind Client), действие запускается с параметрами выполнения по умолчанию.

Особые параметры могут быть заданы в следующих случаях:

- При добавлении действия в Автозапуск 941
- при создании Избранного 2186 действия

Динамические параметры выполнения

Если действие выполняется в неинтерактивном режиме 100, параметры выполнения действия могут динамически выстраиваться "на лету". Для получения более подробной информации см. Привязки параметров ввода 100.

3.6.8 Общие действия

Действия 87 используются для управления контекстами 41 различными способами. Например, удаление 106 контекста является действием, также как и его конфигурация 105.

Некоторые контексты включают в себя действия, которые могут применяться только к ним. К примеру, действие **Инициировать Автозапуск** не сработает в контексте Устройство. Его применение имеет смысл для контекста Автозапуск 1468. Таким образом, оно является **уникальным действием**.

Однако, другие действия могут применяться к различным контекстам. Действие конфигурировать 105, например, позволит вам настроить устройство, запись Автозапуска или тревогу без особых различий. Оно применяется ко всем контекстам схожим образом. Из этого следует, что такое действие является **общим действием**.

Далее в разделе приводится описание всех общих действий, поддерживаемых AtomMind Server.

3.6.8.1 Вызов функции

Данное действие является основным типом действий, используемых в AtomMind Server. Основное предназначение этого действия - позволить пользователю выполнить операцию с аппаратным устройством или объектом системы путем вызова отдельной функции 70 из соответствующего контекста 41.

Неинтерактивный режим 99:	Поддерживается
Права доступа:	Доступно на уровне 486 прав доступа *Наблюдатель*
Параметры 102 выполнения:	Свойства инструментальной панели 926 окна вывода функции Положение окна выхода функции

Порядок действий

1. *[Дополнительно]* Пользователю предлагается подтвердить выполнение при помощи GUI процедуры Подтвердить 89
2. *[Дополнительно]* Пользователю предлагается редактировать входные параметры 70 вызываемой функции при помощи GUI процедуры Редактировать данные 90. Если пользователь отменяет данную GUI процедуру (нажатием кнопки Отмена в диалоговом окне Редактора таблицы данных 382)
3. Происходит вызов функции при помощи действия Вызвать, и сервер выполняет все необходимые операции (создание новой Тревоги, чтение списка всех устройств и его возвращение в качестве выхода функции). Если параметры входа функции не были определены на шаге 2, функция вызывается с использованием параметров по умолчанию.
4. Если функция возвращает ошибку, пользователь может ее увидеть при помощи GUI процедуры Показать ошибку 95. Если параметры входа были определены на шаге 2, поток выполнения действия возвращается на шаг 2, чтобы пользователь мог задать различные параметры.
5. *[Дополнительно]* Если функция возвращает какие-либо данные, их просмотр осуществляется при помощи GUI процедуры Редактировать 90 в режиме "Только чтение"
6. *[Дополнительно]* В некоторых случаях, пользователь видит "Выполнено успешно" или другое сообщение, при помощи GUI процедуры Показать сообщение 97 для указания, что вызов данной функции выполнен успешно.

Поддержка группировки

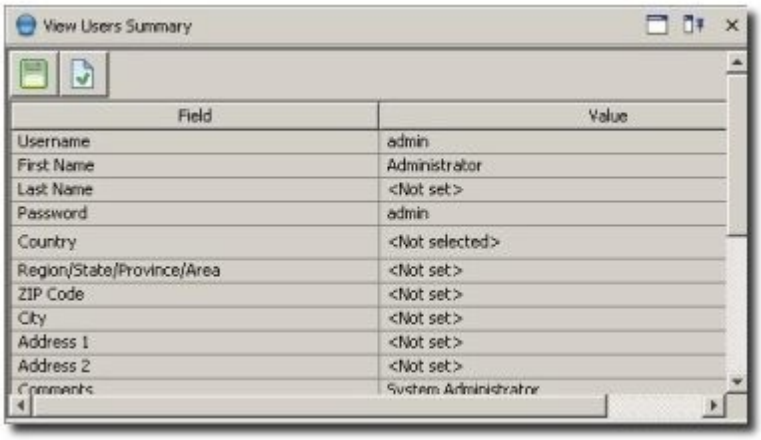
Данное действие поддерживает группировку 101.

- Если требуется подтверждение, оно может применяться ко всем действиям группы.
- Одни и те же параметры входа могут применяться ко всем действиям группы.
- Все сообщения "Выполнено успешно" для действий в группе отображаются в отдельном окне.

Примеры:

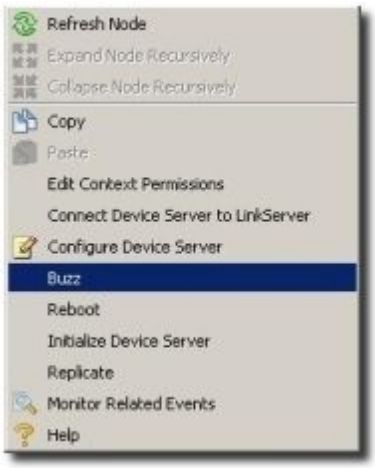
Подключить устройство к AtomMind Server в контексте Внешнее устройство 2112 (запрашивает входные параметры, имеет сообщение "Выполнено успешно")

Посмотреть сводку пользователя в [контексте Пользователи](1604) (не предполагает входные параметры, показывает возвращаемые функцией параметры)



Удалить запрос в [контексте Запрос](1548) (не предполагает передачи и получения параметров, требует подтверждения)

Показать статус в [контексте устройства](2102) (не требует вмешательства пользователя, просто выполняет запрашиваемую операцию)



3.6.8.2 Конфигурировать

Данное действие позволяет пользователю изменить настройки различных аппаратных средств (таких как Устройства 497) или свойства системных объектов (таких как Тревоги 779).

Данное действие также может работать в режиме только чтения, разрешая пользователю только просматривать значения свойств.

Имя действия:	configure (может отличаться в некоторых случаях)
Иконка действия:	 (может отличаться в некоторых случаях)
Клавиша быстрого вызова:	F2
Не интерактивный режим 99:	Не поддерживается
Права доступа:	Доступно на уровне 486 прав доступа *Оператор*
Параметры 102 выполнения:	Положение окна Редактора свойств Свойства инструментальной панели 926 Редактора свойств

Порядок действий

1. Проверьте, имеет ли контекст переменные, которые могут быть сконфигурированы. Если нет, пользователь увидит всплывающее окно сообщения об ошибке. Оно также может означать, что контекст временно недоступен для конфигурации, например, при попытке настроить устройство, которое на данный момент отключено.

2. Измените значения переменных контекста при помощи GUI процедуры Редактировать свойства 91.

Поддержка группировки

Данное действие поддерживает группировку действий 101.

Если действие вызывается для группы контекстов, оно использует UI процедуру Редактировать свойства для открытия настроек первого контекста в группе. Однако, при сохранении, измененные настройки (свойства) записываются во все контексты в группе.

3.6.8.3 Создать

Данное действие применяется для создания Тревог 779, Фильтров событий 762 и т.д.

Имя действия:	create
Иконка действия:	
Не интерактивный режим 99:	Поддерживается
Права доступа:	Доступно на уровне 486 прав доступа *Наблюдатель*

Порядок действий

1. Появляется всплывающее диалоговое окно (или другая форма записи данных) при помощи UI процедуры Редактировать данные 90, пользователю предлагается ввести основные настройки для создаваемого контекста.

2. Создает новый контекст 41 на сервере.

3. Использует UI процедуру Редактировать данные 90, чтобы пользователь смог сконфигурировать только что созданный контекст. Теперь для пользователя доступны и другие свойства.

3.6.8.4 Создать на основе шаблона

Данное действие используется для создания какого-либо системного объекта, например, Виджета или Учетной записи устройства, создавая копию другого объекта того же типа. Оно является действием перетаскивания 101, позволяя пользователю активировать его операцией Перетащить или выбором сначала объекта-"шаблона". Затем создается копия контекста объекта в контексте-контейнере, из которого было активировано действие Создать из отчета. Таким способом, вы можете взять тревогу из одной учетной записи пользователя и создать ее копию в другой.

Имя действия: createFromTemplate

Иконка действия: 

Не интерактивный режим[99]**:** Не поддерживается

Права доступа: Доступно на уровне[486] прав доступа *Менеджер*

Порядок действий

Как только объект-шаблон выбран, порядок данного действия такой же, как и порядок действия Создать копию[109].

3.6.8.5 Удалить

Данное общее действие используется для безвозвратного удаления различных контекстов, таких как Фильтры событий, Виджеты, Избранное и т.д. Удаление контекста останавливает его работу (например, предотвращает выполнение в будущем всех Запланированных задач или поднятие Тревоги), удаляет все связанные с ним настройки (включая долговременные свойства, сохраняемые в базе данных[692]) и удаляет контекст из Списка потомков контекста-родителя.

Действие Удалить основано на Действии Вызвать функцию[103]. Оно не запрашивает параметры и не отображает выходные параметры, однако требует подтверждения.

Имя действия: delete

Иконка действия: 

Клавиша быстрого доступа: Delete

Не интерактивный режим[99]**:** Поддерживается

Права доступа: Доступно на уровне[486] прав доступа *Менеджер*

3.6.8.6 Редактировать права доступа к контексту

Редактировать права доступа является действием Конфигурировать[105], которое используется для определения какие пользователи[478] AtomMind Server имеют доступ к данному контексту. Таблица прав доступа к контексту состоит из двух полей:

Пользователь Имя и описание учетной записи пользователя

Права доступа Уровень[486] прав доступа пользователя для данного контекста

Таблица не сохраняется на сервере постоянно. Она формируется, когда выполняется действие Редактировать права доступа.

Для создания данной таблицы, AtomMind Server читает таблицы прав доступа[487] всех пользователей в системе. Если уровень прав доступа определенного пользователя для текущего контекста выше, чем **Не определен**, в таблицу прав доступа к контексту добавляется новая запись с указанными выше полями (**пользователь** и **права доступа**)

Текущий пользователь (чьё имя и пароль вы используете для доступа к действию) и не отображается в таблице прав доступа к контексту, потому что пользователь не может изменять свои собственные права доступа.

Вы можете добавлять новые записи в таблицу прав доступа при ее редактировании, разрешая тем самым новым пользователям доступ к контексту.

Когда таблица прав доступа к контексту сохранена, она не сохраняется как есть на сервере. Наоборот, сервер меняет таблицы прав доступа отдельных пользователей согласно их правам доступа, определенным в сохраняемой таблице.

Имя действия: perms

Иконка действия: 

Не интерактивный Режим: Не поддерживается

Права доступа: Доступно на уровне прав доступа *Администратор*

ПРИМЕР

Предположим, что изначально таблица прав доступа пользователя **john** выглядит следующим образом:

Context Mask	Permissions
users.john	Manager
users.*	None
*	Manager

Затем мы выполняем Действие **Редактировать права доступа** из **users.admin.alerts.water_level_warning** (данный контекст представляет тревогу "Предупреждение об уровне воды", зарегистрированную под учетной записью **admin**). Мы увидим таблицу прав доступа, похожую на эту:

User	Permissions
newadmin	Administrator
test	Manager

Данная таблица отображает два пользователя с доступом к этой тревоге: пользователь **newadmin** имеет права доступа администратора, а пользователь **test** имеет обычный уровень прав доступа. Пользователь **admin** также имеет некоторые права доступа, однако, мы не видим их здесь, потому что именно эту учетную запись мы используем в данный момент.

Далее, добавим новую запись в таблицу прав доступа. Выберите **john** в поле Пользователь и **Менеджер** в поле Права доступа. Таблица будет выглядеть следующим образом:

User	Permissions
john	Manager
newadmin	Administrator
test	Manager

Затем, нужно сохранить изменения и снова открыть таблицу прав доступа пользователя **john.** Мы видим, что таблица была изменена так, что теперь он имеет права доступа к тревоге "Предупреждение об уровне воды":

Context Mask	Permissions
users.admin.alerts.water_level_warning	Manager
users.john	Manager
users.*	None
*	Manager

Теперь имеет доступ к тревоге "Предупреждение об уровне воды". Он может просматривать, конфигурировать и использовать ее, например, при помощи Системного дерева в AtomMind Client:

Для более подробной информации см. Безопасность и Права доступа.

3.6.8.7 Экспортировать

Действие Экспортировать используется для сохранения выбранных свойств устройств или ресурсов в файл. Оно помогает выбрать отдельные поля данных из контекстов-потомков контекста, в котором оно определено.

Пример: Если вы запускаете действие Экспортировать из контекста **Владельцы карт**, расположенном под определенным контекстом **Отдел**, вы сможете экспортировать имена, адреса и другую информацию о владельцах карт, относящихся к данному отделу. Данный пример взят из решения AtomMind Time and Attendance.

Порядок действий

1. Выбор 90 переменной/поля для экспорта.
2. Система формирует запрос 829, возвращающий выбранные поля в виде отдельной таблицы.
3. Пользователю предлагается просмотреть и настроить сформированный запрос при помощи GUI процедуры Редактировать текст 93.
4. На данном этапе выполняется запрос выборки.
5. Данные, которые вернул запрос, представляются пользователю при помощи GUI процедуры Редактировать данные 90 (в режиме "Только чтение").
6. Пользователь может выбирать, фильтровать и экспортировать данные при помощи средств управления и кнопок инструментальных панелей **Редактора таблицы данных**. Также возможно на основе выбранных данных, сформировать отчет для печати.

Имя действия:	export
Иконка действия:	
Не интерактивный режим 99**:**	Не поддерживается
Права доступа:	Доступен на уровне 486 прав доступа *Менеджер*

3.6.8.8 Импортировать

Действие Импортировать используется для создания нескольких ресурсов и применения параметров, читаемых из файла.

Порядок действий

1. Выбор 90 файла для импорта.
2. *[Дополнительно]* Определение 90 опций импорта, если они доступны для формата выбранного файла.
3. Чтение данных из файла и их конвертирование в Таблицу данных.
4. Просмотр 90 данных для импорта.
5. Размещение 90 свойств создаваемых ресурсов в поля импортируемой Таблицы данных. Поле **Источник/Выражение** может содержать поле, выбираемое из списка, или вводимое вручную выражение 112, ссылающееся на одно или более полей и/или производящее конвертацию типа/значения.

Система пытается самым оптимальным способом рассчитать наиболее подходящие размещения, основанные на именах полей и описаниях. В большинстве случаев, только некоторые распределения нужно определять вручную.

Пример 1: Чтобы конвертировать поле **phone** в международный формат во время импорта, вы можете задать следующее Источник/Выражение (для телефонных номеров США): `"+1" + {phone}`

Пример 2: Возможно, вы захотите использовать поле **comments** создаваемых источников для указания даты импорта. В данном случае Источник/Выражение не будет содержать ссылки на поля импортируемых данных, а будет выглядеть примерно так: `format(now(), "d MMM yyyy HH:mm:ss")`

6. Система начинает обработку каждой записи импортируемой Таблицы данных (файла). Сначала создается новый источник с именем, определенным при размещении поля Имя. После этого, обрабатываются все размещения и обновляются свойства только что созданного источника.
7. Отчет о статусе отображается 90 для пользователя. Он содержит список созданных объектов вместе с ошибками, которые возникли в процессе импорта.

Имя действия: import

Иконка действия: 

Не интерактивный режим 99**:** Не поддерживается

Права доступа: Доступно на уровне 486 прав доступа *Менеджер*.

3.6.8.9 Создать копию

Данное действие используется для клонирования какого-либо системного объекта, например, Тревоги или Виджета. Оно создает новый объект, затем копирует его свойства из исходного объекта в только что созданный при помощи операции копирования контекста.

Действия **Переместить** не существует. Для перемещения объекта скопируйте его, а затем удалите.

Имя действия: makeCopy

Иконка действия: 

Не интерактивный режим 99**:** Не поддерживается

Права доступа: Доступно на уровне 486 прав доступа *Менеджер*

Порядок действий

1. Создается новый контекст того же типа, что и копируемый. Его имя формируется автоматически путем добавления "_copy" к имени копируемого контекста. Его описание также формируется автоматически путем добавления слова "Copy" к Описанию оригинального контекста.

2. Пользователю предлагается 90 изменить имя и описание создаваемого контекста.

3. Все настройки копируются из исходного контекста в новый. Переменные, доступные только для чтения, не копируются. Для более подробной информации о логике операций копирования обратитесь к разделу Операции копирования контекста.

3.6.8.10 Просмотр событий

Данное действие помогает обнаружить и анализировать деятельность какого-либо аппаратного устройства или объекта системы. Это происходит в результате управления событиями 73, которые возникают в контексте. При запуске данного действия AtomMind Server находит все события в данном контексте, которые могут быть полезны для пользователей, и запускает Журнал событий при помощи процедуры пользовательского интерфейса Показать Журнал событий 96.

Теперь пользователь может контролировать все события в реальном времени, просматривать историю 73, фильтровать, сортировать, просматривать связанные с ними данные, подтверждать 73 события и т.д.

Имя действия: monitor

Иконка действия: 

Не интерактивный режим 99**:** Не поддерживается

Права доступа: Доступно на уровне 486 прав доступа *Наблюдатель*

Параметры 102 выполнения: Положение окна Журнала событий

Свойства инструментальной панели 926 Журнала событий

Примеры относительных событий

В контексте Избранное 1524, событие "info 77" ("информация") активируется во время создания нового избранного или удаления каких либо избранных объектов.

В контексте Устройство 1494, событие "event" ("Событие Устройства") активируется, когда от аппаратного устройства получено событие.

Фильтры относительных событий

Можно ограничить количество событий, отображаемых в Истории событий, используя фильтры, которыми вы фильтруете имеющиеся события. Например, вы можете отфильтровать отображаемые события по типу, дате или числу ваших событий. Дополнительные фильтры можно найти, кликнув по кнопке "Обновить".

3.6.8.11 Поиск/Фильтр

Данное действие открывает суб-дерево контекстного дерева сервера с текущим контекстом в качестве корня при помощи GUI процедуры Показать системное дерево 98.

Открываемое Системное дерево имеет инструментальную панель **Фильтр** для фильтрации или поиска контекстов.

Имя действия: filter

Иконка действия: 

Не интерактивный режим 99: Не поддерживает

Права доступа: Доступно на уровне 486 прав доступа *Наблюдатель*

3.6.8.12 Реплицировать

Данное действие перетаскивания (Drag-and-Drop) 101, используемое для копирования свойств из данного контекста в другой схожего типа. Оно позволяет вам копировать значения одного, нескольких или всех свойств данного контекста (таких как Учетная запись пользователя 1608 или Учетная запись устройтства 1494) из "принятого" контекста. Для более подробной информации обратитесь к Операциям копирования контекста.

Являясь действием перетаскивания 101, действие Размножить принимает контексты того же типа, что и контекст, в котором он определен. Таким образом, размножение свойств выполняется только между контекстами одного типа.

Имя действия: replicate

Иконка действия: 

Не интерактивный режим 99: Не поддерживаются

П: Доступно на уровне 486 прав доступа *Менеджер*

Порядок действий

1. UI процедура Редактировать данные 90 позволяет пользователю выбрать переменные и их поля для размножения. Возможно редактировать значения переменных в процессе.

2. AtomMind Server выполняет репликацию.

3. Отчет о статусе размножения показывается пользователю при помощи UI процедуры Редактировать данные 90 в режиме "Только чтение". Он включает одну запись на каждую размножаемую переменную. Данная запись отображает статус размножения (Успешно/Неуспешно) и ошибки, которые возникли в процессе размножения данной переменной.

3.6.8.13 Копировать в дочерние контексты

Данное действие перетаскивания 101, которое обычно определено в контекстах-контейнерах (таких как Тревоги 1452 или Запросы 829). Оно копирует настройки "принятого" контекста ко всем его потомкам. Это помогает сконфигурировать один, несколько или все свойства контекстов-потомков таким же способом, что и соответствующее свойства "принятого" контекста. Например, оно может размножить настройки "принятого" контекста Тревога 1454 во все контексты **Тревог**, определенных под контекстом **Тревоги**, определяющим данное действие. Для более подробное информации см. Операции копирования контекста

Являясь действием перетаскивания 101, действие Размножить потомкам принимает контексты того же типа, что и все потомки контекста, в котором оно определено. Таким образом, свойства могут быть размножены только между контекстами одного типа.

Имя действия: replicate

Иконка действия: 

Не интерактивный режим 99: Не поддерживается

Права доступа: Доступно на уровне 486 прав доступа *Менеджер*

Порядок действий

1. UI процедура Редактировать данные 90 позволяет пользователю выбирать переменные и их поля для размножения. Также возможно редактировать значения переменной в процессе.

2. Сервер выполняет репликацию.

3. Отчет о статусе размножения показывается пользователю при помощи UI процедуры Редактировать данные 90 в режиме "Только чтение". Он включает одну запись на каждую размножаемую переменную. Данная запись отображает статус размножения (Успешно/Неуспешно) и ошибки, которые возникли в процессе размножения данной переменной.

3.6.8.14 Показать статус

Позволяет пользователю посмотреть статус системного ресурса, отображая все переменные его статуса. Для большинства контекстов, единственной переменной статуса является Состав группы 67 и Активные тревоги 67. Другие переменные статуса зависят от контекста.

Например, для устройства данное действие показывает его онлайн статус, статус синхронизации, время последней синхронизации, прогресс синхронизации и отдельные статусы синхронизации настроек 506.

Тип действия: Конфигурировать 105 (режим "Только чтение")

Имя действия: status

Иконка действия: 

4 Выражения

AtomMind имеет собственный *язык выражений*. Выражения подобны формулам, используемым в динамических таблицах: эти формулы используются для ссылки на значения из окружающих ячеек, в то время как выражения AtomMind ссылаются на значения в пределах Единой модели данных 41.

Язык выражений AtomMind используется:

- В учетных записях устройств 497 для фильтрации входящих значений, оценки пользовательских статусов и т.д.
- В триггерах тревог 779 для поднятия тревоги, когда удовлетворяется сложное условие
- В фильтрах событий 762 для отбора событий, удовлетворяющих нескольким критериям
- В датчиках 2181 для мониторинга составных значений
- В отчетах 928 для получения исходных данных отчета
- В моделях 810 и инструментальных панелях 912 для привязки значений данных сервера и устройства между собой и с компонентами пользовательского интерфейса
- Для обработки бизнес-правил 813
- Для динамической ассоциации 749 ресурсов сервера друг с другом
- В графиках 1051 для расчета значений элементов данных
- Для построения запросов 829 "на лету"
- И почти во всех других аспектах системы

Выражения легко использовать благодаря автоматической конвертации типа.

Визуальный Редактор выражений 404 помогает создавать, тестировать и отлаживать сложные выражения.

Побочные эффекты

Вычисление выражения может иметь *побочные эффекты*. Например, если выражение содержит ссылку на функцию "Перезагрузить" контекста Устройство 1494, его вычисление заставит аппаратное устройство перезагрузиться. Это и называется побочным эффектом.

4.1 Синтаксис

Каждое выражение может состоять из следующих элементов:

- Литералы 114, т.е. числовые или строковые константы
- Операторы 115, такие как сложение или логическое ИЛИ
- Функции 124, которые осуществляют преобразование своих аргументов и/или влияют на единую модель данных 41 сервера
- Ссылки 117, которые загружают данные из единой модели данных и других источников
- Комментарии 139, используемые для описания сложных выражений

Выражение сравнимо с "предложением" или строчкой текста, которая обрабатывается AtomMind и возвращает результат. Например, если вам кто-нибудь скажет: "Все яблоки синие", вы обдумаете это в короткий промежуток времени и придете к выводу, что данное выражение является неправильным (или `FALSE` на языке программирования). Вы просто обработали выражение, сами по себе.

В AtomMind выражение является комбинацией *значений*, или *литералов* (`"True"`, `"False"`, `"5"`, `"John"`), *операторов* (таки как +, -, * и т.д., полный список см. далее) и *ссылок* (таких как `users.admin.deviceservers.ds1:buzz()`). Вы объединяете все это специальным способом, в результате получаете строку текста (выражение), которое AtomMind может понять и *вычислить*.

Вдобавок к обычным операторам и литералам язык выражений имеет большую библиотеку функций 124:

- Функции обработки чисел
- Функции обработки строк
- Функции обработки даты и времени
- Функции обработки цвета
- Функции обработки таблицы
- Функции доступа к контекстам
- И другие

Для привлечения к расчету внешних значений данных, почти каждое выражение включает одну или более ссылок 117.

Когда AtomMind *вычисляет* выражение, он его обрабатывает. Он систематически проходит через выражение, выполняя вызываемые им операции (получение значений переменных, запуск функций в составе выражения и получение их результатов, и т.д.), до окончания оценки и получает результат, который является *значением* данного выражения.

В AtomMind выражения широко используются по всей системе. Они всегда пишутся одним и тем же способом, как в языке программирования. Данный язык (или "способ составления ссылок") называется *Языком Выражений* AtomMind.

Результат выражения - это значение, которое может быть целым числом, строкой, логической переменной или любым другим типом данных, определенных в AtomMind. Тип значения зависит от того, что вычисляется: Выражение `1 + 1` вернет целое число (`2`), в то время как выражение `"the" + " dog"` вернет строку (`"the dog"`). Если выражение содержит ссылку (например, `{username} + 1`), нельзя точно сказать, какой тип значения получится, потому что он зависит от типа данных того, на что осуществляется ссылка.

4.2 Типы данных и конвертация типов

Язык выражений внутренне имеет дело со следующими типами (и возвращает значения этих типов):

Тип	Описание
Null	*Null* обозначает "отсутствие значения". Это значение может быть записано в ячейку таблицы данных только если поле может быть NULL 51.
Object	Супертип всех типов. Если объявляется, что функция возвращает Object, она может возвращать значения любого типа в зависимости от входных параметров.
Number	Супертип всех числовых типов. Если объявляется, что функция обрабатывает Number, она может обрабатывать данные типа Integer, Long, Float или Double. Дополнительно, все прочие объекты будут преобразованы в числовые, насколько это возможно.
Integer	32-битное целое число со знаком.
Long	64-битное целое число со знаком.
Float	32-битное число с плавающей запятой.
Double	64-битное число с плавающей запятой.
String	Строка любой длины. Строки могут также содержать бинарные данные.
Boolean	TRUE или FALSE
Color	8-битное значение цвета (RGB).
Date	Временная метка, содержащая как дату, так и время. Может использоваться функциями обработки даты/времени 124.
Data Table	Таблица данных 49.

Все типы имеют аналоги среди типов полей таблицы данных 52.

Конвертация типов

Движок обработки языка выражений делает все возможное для необходимой конвертации типов данных и вычисления выражения с получением результата. Например, если числовой аргумент передан функции, которая обрабатывает строки, число автоматически конвертируется в свою строковую репрезентацию. Другой пример: если число с плавающей запятой добавляется к целому числу, результирующее значение тоже будет числом с плавающей запятой. Это помогает избежать ошибок округления.

Если правильная конвертация типа невозможна, вычисление выражения выдает сбой с сообщением об ошибке.

Явная конвертация типов осуществляется с использованием функций конвертации типов 124.

4.3 Среда вычисления

Когда в AtomMind вычисляется выражение, существует ряд параметров, способных повлиять на результат. Полный набор данных параметров называется "среда вычисления". Большинство этих параметров влияют на преобразование ссылок 117 внутри выражения.

Среда вычисления включает в себя:

Контекст по умолчанию	Контекст по умолчанию 119, используемый для разрешения относительных контекстных путей, появляющихся в стандартных ссылках 118.
Таблица данных по умолчанию	Таблица данных по умолчанию 120, используемая для разрешения стандартных ссылок, которые неявно указывают на переменную/функцию определенного контекста.
Ряд по умолчанию	Ряд по умолчанию 119, используемая для определения, какой ряд в таблице данных должен быть доступен при разрешении стандартных ссылок, которые сами не указывают ряд.
Переменные среды	Любые переменные среды, которые извлекаются путем использования ссылок среды 123 внутри выражения.

4.4 Литералы

Литерал является константой. AtomMind рассматривает его "буквально" - система просто берет его, как есть, и использует в качестве основы вычислений и т.д. Это значение постоянно.

Язык выражения AtomMind определяет различные типы литералов:

- Литералы Null: `null`
- Литералы Boolean: `true` или `false`
- Десятичные литералы (`0, 1, 123, -1234567890, ...`)
- Шестнадцатеричные литералы (`0x0A, 0xFFFF, ...`)
- Двоичные литералы (`0b01, 0b00110011`)
- Восьмеричные литералы (`00, 055, 01234567, ...`)
- Литералы с плавающей запятой (`3.1, -44.5, 1.3E12, ...`)
- Строковые литералы (`"This is a String", 'test', ...`)

Экранирование строк

Обратные слеши (`\`), появляющиеся в строках, должны быть экранированы при помощи другого обратного слеша (т.е. `\` становится `\\`).

Пример: `"Строка с одним обратным слешем: \\ "`

Кавычки также должны быть экранированы, используя обратный слеш (`\"` , `\'`), если тот же тип кавычек применялся в строке.

Примеры:

- `"У этой строки \"внутри\" есть двойные кавычки"`
- `У этой строки \'внутри\' есть одинарные кавычки'`

Символ, стоящий после обратного слеша (\), является управляющей последовательностью и имеет особое значение. Данная таблица показывает управляющие последовательности:

Управляющая последовательность	Описание
\t	Соответствует табуляции.
\b	Соответствует клавише "стереть влево".
\n	Соответствует новой строке.
\r	Соответствует каретке.
\f	Соответствует прогону страницы.

Символы Unicode

Символы Unicode могут вставляться в строковые литералы путем использования следующей синтаксической конструкции: *\uhhhh*

Например, для вставки символа Unicode с шестнадцатиричным кодом 90FA используйте следующий синтаксис: `\u90FA`

4.5 Операторы

Оператором является обозначение, которое применяется к определенным данным (таким, как числа или строки) и сообщает системе, что с ними нужно делать. Как только система выполнила оператор для данных (т.е. как только оператор "сработал"), вы всегда получаете какой-либо результат или значение. Общие операторы включают + (сложение, как в 1+1), - (вычитание, как в 2-1) и т.д.

Одно выражение может содержаться несколько операторов, таких как 1 + 2 * 3. Операторы вычисляются в определенной последовательности - не так, как они записаны в строке (слева направо). Например, 1 + 2 * 3 вычисляется в 7, а не в 9, потому что оператор умножения имеет высший приоритет по сравнению с оператором сложения. Последующая таблица также показывает приоритет операторов - операторы, расположенные вверху таблицы, вычисляются первыми при вычислении выражения.

Приоритетнось операторов

Описание	Синтаксис	Пример	Типы аргументов	Тип результата
Скобки	()	`(1 + 2) * 3`	Any	Any
Побитовый НЕ оператор	~	`~{a}`	Numeric	Numeric
Логический НЕ оператор	!	`!{x} > 0`	Boolean	Boolean
Мультипликативные операторы	* / %	`{d} * 3`	Numeric	Numeric
Аддитивные операторы	+ -	`{b} + {c}`	Numeric (или String для "+")	Numeric (или String для "+")
Операторы битового сдвига и битового сдвига без знака	>> << >>>	`{a} >> 2`	Numeric	Numeric
Операторы отношения	> < >= <=	`{a} >= 123`	Numeric	Boolean
Операторы равенства и совпадения с	== != ~=	`{name} == "test"`	Любой (Strings только для "!=")	Boolean

регулярным выражением2142				
Битовый И оператор	&	{z} & 0xFFFF	Numeric	Numeric
Битовый "исключающий ИЛИ" оператор	^	{y} ^ 0x1A	Numeric	Numeric
Битовый ИЛИ оператор	|	{x} | 0xFF	Numeric	Numeric
Логический И оператор	&&	{z} = 1 && {x} > 0	Boolean	Boolean
Логический ИЛИ оператор	||	{x} > 5 || {y} < 1	Boolean	Boolean
Условный оператор	? :	{y} > 5 ? 1 : 0	Boolean для первого аргумента, любой тип для других аргументов	Любой

Скобки

Выражения могут также включать скобки, которые меняют приоритет операторов.

Пример:

`1 + 2 * 3` вычисляются в 7

но

`(1 + 2) * 3` вычисляются в 9

Условный оператор

Условный оператор, `?:`, может использоваться для условного вычисления выражений. Оператор состоит из трех аргументов. Первый аргумент вычисляется и возвращает значение типа Boolean. Если он возвращает `TRUE`, результатом оператора станет второй аргумент. Если первый аргумент является `FALSE`, результатом оператора станет третий аргумент.

```
condition ? value_if_true : value_if_false
```

Пример:

```
2 * 2 == 4 ? "Yes" : "No"
```

Разрешается в строку "Yes".

Оператор совпадения с регулярным выражением

Оба аргумента являются строками. Оператор возвращает true, если первая строка совпадает с регулярным выражением2142, определяемым вторым аргументом.

Пример:

```
"abXXXX" ~= "^abc.*"
```

Возвращает Boolean *false*.

Аддитивный оператор

Если хотя бы один из аргументов - строка, второй аргумент конвертируется в его строковое представление. В данном случае, результатом вычисления становится результат сцепления этих строк.

Пример:

```
"test" + 123
```

Имеет результат "test123".

4.6 Ссылки

Ссылки используются для указания местоположения данных. В зависимости от контекста, ссылки можно использовать для чтения данных (т.е. данные будут извлекаться из тех мест, на которые указывает ссылка) или для их записи (данные сохранятся в месте, на которое указывает ссылка).

Ссылки могут указывать на любое значение, представляющее внутренние данные AtomMind. Общие правила синтаксиса ссылки приведены в примере:

```
schema/server^context:entity(parameter_list)$field[row]#property
```

Все части ссылки являются опциональными, поэтому самая короткая возможная ссылка - это пустая строка.

Использование ссылок в выражениях

При использовании внутри выражений, *ссылки* должны заключаться в фигурные скобки.

Ссылки выполняются при вычислении выражения. Выглядит это так: `{temperature} <= 100`. В данном случае `temperature` - это ссылка.

Схема ссылок

Схема ссылок определяет, как система разрешает оставшуюся часть ссылки. Она помогает системе "понять", что означает оставшаяся часть ссылки. Схемы широко применяются в программировании. Например, в адресе `http://www.google.com` часть **http://** является схемой. Она сообщает браузеру, что это ссылка HTTP и что он должен рассматривать оставшуюся часть строки соответствующим образом. Ссылка FTP может отличаться по структуре, например, `ftp://joeuser:xxx@gmail.com`. В этом случае тоже, часть **ftp://**определяет схему, и теперь ваша программа знает, как обращаться с окончанием ссылки.

В AtomMind Схемы используются для ссылки на различные части системы:

`env/` схема используется для ссылки на переменные среды 123 (которые являются свойствами, моментально принимающими во внимание выполнение текущей операции), `form/` схема используется в конструкторе UI для ссылки на свойства графических компонентов, из которых состоит виджет.

Схема является опциональной: вам необязательно включать ее в свою ссылку. Она просто говорит системе, что ваша ссылка указывает на переменную среды (`env/`) или графический компонент (`form/`). Если ваша ссылка указывает на данные из переменной контекста, функцию или ячейку Таблицы данных и т.д., вы можете опустить схему и записать ссылку в формате по умолчанию, который описан в следующем разделе (Разрешение ссылок).

Например, ссылка `form/TextField1#text` использует `form/` схему и указывает на свойство "text" компонента виджета под названием "TextField1". Ссылка `users.admin:childInfo$firstname` не включает в себя схему и указывает на имя пользователя "admin" (которое является полем переменной `childInfo`).

СХЕМЫ ССЫЛОК, ИСПОЛЬЗУЕМЫЕ В ATOMMIND

Схема	Описание
Не задана	Указывает, что ссылка является стандартной 118.
action/	Используется для запуска действия 87, например из цели 1295 привязки 1295 виджета 943 или инструментальной панели 223.
env/	Указывает на ссылку среды 123.
form/	Ссылка на компонент UI. Подобные ссылки используются в выражениях привязки виджета 943 и инструментальной панели 223.
menu/	Используется внутри активатора 1296 привязки 1295 виджета 943 или инструментальной панели 223 для реагирования на выбор элемента контекстного меню 1276.
previous/	Используется в выражениях массива данных графиков (см. свойства компонента График Данные, основанные на событии 1064 и Данные, основанные на переменной 1061). Они используют тот же формат, что и стандартные ссылки 118. Однако, Таблица данных по умолчанию 120, на которую указывает такая ссылка, является Таблицей данных, содержащей значение для *предыдущей* точки данных (т.е. данные события или значение измененной переменной). См. данный пример 124, чтобы понять, как ссылка `previous/` может использоваться для рассчета различия между двумя точками данных.
statistics/	Используется для ссылки на статистические 718 данные из массива данных графика 1063.

table/	Когда используется в качестве цели привязки таблицы данных 741, дает команду системе заменить всю таблицу, чьи привязки обрабатываются, на таблицу, возвращенную текущим выражением привязки.

Разрешение ссылок

Разрешение ссылок - процесс получения текущего значения из местоположения, на которое указывает ссылка. Алгоритм разрешения зависит от схемы ссылки.

Если ссылка не задает схему, она разрешается, как стандартная 118.

4.6.1 Стандартные ссылки

Стандартные ссылки могут указывать на:

- Таблицы данных 49, содержащие значения переменных контекста, ввод/вывод функций контекста или другие данные.
- Отдельные ячейки вышеобозначенных таблиц.
- Свойства определений переменных/функций/событий контекста (их описания или флажки, указывающие доступна ли переменная для чтения/записи).
- Свойства полей таблиц данных (например, тексты их описания и "справки").
- И другие элементы единой модели данных 41.

Полный синтаксис стандартной ссылки: `context:entity(parameter_list)$field[row]#property`

В большинстве случаев использование ссылок может быть заменено на использование функций 124 для работы с **контекстами** и **таблицами**. Ссылки будут в большинстве случае выглядеть короче и проще, но их использование не принесет никакой выгоды во время вычисления выражения по сравнению с использованием функций.

Варианты синтаксиса

Таблица показывает и описывает различные возможные варианты синтаксиса стандартных ссылок:

Синтаксис	Описание	Аналог на базе функций
`field`	Возвращает значение ячейки таблицы данных 120, указанной **field** и рядом по умолчанию 119.	`cell(dt(), "field")`
`field[row]`	Возвращает значение ячейки таблицы данных по умолчанию 120, указанной **field** и **row**.	`cell(dt(), "field", row)`
`variable$`	Возвращает таблицу данных, представляющую значение **variable** в контексте по умолчанию 119.	`getVariable(dc(), "variable")`
`function(parameter_list)`	Вызывает **function** контекста по умолчанию 119 с параметрами, указанными в **parameter_list**. Возвращает таблицу данных, представляющую выход функции.	`callFunction(dc(), "function", parameter_list)`
`variable$field`	Получает таблицу данных, представляющую значение **variable** из контекста по умолчанию 119. Возвращает значение ячейки этой таблицы данных, указанной **field** и рядом по умолчанию 119.	`cell(getVariable(dc(), "variable"), "field")`
`variable$field[row]`	Получает таблицу данных, представляющую значение **variable** из контекста по умолчанию 119. Возвращает значение ячейки этой таблицы данных, указанной **field** и **row**.	`cell(getVariable(dc(), "variable"), "field", row)`
`function(parameter_list)$field`	Получает таблицу данных, представляющую выходное значение, возвращаемое **function** контекста по умолчанию 119, которая вызывается с	`cell(callFunction(dc(), "function", parameter_list), "field")`

	параметрами из **parameter_list**. Возвращает значение ячейки этой таблицы, указанной **field** и рядом по умолчанию 119.	
`function(parameter_list)$field[row]`	Получает таблицу данных, представляющую выходное значение, возвращаемое **function** контекста по умолчанию 119, которая вызывается с параметрами из **parameter_list**. Возвращает значение ячейки этой таблицы данных, указанной **field** и **row.**	`cell(callFunction(dc(), "function", parameter_list), "field", row)`
`context:variable`	Возвращает таблицу данных, представляющую значение **variable** в **контексте (context)**.	`getVariable("context", "variable")`
`context:function(parameter_list)`	Вызывает **function** в контексте **(context)** с параметрами, указанными в **parameter_list**. Возвращает таблицу данных, представляющую выход функции.	`callFunction("context", "function", parameter_list)`
`context:variable$field`	Возвращает таблицу данных, представляющую значение **variable** в **контексте (context)**. Возвращает значение ячейки этой таблицы данных, указанной **field** и рядом по умолчанию 119.	`cell(getVariable("context", "variable"), "field")`
`context:variable$field[row]`	Получает таблицу данных, представляющую значение **variable** из **context**. Возвращает значение ячейки этой таблицы данных, указанной **field** и **row.**	`cell(getVariable("context", "variable"), "field", row)`
`context:function(parameter_list)$field`	Получает таблицу данных, представляющую выходное значение, возвращаемое **function** контекста **(context),** которая вызывается с параметрами из **parameter_list**. Возвращает значение ячейки этой таблицы, указанной **field** и рядом по умолчанию 119.	`cell(callFunction("context", "function", parameter_list), "field")`
`context:function(parameter_list)$field[row]`	Получает таблицу данных, представляющую выходное значение, возвращаемое **function** контекста **(context)** которая вызывается с параметрами из **parameter_list**. Возвращает значение ячейки этой таблицы данных, указанной **field** и **row.**	`cell(callFunction("context", "function", parameter_list), "field", row)`
`.:`	Возвращает путь контекста по умолчанию 119.	`dc()`
`(empty reference text)`	Возвращает таблицу по умолчанию 120.	`dt()`

Ряд по умолчанию

Если **ряд** не задан, значение берется из:

- *текущего* ряда, если обрабатывается каждый ряд таблицы данных в текущей среде;
- ряда 0 (первый ряд в Таблице данных) в остальных случаях.

Полный синтаксис: `context:entity(parameter_list)$field`

Значение из Контекста по умолчанию 119: `entity(parameter_list)$field`

Значение из Таблицы данных по умолчанию 120: `field`

Контекст по умолчанию

Если в ссылке задан **контекст**, то данные берутся из него.

Если контекст не определен, данные берутся из *контекста по умолчанию*.

Контекст по умолчанию автоматически определяется согласно тому, что вы сейчас делаете. Например, при фильтрации событий, контекстом по умолчанию станет контекст, в котором появилось данное событие (с которым мы работаем). Можно сказать, что контекстом по умолчанию является "текущий контекст".

Пример: При работе с виджетами 946, контекстом по умолчанию является тот, из которого запускается виджет.

Относительные 43 пути контекстов (например, `devices.device1`) разрешаются, начиная с данного контекста по умолчанию.

Пример: `childInfo$firstname` ссылается на контекст по умолчанию, в то время как `users.admin:childInfo$firstname` ссылается на определенный контекст.

`{.:}` ссылка вернет путь контекста по умолчанию. `dc()` функция также вернет этот путь.

Таблица данных по умолчанию

Подобно тому, что вы можете опустить контекстную часть ссылки и всё равно она будет пригодной, вы можете пропустить объектную часть (entity part). При ее пропуске, предполагается, что вы ссылаетесь на так называемую таблицу данных "по умолчанию", которая также называется "текущая таблица данных". Например, при фильтрации событий, таблица данных по умолчанию - таблица для фильтруемого на данный момент события.

Примером использования Таблицы данных по умолчанию может стать ссылка `firstname`. Это ссылка целиком -- просто `firstname`. Во время ее записи, вы предполагаете, что система "знает, что вы имеете в виду" -- т.е. она будет обращаться к полю `firstname` таблицы данных для каждого фильтруемого события, в случае фильтрации событий.

`{}` (пустая) ссылка вернет целую таблицу данных (которая может быть использована, например, функциями обработки таблиц данных 124). Функция `dt()` также вернет эту таблицу.

Разрешение объекта

Часть **объекта**, заданная в ссылке, является именем переменной или функции в контексте, на который вы ссылаетесь. Она рассматривается в качестве функции, если ссылка содержит в скобках **parameter_list**, и в качестве переменной, если не содержит. Таким образом, `users.admin:childInfo` ссылается на переменную, в то время как `users.admin:delete()`ссылается на функцию. Если объект, на который вы ссылаетесь (например, `delete()`), не существует, разрешение не будет выполнено.

Если ссылка указывает на функцию, она вызывается параметрами, заданными при помощи **parameter_list** во время разрешения.

Пример: Разрешение ссылки `:register("charlie", "12345", "12345")` приведет к выполнению функции **register** из корневого контекста.

СПИСОК ПАРАМЕТРОВ ФУНКЦИИ

parameter_list используется для создания таблицы данных согласно формату передаваемых функции параметров, как описано здесь.

ОБРАБОТКА ОШИБОК

Если возникает ошибка при получении значения переменной вызываемой функции, разрешение ссылки будет не выполнено.

Разрешение свойств

Иногда вам может понадобиться задать *свойство* для определения какого-либо свойства контекста, переменной, функции, таблицы данных или поля таблицы данных.

Возможно у вас возникнет вопрос, в чем разница между *полем* и *свойством*. *Поле* содержит актуальные данные, а **свойство** - "метаданные", то есть данные о данных. Предположим, мы работаем с переменной `version` контекста `server\` (его путь - `server:version`). Переменная *содержит* поле (тоже с именем

version) с актуальными данными, например, "4.01.00" (версия сервера). Вы можете ссылаться на данное поле через server:version$version.

Та же переменная также имеет *свойство* description, содержащее данные *о* переменной. В этом случае, свойство будет содержать строку, "Версия сервера". Свойство не сохраняется в Таблице данных переменной - наоборот, оно "касается" определения переменной (а не ее значения). Вы можете сослаться на данное свойство через server:version#description.

Поэтому при ссылке на поле вы используете знак $, а при ссылке на свойство - знак #. Это важно, потому что поля часто *имеют* свойства (т.е. поле имеет описание). Более подробно об этом см. далее.

СВОЙСТВА КОНТЕКСТА

Синтаксис ссылки на свойство контекста: context:#property

Свойство	Тип	Описание
name	String	Имя контекста
description	String	Описание контекста.
type	String	Тип 43 контекста.
icon	Image	Иконка контекста. Может использоваться компонентом виджета 943 Изображение 993.

СВОЙСТВА ОПРЕДЕЛЕНИЯ ПЕРЕМЕННОЙ

Синтаксис ссылки на свойство определения переменной: context:variable#property

Свойство	Тип	Описание
description	String	Описание переменной.
icon	Image	Иконка свойства. Может использоваться компонентом виджета 943 Изображение 993.
readable	Boolean	True, если переменная доступна для чтения для текущего пользователя.
writable	Boolean	True, если переменная доступна для записи для текущего пользователя.

СВОЙСТВА ОПРЕДЕЛЕНИЯ ФУНКЦИИ

Синтаксис ссылки на свойство определения функции: context:function(parameter_list)#property

Свойство	Тип	Описание
description	String	Описание функции

СВОЙСТВА ТАБЛИЦЫ ДАННЫХ

Синтаксис ссылки на свойства Таблицы данных: context:entity(parameter_list)#property

Свойство	Тип	Описание
records	Integer	Количество записей в таблице. Может быть полезным для вычисления, сколько записей было получено в ответ на вашу ссылку. Например, users:list() #records вызывает функцию, просматривающую всех пользователей в системе и возвращает число записей в результате (т.е. сколько пользователей вы можете увидеть в системе согласно вашим правам доступа).
quality	Integer	Качество таблицы данных. Объясняет, насколько надежен экземпляр данных, представленный таблицей данных.
timestamp	Date	Временная метка значения таблицы данных. Некоторые части системы предоставляют информацию о времени вместе с образцом данных. Например, некоторые типы драйверов в качестве значения для этого свойства используют временную метку физического устройства, если это предусмотрено коммуникационным протоколом.

СВОЙСТВА ПОЛЯ ТАБЛИЦЫ ДАННЫХ

Синтаксис ссылки на свойство поля Таблицы данных: `context:entity(parameter_list)$field#property`

Свойство	Тип	Описание
description	String	Описание поля
help	String	"Справка" поля (подробное описание)
svdesc	String	Описание значения выборки, т.е. строки, используемой для представления значения текущего поля в пользовательском интерфейсе.

Например, `users.admin:childInfo$firstname` возвращает что-то похожее на "John" (значение поля), в то время как `users.admin:childInfo$firstname#description` вернет нечто похожее на "Имя пользователя" (свойство описания поля).

ССЫЛКИ, СОСТОЯЩИЕ ТОЛЬКО ИЗ СВОЙСТВ

Если ссылка состоит только из **свойства** (например, `#property`), она разрешается в следующее свойство:

Свойство	Тип	Описание
row	Integer	Данное свойство разрешается в текущий ряд обрабатываемой Таблицы данных. Например, когда виджет график 1051 основан на пользовательских данных 1057, его таблица **Исходные данные** обрабатывается построчно. Рассчитываются выражения привязки Исходных данных для каждой строки. Таким образом, мы можем использовать `{#row}` в любом выражении Привязки исходных данных для создания последовательной серии или категории имен (**0, 1, 2, ...**).

Примеры

Текст ссылки	Комментарии
`recipient`	Данная ссылка возвращает значение поля `recipient` в таблице данных по умолчанию ("текущая таблица"). Это полезно, например, во время работы с выражениями фильтра событий 762.
`recipient[3]`	То же, что и выше, но значение берется из четвертого ряда.
`users.admin:childInfo$email`	Разрешается в значение поля `email` в Таблице данных, содержащего значение для переменной `childInfo` контекста `user.admin`. В данном случае будет содержать email адрес системного администратора.
`:info$description`	Данный пример начинается с двоеточия. Поэтому она ссылается на контекст "" (пустую строку), что является именем корневого контекста. Таким образом, ссылка вернет описание поля переменной info корневого контекста. Имейте в виду, что "description" - имя **свойства**, которое также может быть именем **поля**. Различие заключается в том, используете ли вы `#` или `$` для ссылки на него.
`DS_setting#description`	Разрешается в описание *свойства поля* `DS_setting` в таблице данных по умолчанию.
`users.test.deviceservers.ds1:buzz()`	Разрешается в Таблицу данных, полученную в результате выполнения функции "Показать все устройства" в контексте `users.test.deviceservers`. Чтобы получить таблицу данных, должна быть вызвана функция "Показать все устройства", что заставит устройство `ds1` Device Server моргать светодиодами.
`.:childInfo$firstname`	Разрешается в значение поля `firstname` переменной `childInfo` контекста по умолчанию ("текущего" контекста). Данная ссылка использует относительный контекстный путь (".."). Вы можете ничего не ставить перед двоеточием, однако, точка будет работать точно так же. Вы получите строку с именем пользователя ("Joe").

`.:childInfo$country#svdesc`	Возвращает название страны пользователя (т.е. описание значения выборки поля `country`). Имейте в виду, что `.:childInfo$country` вернет системный цифровой код страны.
`users.admin.deviceservers.ds1.devices.1:selfTest("quick")$status`	Данная ссылка указывает на поле `status`, возвращаемое функцией `selfTest`, определенной в контексте `users.admin.deviceservers.ds1.devices.1`. Данная функция вызывается с одним параметром.
`attendance:dailyActivityData('{.:}','{date}')`	Данная ссылка разрешится в Таблицу данных, полученную в результате выполнения функции `dailyActivityData`, определенной в контексте `attendance`. Данная функция вызывается с двумя параметрами: • Результат оценки выражения `{.:}`, т.е. путь к контексту по умолчанию (`.`). • Результат оценки выражения `{date}`, т.е. значение поля `date` в первом ряду таблицы данных по умолчанию.
`users.admin#icon`	Укажет на иконку контекста пользователя ().

4.6.2 Ссылки среды

Ссылки среды указывают на специальные *переменные среды*, определенные во время оценки выражения. Например, ссылки среды могут использоваться в выражении фильтра событий[765]: в его состав входят переменная среды `context`, указывающая на контекст, в котором произошло событие, и переменная `level`, содержащая его критичность (подробнее об этом см. здесь[768]).

Переменные среды определяются системой. Не существует способа задать новые переменные среды или изменить их значения.

Ссылки на переменные среды форматируются следующим образом:

`env/xxx`: `env` -- заранее определенная строка (не меняйте ее). Она нужна для указания *схемы*, которую вы используете (т.е. переменные среды) и, таким образом, указывает, что ссылка указывает на переменную среды. `xxx` - имя переменной. Переменная среды не одно и то же с переменной контекста[61], поэтому у нее нет пути. Она ссылается на выполняемую в данный момент операцию - вы просто вписываете ее имя.

Пример:

`env/level`

Данная переменная среды содержит уровень критичности текущего обрабатываемого события. В данном случае, этот пример может показаться не практичным, чтобы исправить это, нужно вставить его в *Выражение AtomMind*, что мы и будем делать в следующей главе.

В данной документации описание среды вычисления[114] любого выражения содержит перечень переменных среды, доступных в процессе вычисления.

СТАНДАРТНЫЕ ПЕРЕМЕННЫЕ СРЕДЫ

Некоторые переменные определены в любой среде:

Имя переменной	Описание
count	Количество вычислений, выполненных в текущей переменной. Равняется нулю во время первого вычисления и увеличивается на 1 с каждым вычислением.

	Например, если ваше выражение, ссылающееся на `{env/count}`, используется для рассчета значения данных для графика, вы можете вписать что-то похожее на `{env/count} > 0 ? {value} - {previous/value} : null` для отмены вычисления первого значения данных (потому что `{previous/value}` еще не определено).
previous	Результат предыдущего вычисления в текущей среде.
time	Используется в выражениях массива данных графика (см. свойства Данные, основанные на событии 1064 и Данные, основанные на переменной 1061 компонента График). Возвращает временную метку обрабатываемой на данный момент точки данных (время возникновения события или время изменения переменной). Переменные среды **time** и **previousTime** могут использоваться для создания временных графиков значений, которые увеличиваются при каждом измерении. Например, если ссылка `{incomingBytes}` возвращает количество байт, полученных устройством с момента запуска, вы можете создать график трафика при помощи следующего выражения: `{previousTime} != null ? ({incomingBytes} - {previous/incomingBytes}) / ({env/time} - {env/previousTime}) : null`
previousTime	Также используется в выражениях массива данных графика. Возвращает временную метку предыдущей точки данных или NULL для первой.

4.7 Функции

Функции языка выражений выполняют обработку входных параметров и возвращают какое-либо значение. Некоторые функции (такие как **random**) не имеют входных параметров.

Важно различать функции языка выражений и контекстные функции 70, которые могут быть в составе ссылок 112.

4.7.1 Функции обработки чисел

Раздел описывает функции языка выражений, относящиеся к обработке целых чисел и чисел с плавающей точкой.

Функция	**Описание**	**Тип результата**
abs(Double value)	Возвращает абсолютное значение. Если аргумент не отрицательный, он возвращается. Если же аргумент отрицательный, возвращается отрицание аргумента. Если тип аргумента является Целым или Длинным, возвращаемое значение будет типа Длинное.	Double
acos(Double value)	Возвращает арккосинус угла, в диапазоне от 0.0 до pi.	Double
asin(Double value)	Возвращает арксинус угла, в диапазоне от -pi/2 до pi/2.	Double
atan(Double value)	Возвращает арктангенс угла в диапазоне от -pi/2 до pi/2.	Double
cbrt(Double value)	Возвращает кубический корень значения.	Double
ceil(Double value)	Возвращает наименьшее (близкое к отрицательной бесконечности) значение, которое больше или равно аргументу и является математическим целым числом.	Double
cos(Double value)	Возвращает тригонометрический косинус угла. Угол измеряется в радианах.	Double
cosh(Double value)	Возвращает гиперболический косинус значения	Double
e()	Возвращает основание натурального логарифма.	Double

eq(Long first, Long second)	Возвращает true, если *первый* аргумент равен *второму* аргументу (то же, что и оператор = =).	Булево
exp(Double value)	Возвращает число Эйлера e, возведенное в степень, равную значению.	Double
floor(Double value)	Возвращает самое большое (близкое к положительной бесконечности) значение, которое меньше или равно аргументу и является целым числом.	Double
formatNumber(Number number, String pattern)	Форматирует числовое значение в строку. Образцы форматирования чисел описаны здесь.	String
ge(Long first, Long second)	Возвращает true, если *первый* аргумент больше или равен *второму* (то же, что и оператор >=).	Boolean
gt(Long first, Long second)	Возвращает true, если *первый* аргумент больше *второго* (то же, что и оператор >).	Boolean
le(Long first, Long second)	Возвращает true, если *первый* аргумент меньше или равен *второму* (то же, что оператор <=).	Boolean
log(Double value)	Возвращает натуральный логарифм значения (по основанию e).	Double
log10(Double value)	Возвращает базовый десятичный логарифм значения.	Double
lt(Long first, Long second)	Возвращает true, если *первый* аргумент меньше *второго* (то же, что и оператор <).	Boolean
min(Double first, Double second)	Возвращает меньшее из двух значений. Если типы аргументов являются Целыми или Длинными, возвращает значение типа Длинное.	Double
max(Double first, Double second)	Возвращает большее из двух значений. Если типы аргументов являются Целыми или Длинными, возвращает значение типа Длинное.	Double
ne(Long first, Long second)	Возвращает true, если *первый* аргумент не равен *второму* (то же, что и оператор !=).	Boolean
pi()	Возвращает отношение длины контура окружности к ее диаметру.	Double
pow(Double base, Double power)	Возвращает значение первого аргумента, возведенное в степень второго.	Double
random()	Возвращает значение с положительным знаком, большее или равное 0.0 и меньшее 1.0. Полученные значения выбираются псевдослучайно с относительно равномерным распределением в этом диапазоне.	Double
round(Double value)	Возвращает ближайшее целое число к аргументу.	Long
signum(Double value)	Возвращает сигнум-функцию аргумента; 0, если аргумент равен 0; 1.0, если аргумент больше ноля; -1.0, если аргумент меньше нуля. Если тип аргументов Целое или Длинное, возвращает значение типа Длинное.	Double
sin(Double value)	Возвращает тригонометрический синус угла. Угол измеряется в радианах.	Double
sinh(Double value)	Возвращает гиперболический синус значения.	Double
sqrt(Double value)	Возвращает правильно округленный квадратный корень значения.	Double
tan(Double value)	Возвращает тригонометрический тангенс угла. Угол измеряется в радианах.	Double
tanh(Double value)	Возвращает гиперболический тангенс значения.	Double

4.7.2 Функции обработки строк

Раздел описывает функции языка выражений, относящиеся к обработке строк.

Функция	**Описание**	**Тип результата**

contains(String string, String substring)	Возвращает true, только если *строка* содержит определенную *подстроку*.	Boolean
endsWith(String string, String suffix)	Возвращает true, если строка заканчивается на специальный *суффикс*. Результат будет true, если *суффикс* является пустой строкой или равен аргументу *строки*.	Boolean
format(String pattern, Object parameter1, ...)	Форматирует несколько *параметров* в строку, используя предоставленный *образец*. См. Форматирование общих объектов 2149.	String
groups(String source, String regex)	Сопоставляет строку-*источник* с данным регулярным выражением и возвращает найденные значения групп регулярных выражений. См. раздел Группы и сбор данных 2145 в приложении Синтаксис Регулярных Выражений, чтобы узнать, как определять и использовать группы. Если найдена всего лишь одна группа, ее содержание возвращается в виде Строки. Если было найдено больше групп, эта функция возвращает Таблицу Данных с единственной записью и множеством строковых полей, содержащих значения всех групп. Группа 0 (целое выражение) никогда не возвращается и не считается.	Object
isDigit(String character)	Возвращает true, если первый символ строки *символов* является цифрой.	Boolean
isLetter(String character)	Возвращает true, если первый символ строки *символов* является буквой.	Boolean
isLowerCase(String character)	Возвращает true, если первый символ строки *символов* находится в нижнем регистре.	Boolean
isUpperCase(String character)	Возвращает true, если первый символ строки *символов* находится в верхнем регистре.	Boolean
isWhitespace(String character)	Возвращает true, если первый символ строки *символов* является пустой областью.	Boolean
index(String string, String substring [, Integer fromIndex])	Возвращает порядковый номер символа в *строке*, начиная с 0, первого вхождения *субстроки* в первый аргумент или -1, если таковая не обнаружена.	Integer
lastIndex(String string, String substring, Integer fromIndex)	Возвращает порядковый номер символа в *строке*, начиная с 0, первого вхождения справа *субстроки* в первый аргумент или -1, если не обнаружено. Поиск идет с fromIndex.	Integer
length(String string)	Возвращает длину *строки*.	Integer
lower(String string)	Конвертирует все символы в *строке* в нижний регистр.	String
replace(String string, String target, String replacement)	Заменяет каждую субстроку *строки*, которая совпадает с целевой субстрокой target заданной строкой *replacement*. Замещение происходит от начала строки к концу, например, замещение "aa" на "b" в строке "aaa" даст "ba", а не "ab".	String
split(String string, String regex [, String fieldName [, Integer limit]])	Разбивает эту строку на совпадения данного регулярного выражения 2142. Таблица, возвращенная этой функцией, содержит каждую подстроку данной строки, что прерывается другой подстрокой, соответствующей данному регулярному выражению, или прекращается к концу строки. Подстроки в таблице стоят в том порядке, в котором они имели место в этой строке. Если выражение не соответствует какой-либо части ввода, то итоговая таблица имеет только одну запись, а именно эту строку. Параметр *предел* контролирует, сколько раз применяется шаблон, влияя на количество записей в результирующей таблице. Если предел **n** больше нуля, то образец будет применяться не чаще **n - 1** раз, длина таблицы не будет	DataTable

	больше, чем **n**, и последняя запись таблицы будет содержать весь ввод за пределами последнего подходящего символа-разграничителя. Если **n** не является положительным, то образец будет применяться столько раз, сколько это возможно, и таблица может иметь любую длину. Если **n** равен нулю (это значение по умолчанию), то образец будет применяться столько раз, сколько это возможно, таблица может иметь любую длину, и конечные пустые строки будут отброшены.
	
	Результирующая таблица имеет единственное поле Строка под названием *fieldName* (по умолчанию - `element`). Значения в этом столбце представляют элементы исходящей строки.
	
	Примеры:
	
	`split("boo:and:foo", ":", "field", 2)`
	возвращает таблицу с двумя записями: "boo", "and:foo"
	
	`split("boo:and:foo", ":", "field", 5)`
	возвращает таблицу с тремя записями: "boo", "and", "foo"
	
	`split("boo:and:foo", ":", "field", -2)`
	возвращает таблицу с тремя записями: "boo", "and", "foo"
	
	`split("boo:and:foo", "o", "field", 5)`
	возвращает таблицу с пятью записями: "b", "", ":and:f", "", ""
	
	`split("boo:and:foo", "o", "field", -2)`
	возвращает таблицу с пятью записями: "b", "", ":and:f", "", ""
	
	`split("boo:and:foo", "o", "field", 0)`
	возвращает таблицу с тремя записями: "b", "", ":and:f"	

startsWith(String string, String prefix)	Возвращает true, если строка начинается с заданного *префикса*. Имейте в виду, что вернется true, если *префикс* является пустой строкой или равен *строке*.	Boolean
substring(String string, Integer beginIndex [, Integer endIndex])	Возвращает новую строку, которая является субстрокой *строки*. Субстрока начинается с символа, заданного *beginIndex* и заканчивается в конце строки символом *endIndex – 1* или последним символом исходной строки, если параметр *endIndex* не указан. Примеры: `substring("unhappy", 2)` вернет "happy" `substring("Harbison", 3)` вернет "bison" `substring("emptiness", 9)` вернет "" (пустую строку) `substring("hamburger", 4, 8)` вернет "urge" `substring("smiles", 1, 5)` вернет "mile"	String
trim(String string)	Возвращает копию *строки* без начального и конечного пробелов.	String
upper(String string)	Конвертирует все символы в *строке* в верхний регистр.	String
urlDecode(String string, String encoding)	Декодирует *строку* **application/x-www-form-urlencoded**, используя специальную схему кодирования. Данное *кодирование* используется для определения символов, представленных любыми последовательностями формы `%xy`.	String

	Примечание: Рекомендации Консорциума Всемирной Паутины утверждают, что для кодирования нужно использовать UTF-8. Если этого не делать, может произойти несовместимость.	
urlEncode(String string, String encoding)	Переводит *строку* в формат **application/x-www-form-urlencoded** с использованием специальной схемы кодирования. Этот метод использует данное *кодирование* с целью получения байтов для небезопасных символов. Примечание: Рекомендации Консорциума Всемирной Паутины утверждают, что для кодирования нужно использовать UTF-8. Если этого не делать, может произойти несовместимость.	String

4.7.3 Функции обработки даты/времени

Раздел описывает функции языка выражений, относящиеся к обработке временных меток.

Функция	**Описание**	**Тип результата**
date(Integer year, Integer month, Integer day, Integer hour, Integer minute, Integer second [, Integer millisecond [, String timezone]])	Формирует дату по заданным значениям года, месяца, дня, часа, минуты и секунды. Имейте в виду, **что месяц отсчитывается с нуля**, т.е. значение Января равно 0. Временная зона по умолчанию, используемая для построения временных меток - UTC. Пользовательские временные зоны могут задаваться в форме строки, например: • GMT-8 • GMT-08:00 • America/Los_Angeles	Date
dateAdd(Date date, Integer count, String unit [, String timezone])	Добавляет *число* периодов времени, заданное в упомянутых выше *единицах*, к *дате*, и возвращает результат. См. названия единиц выше. Временная зона по умолчанию - UTC. Пользовательские временные зоны могут задаваться в форме строки, например: • GMT-8 • GMT-08:00 • America/Los_Angeles	Date
dateDiff(Date first, Date second, String unit)	Рассчитывает время, прошедшее с первой даты до второй, измеряемое в: • **millisecond** или **ms** для миллисекунд • **second**, **sec** или **s** для Секунд • **minute**, **min** или **m** для Минут • **hour**, **hr** или **h** для Часов • **day** или **d** для Дней • **week** или **w** для Недель • **month** для Месяцев (Имейте в виду, что месяц с нулевой базой, т.е. значение для Января равно 0.) • **year** или **y** для Лет.	Long
day(Date date [, String timezone])	Возвращает день заданной временной метки. См. описание функции **formatDate**, чтобы узнать, как установить определенную временную зону.	Integer

dayOfWeek(Date date [, String timezone])	Возвращает день недели заданной временной метки (**1 для воскресенья**). См. описание функции **formatDate**, чтобы узнать, как установить определенную временную зону.	Integer
dayOfYear(Date date [, String timezone])	Возвращает день года заданной временной метки. См. описание функции **formatDate**, чтобы узнать, как установить определенную временную зону.	Integer
formatDate(Date date, String pattern [, String timezone])	Форматирует значение Даты в Строку. Образцы форматирования даты описаны здесь 2146. Временная зона по умолчанию, используемая для форматирования временных меток - UTC. Пользовательские временные зоны могут задаваться в форме строки, например: • GMT-8 • GMT-08:00 • America/Los_Angeles Для форматирования даты в строку, подходящую для перестройки Таблицы Данных из списка параметров строки 60, используйте следующий образец: `yyyy-MM-dd HH:mm:ss.SSS`.	String
hour(Date date [, String timezone])	Возвращает час заданной временной метки. См. описание функции **formatDate**, чтобы узнать, как установить определенную временную зону.	Integer
millisecond(Date date)	Возвращает миллисекунду заданной временной метки.	Integer
minute(Date date [, String timezone])	Возвращает минуту заданной временной метки. См. описание функции **formatDate**, чтобы узнать, как установить определенную временную зону.	Integer
month(Date date [, String timezone])	Возвращает месяц заданной временной метки (**0 для Января**). См. описание функции **formatDate**, чтобы узнать, как установить определенную временную зону.	Integer
now()	Возвращает текущую дату/время.	Date
second(Date date)	Возвращает секунду заданной временной метки.	Integer
parseDate(String source, String pattern [, String timezone])	Разбирает значение Даты из Строки. Шаблоны дат описаны здесь 2146. Временная зона по умолчанию, используемая для построения временных меток - UTC. Пользовательские временные зоны могут задаваться в форме строки, например: • GMT-8 • GMT-08:00 • America/Los_Angeles	String
printPeriod(Long period [, minUnit [, maxUnit]])	Интеллектуально форматирует временной *период* (выраженный в миллисекундах) в строку в следующей форме: `X days X hours X minutes X seconds`. *minUnit* и *maxUnit* - это минимальные и максимальные типы целых единиц для использования в отрисовке и редактировании периода. Доступные единицы описаны здесь 2160, единицы Неделя и Квартал не поддерживаются.	String
time(Date date)	Возвращает число миллисекунд с Начала отчета для заданной временной точки.	Long
year(Date date [, String timezone])	Возвращает год заданной временной метки. См. описание функции **formatDate**, чтобы узнать, как установить определенную временную зону.	Integer

4.7.4 Функции обработки таблиц данных

Раздел описывает функции языка выражений, относящиеся к обработке таблиц данных 49.

Функция	Описание	Тип результата
addColumns(DataTable table, String format1, String expression1, String format2, String description2, ...)	Добавляет один и более столбцов к таблице table. Формат первого добавляемого столбца определен аргументом *format1*, его значение задается через *expression1*, и т.д. Формат закодирован в строку, согласно разделу кодирование формата 124 (используются видимые разделители). Значение для данного поля рассчитывается путем выполнения выражения, которое может содержать ссылки 117 на другие ячейки данной таблицы. Если *ряд* не определен в ссылке, она возвращается к текущему ряду (т.е. к тому, для которого рассчитывается значение нового поля). Пример: `addColumns({.:hrStorageTable}, "<usage><S>", "{hrStorageUsed} * 100 / {hrStorageSize} + ' %'")` вернет копию исходной таблицы с одним новым столбцом под названием *usage* типа Строка. Значения для данного столбца рассчитываются при помощи выражения `{hrStorageUsed} * 100 / {hrStorageSize} + ' %'`, которое ссылается на два других столбца той же таблицы.	DataTable
addRecords(DataTable table, Object field1, Object field2, ...)	Добавляет к таблице одну или более записей. Если заданная *таблица* имеет N столбцов, будут использоваться первые N аргументов *полей* для заполнения первой добавленной записи, вторые N аргументов заполнят вторую и так далее. Если для заполнения всей новой записи недостаточно аргументов *поля*, оставшиеся столбцы будут выставлены по умолчанию.	DataTable
adjustRecordLimits(DataTable table, Integer minRecords, Integer maxRecords)	Настраивает минимальное и максимальное количеств записей для *таблицы* и возвращает модифицированную таблицу. Дополнительные пустые записи добавляются, если в *таблице* меньше, чем *minRecords* записей. Конечные записи удаляются, если в *таблице* больше, чем *maxRecords* записей.	DataTable
aggregate(Object source, String expression, Object initialValue)	Рассчитывает некторое "объединенное значение" для Таблицы данных (если *исходный* аргумент является Таблицей данных) или количество контекстов (если *исходный* аргумент является Строкой, представляющей маску контекста). Если первый аргумент - Таблица данных, она проходит через каждую ее запись и рассчитывает *выражение; исходная* Таблица данных, таким образом, становится таблицей по умолчанию 120, а текущий ряд - рядом по умолчанию 119. Если *источником* является маска контекста, *выражение* рассчитывается для каждого контекста, соответствующего маске, он становится контекстом по умолчанию 119. Выражение обычно оперирует двумя значениями: зачением предыдущего вычисления, ссылка на которое осуществляется с помощью `{env/previous}` (см. Ссылки среды 124) и значением(ями) из текущего контекста/записи, которые доступны через стандартные ссылки 118. Во время первого вычисления, `{env/previous}` возвращает значение аргумента *initialValue*. Пример 1: `aggregate("users.*.devices.*", "{env/previous} + ({.:status$connectionStatus} == 1 ? 1 : 0)", 0)` Данное выражение (`{.:status$connectionStatus} == 1`) рассчитает количество устройств онлайн в системе, пройдя через все контексты устройств (маска `users.*.devices.*`) и добавляя 1 к "объединенному" значению, если устройство в сети. Начальное значение равно нулю. Пример 2: `aggregate({}, '{env/previous} || ({prtCoverStatus} == "open")', false)` В данном случае, результат рассчитывается путем прохождения через все записи таблицы данных по умолчанию (возвращенной ссылкой `{}`). Начальное "объединенное" значение является `false`. Конечный результат будет `true`, если поле	Object

	`prtCoverStatus` равно `open` хотя бы в одной записи, и `false` в противном случае. Пример 3: `aggregate({}, max({env/previous}, {field}), 0)` Вышеуказанное выражение вернет максимальное значение, обнаруженное в столбце `field`. Пример 4: `aggregate({}, '({env/previous} * {#row} + {field}) / ({#row} + 1)', 0)` Вышеуказанное выражение рассчитает среднее значение столбца `field`.	
cell(DataTable table, String field [, Integer row [, Boolean description]])	Возвращает значение ячейки Таблицы данных, проходящей в первом аргументе. Ячейка задана параметрами *field* и *row*. Если ряд не задан, возвращается значение *field* в первом ряду. Если *field* является числом, оно рассматривается как индекс поля вместо его имени. Если параметр *description* задан и является истиной, функция возвращает описание значения ячейки вместо самого значения (только если формат поля определяет значения выборки 49). Если *field* не задан, эта функция вернет содержимое первой ячейки таблицы (например, из первого поля и первой строки). Пример: `cell({users:list()}, "firstname", 5)` вернет поле *firstname* из 6-го ряда (индекс = 5) таблицы, содержащей список учетных записей пользователей (полученный с помощью контекстной функции 70 list 1608() из контекста Пользователи 1604).	Object
clear(DataTable table)	Удаляет все записи из *таблицы*. Если минимальное количество записей, разрешенных форматом таблицы, больше нуля, удаляет записи, начиная с конца таблицы, до достижения минимального количества записей.	DataTable
convert(DataTable table, String format)	Конвертирует *таблицу* в формат, определяемый аргументом *format. Формат* кодируется в строку согласно разделу кодирование формата 2124 (используются видимые разделители). После создания новой пустой таблицы заданного формата, функция оптимальным образом копирует все данные из таблицы в новую при помощи операции интеллектуальное копирование Таблицы данных. Пример: выражение `convert({}, '<<f1><S><A=123>><f2><B>>')` создаст таблицу со строковым полем f1 и полем Boolean f2 и скопирует все данные из таблицы данных 120 в только что созданную. Если в таблице по умолчанию поле f1 является целочисленным, его тип будет конвертирован, но все данные сохранятся.	DataTable
copy(DataTable source, DataTable target)	Копирует максимально возможное количество данных из *исходной* таблицы в *целевую* таблицу и возвращает целевую таблицу. Добавляет/удаляет записи из целевой таблицы для соответствия количеству записей в исходной таблице (если это позволяет формат целевой таблицы). При необходимости конвертирует типы значений.	DataTable
decode(String stringToDecode)	Декодирует таблицу данных 49, используя значение *stringToDecode,* которая была кодирована согласно правилам кодирования таблиц данных 2123. Функция автоматически определяет режим кодирования таблицы (с использованием как видимых, так и невидимых разделителей).	DataTable
describe(DataTable table, String field1, String description1, String field2, String description2, ...)	Изменяет описание поля *field1* на *description1* и т.д. Возвращает таблицу. Пример: `describe({.:ifTable}, "ifDescr", "Name", "ifAdminStatus", "Status", "ifType", "Type", "ifSpeed", "Speed")` возвращает копию исходной таблицы, где описания полей *ifDescr*, *ifAdminStatus*, *ifType*, и *ifSpeed* изменяются на *Name*, *Status*, *Type* и *Speed* соответственно.	DataTable

description(DataTable table)	Возвращает описание *таблицы*, т.е. результат вычисления ее выражения названия 49.	String
distinct(DataTable table)	Удаляет дубли записей из *таблицы* и возвращает отфильтрованную таблицу.	DataTable
encode(DataTable table [, boolean useVisibleSeparators])	Кодирует *таблицу* в строку в соответствии с правилами кодирования таблицы данных 2123.	String
filter(DataTable table, String filterExpression)	Возвращает таблицу, содержащую только те строки исходной *таблицы,* которые соответствуют *filterExpression.* Пример: `filter({users:list()}, "contains({firstname}, 'John')")` вернет только тех пользователей, чьи имена соедржат строку `John`.	DataTable
getFormat(DataTable table [, Boolean useVisibleSeparators])	Возвращает формат *таблицы* в виде Строки. Видимость разделителей определяется параметром *useVisibleSeparators*. Если этот параметр не задан, используются невидимые разделители.	String
hasField(DataTable table, String field)	Возвращает true, если *таблица* имеет поле под названием *поле.* В противном случае возвращает false.	Boolean
intersect(DataTable sourceTable, String fieldInSourceTable, DataTable sampleTable, String fieldInSampleTable [, Boolean filterType])	Возвращает таблицу, содержащую те же строки, что и *sourceTable,* за исключением тех, которые были найдены в *sampleTable*, если *filterType* true. Возвращает таблицу, содержащую те же строки, что и *sourceTable,* за исключением тех, которые не были найдены в *sampleTable*, если *filterType* false или не задан. Сравнивает записи в *sourceTable* и *sampleTable* по значениям полей *fieldInSourceTable* и *fieldInSampleTable*.	DataTable
join(DataTable left, DataTable right)	Объединяет две таблицы горизонтально. Длина результирующей таблицы будет равняться максимальной длине *левой* и *правой* таблиц. Первые столбцы результирующей таблицы будут взяты из *левой* таблицы, а последующие столбцы будут взяты из *правой* таблицы. Если в *левой* таблице есть ключевые поля, функция join сопоставит записи из правой таблицы, используя эти ключевые поля.	DataTable
print(DataTable table, String expression, String separator)	Рассчитывает *выражение* для каждой записи *таблицы* и добавляет выход в строку, отделяя результаты друг от друга при помощи *разделителей*. Пример: `print({users:list()}, "{firstname}", ", ")` вернет имена пользователей, например, `Amanda, Michelle, John, Donald, Paul`	String
records(DataTable table)	Возвращает количество записей (строк) в *таблице*.	Integer
removeColumns(DataTable table, String column1, String column2, ...)	Удаляет несколько столбцов из *таблицы*.	DataTable
removeRecords(DataTable table, String fieldToCheck, Object value)	Удаляет заданные записи из таблицы, т.е. возвращает новую таблицу, содержащую все записи из исходной, кроме тех, которые имеют *fieldToCheck* равную *значению*.	DataTable
select(DataTable table, String fieldToSelect, String fieldToCheck, Object value)	Сканирует каждый ряд в таблице и проверяет, равняется ли значение поля *fieldToCheck значению.* Если оно true, возвращает значение *fieldToSelect* из текущей записи. Если в *таблице* не обнаружено совпадающих записей, данная функция вернет NULL. Пример: `select({users:list()}, "firstname", "username", "john")` вернет имя пользователя со значением поля username *john,* выбранным из таблицы статистики пользователей (см. пример выше).	Object
set(DataTable table, String field, Integer row, Object value)	Изменяет значение ячейки *таблицы*, указанное *полем* и *строкой,* на *значение* и возвращает измененную таблицу.	DataTable
setMultiple(DataTable table, String field, Object value [, String	Сканирует каждую строку таблицы и задает *значение* на *поле*, если выражение *условия* - true. Устанавливает *значение* на *поле*	DataTable

condition])	для всех строк в таблице, если *условие* отсутствует.	
sort(DataTable table, String field, boolean ascending)	Сортирует записи *таблицы* в соответствии с естественным порядком значений *поля*. Значение флажка *возрастание* определяет порядок сортировки.	DataTable
subtable(DataTable table, String field1, String field2, ...)	Принимает *таблицу* в качестве аргумента и возвращает другую таблицу, содержащую только те поля, имена которых определяются аргументом *fields.* Формат и содержание сохраняются в возвращаемой субтаблице. Пример: `subtable({users:list()}, "firstname", "lastname")` вернет таблицу с двумя столбцами, содержащими имя и фамилию всех пользователей AtomMind Server.	DataTable
table(String format, Object field1, Object field2, ...)	Создает новую таблицу, используя *формат* закодированный в строку 2124 и массив параметров. Правила, используемые для заполнения таблицы данными, описаны здесь 60. Функция автоматически определяет режим кодирования таблицы 2130, принимая таблицы, закодированные с использованием как видимых, так и невидимых разделителей. Пример: `table("<<from><I>><<to><I>>", 2, 5, 3, 7)` вернет таблицу с двумя столбцами с Целыми числами и две строки. Первая строка будет иметь поле `from`, равное 2, и поле `to`, равное 5. Значение второй строки будет `from`=3 и `to`=7. Если *формат* не задан, эта функция создаст таблицу с пустым форматом (например, без столбцов). Затем эту таблицу можно расширить, используя другие функции обработки таблиц.	DataTable
union(DataTable left, DataTable right)	Объединяет две таблицы вертикально. Формат результирующей таблицы будет состоять из форматов двух таблиц ввода. Первые записи результирующей таблицы будут взяты из *левой* таблицы, а последующие записи будут взяты из *правой* таблицы.	DataTable
validate(DataTable table [, Boolean throwErrors])	Выполняет валидаторы таблицы для проверки правильности данных в таблице. Если параметр *throwErrors* false или отсутствует, возвращает null в случае успеха и сообщение об ошибке при сбое валидации. Если параметр *throwErrors* true, возвращает саму таблицу. В этом случае все ошибки выдаются как исключения фукнции.	Object

4.7.5 Функции обработки цвета

Раздел описывает функции языка выражений, относящиеся к обработке зачений типа "цвет".

Функция	**Описание**	**Тип результата**
blue(Color color)	Возвращает синий компонент цвета (0...255).	Integer
brighter(Color color)	Добавляет дополнительный коэффициент к каждому из трех RGB компонентов аргумента color для создания более яркой версии цвета.	Color
color(Integer red, Integer green, Integer blue)	Формирует цвет тремя RGB значениями. Каждый компонент должен быть в диапазоне от 0 до 255.	Color
darker(Color color)	Добавляет дополнительный коэффициент к каждому из трех RGB компонентов аргумента color для создания более темной версии данного цвета.	Color
green(Color color)	Возвращает зеленый компонент цвета (0...255).	Integer
red(Color color)	Возвращает красный компонент цвета (0...255).	Integer

4.7.6 Функции, относящиеся к контексту

Раздел описывает функции языка выражений, относящиеся к извлечению данных из контекстов[41] и работе с ними.

Функция	Описание	Тип результата
available(String context [, String schema])	Возвращает true, если контекст с путем *контекст* существует и доступен вызывающему.	Boolean
callFunction(String context, String function, Object parameter1, Object parameter2, ...)	Вызывает функцию[70] под названием *функция* контекста с путем *контекст* и возвращает ее выход. Ввод функции таблица данных[49] строится из массива параметров (*parameter1*, *parameter2*, ...). Правила, используемые для заполнения таблицы данными, описаны здесь[60].	DataTable
dc([String schema])	Возвращает путь контекста по умолчанию[119], определенный во время оценки выражения. Другой способ получения пути контекста по умолчанию - это использование ссылки[118] `{.:}`.	String
dr()	Возвращает строку по умолчанию[119], определенную во время оценки выражения.	Integer
dt()	Возвращает таблицу по умолчанию[120], определенную во время оценки выражения. Другой способ получения пути контекста по умолчанию - это использование ссылки[118] `{}`.	DataTable
eventAvailable(String context, String event [, String schema])	Возвращает true, если контекст с путем, определенный аргументом *контекст,* имеет переменную, определенную аргументом *событие,* и это событие доступно вызывающему.	Boolean
eventFormat(String context, String event [, String schema])	Возвращает строковое представление формата события или ноль, если *контекст*/*событие* недоступен или формат *события* динамический. Использует режим кодирования невидимых разграничителей[2130] для кодирования формата в строку. Для построения формата с использованием формата, возвращенного этой функцией, и списка значений ячеек, используйте функцию `table`.	String
fireEvent(String context, String event, Integer level, Object parameter1, Object parameter2, ...)	Запускает событие под названием *событие* в *контексте*. Использует определенный *уровень* или уровень события по умолчанию, если параметр *уровня* - NULL. Коды уровня события описаны здесь[75]. Если предоставляется лишь один параметр типа Data Table, эта таблица будет использоваться в качестве таблицы данных. В ином случае Таблица данных[49], представляющая данные события, строится из массива параметров (*parameter1*, *parameter2*, ...). Правила, используемые для заполнения таблицы данными, описаны здесь[60]. Эта функция возвращает идентификатор сгенерированного события или ноль, если генерация событий блокируется системой. Функция fireEvent() сработает лишь в выражении, которое оценивается в AtomMind Server. Выражения, которые оцениваются в AtomMind Client или Agent, не вызовут генерирование событий сервера.	Long
fullDescription(String context [, String delimiter])	Возвращает полное описание определенного контекста, т.е. описания всех родительских контекстов, разделенных строкой-разделителем. Строка-разделитель по умолчанию "-".	String
functionAvailable(String context, String function [, String schema])	Возвращает true, если контекст с путем, указанным аргументом *контекст*, имеет функцию, определенную аргументом *функция*, и эта функция доступна для вызывающего.	Boolean

functionInputFormat(String context, String function [, String schema])	Возвращает строковое представление формата ввода функции или ноль, если *контекст*/*функция* недоступны или формат ввода *функции* динамический. Использует режим кодирования невидимых разграничителей 2130 для кодирования формата в строку. Для построения формата с использованием формата, возвращенного этой функцией, и списка значений ячеек, используйте функцию table.	String
functionOutputFormat(String context, String function [, String schema])	Возвращает строковое представление формата вывода функции или ноль, если *контекст*/*функция* недоступны или формат вывода *функции* динамический. Использует режим кодирования невидимых разграничителей 2130 для кодирования формата в строку. Для построения формата с использованием формата, возвращенного этой функцией, и списка значений ячеек, используйте функцию table.	String
getVariable(String context, String variable)	Получает переменную 61 под названием *переменная* из контекста с путем *контекст* и возвращает ее значение.	DataTable
setVariable(String context, String variable, Object parameter1, Object parameter2, ...)	Устанавливает переменную 61 под названием *переменная* контекста с путем *контекста*. Если обеспечивается лишь один параметр типа Data Table, эта таблица будет использоваться в качестве нового значения переменной. В ином случае Таблица данных 49, представляющая новое значение переменной, строится из массива параметров (*parameter1*, *parameter2*, ...). Правила, используемые для заполнения таблицы данными, описаны здесь 60.	Null
setVariableField(String context, String variable, String field, Integer record, Object value)	Устанавливает значение ячейки указанной переменной 61 *контекста*. Ячейка задается параметрами *поле* и *запись*.	Null
setVariableRecord(String context, String variable, Integer record, Object parameter1, Object parameter2, ...)	Изменяет значения определенного ряда переменной 61 *контекста*. Номер ряда задается параметром *запись*. Другие параметры используются для изменения значений этого ряда.	Null
variableAvailable(String context, String variable [, String schema])	Возвращает true, если контекст с путем, определенный аргументом *контекст,* имеет переменную, определенную аргументом *переменная,* и эта переменная доступна вызывающему.	Boolean
variableFormat(String context, String variable [, String schema])	Возвращает строковое представление формата переменной или ноль, если *контекст*/*переменная* недоступны или формат *переменной* динамический. Использует режим кодирования невидимых разграничителей 2130 для кодирования формата в строку. Для построения формата с использованием формата, возвращенного этой функцией, и списка значений ячеек, используйте функцию table.	String
variableReadable(String context, String variable [, String schema])	Возвращает true, если контекст с путем, определенный аргументом *контекст,* имеет переменную, определенную аргументом *переменная,* и эта переменная читается вызывающим.	Boolean
variableWritable(String context, String variable [, String schema])	Возвращает true, если контекст с путем, определенный аргументом *контекст,* имеет переменную, определенную аргументом *переменная,* и эта переменная записывается вызывающим.	Boolean

4.7.7 Функции конвертации типов

Раздел описывает функции языка выражений, относящиеся к ручной конвертации типов значений.

Функция	**Описание**	**Тип результата**
boolean(Object value)	Конвертирует аргумент в Boolean.	Boolean
double(Object value)	Конвертирует аргумент в Double.	Double
integer(Object value [, Integer radix])	Конвертирует аргумент в Integer.	Integer

	Если *основание* указано, эта функция преобразует значение в строку и анализирует его как целое число в этой системе счисления. В этом случае все символы в строке должны быть цифрами указанной системы счисления, за исключением, что первый символ может быть в формате ASCII знак минус ("-"), чтобы указать отрицательное значение.	
float(Object value)	Конвертирует аргумент в Float.	Float
long(Object value [, Integer radix])	Конвертирует аргумент в Long. Если *основание* указано, эта функция преобразует значение в строку и анализирует его как целое число в этой системе счисления. В этом случае все символы в строке должны быть цифрами указанной системы счисления, за исключением, что первый символ может быть в формате ASCII знак минус ("-"), чтобы указать отрицательное значение.	Long
string(Object value)	Конвертирует аргумент в String.	String
timestamp(Object value)	Конвертирует аргумент в Date. Аргумент *значение* первым конвертируется в число Integer. Затем число интерпретируется как временная метка Unix, т.е. количество миллисекунд с Эпохи.	Date
dataBlock(Object value [, String charset])	Преобразует аргумент в строку для использования в Данных. Если определен *набор символов*, данная функция кодирует значение в строку используя данную *кодировку*.	String

4.7.8 Другие функции

Раздел описывает функции языка выражений, которые не могут быть включены в другие разделы.

Функция	**Описание**	**Тип результата**
catch(Object normalResult [, Object errorResult])	Функция рассчитывает аргумент *normalResult* и возвращает его, если оценка прошла успешно. Однако если во время расчета *normalResult* произошла ошибка, поведение отличается: • Если *errorResult* не определен, возвращается текст расчета ошибки *normalResult* • Если *errorResult* определен, то он сам и возвращается	Object
evaluate(String expression [, String defaultContext [, DataTable defaultTable [, Integer defaultRow]]])	Обрабатывает строковый аргумент как выражение AtomMind 112 и возвращает результат. Использует кастомные *defaultContext*, *defaultTable* и *defaultRow*, если есть.	Object
fetchDataBlock(DataBlock data)	Возвращает данные кодированными байтами бинарного блока, извлеченными из базы данных.	DataBlock
ld(String variable)	Возвращает значение локальной переменной среды под названием *переменная*. Эта переменная может определяться лишь вызовом функции `st()`. Внешние (нелокальные) переменные среды возвращаться не будут, на них можно сослаться 123 через `{env/variable}`.	Object
login()	Возвращает имя логина текущего пользователя (например, `john@company.com`). В отличие от функции user(), эта функция возвращает имя пользователя, которое было использовано при внешней аутентификации (например, через Active Directory или LDAP).	String

	Если внешняя аутентификация не применялась, эта функция возвращает значение пользователя AtomMind Server, т.е. работает так же, как функция user()	
script(String name, Object parameter1, Object parameter2, ...)	Вызывает скрипт виджета 1308 под названием *имя* и передает ему несколько параметров. Возвращает объект вывода скрипта.  Эта функция доступна только в выражениях привязки виджета 1296.	Object
sleep(Integer milliseconds)	Приостанавливает процесс обработки выражения на определенное количество миллисекунд.	Null
st(String variable, Object value)	Определяет или обновляет локальную переменную среды под названием *переменная* путем ее установки на *значение*. Также возвращает *значение*. Эта функция полезна для повторного использования определенных частей выражения с целью оптимизации производительности оценки и ее большей компактности. Пример: `st('var', {a} * {b} * {c} * {d} * {e}) > 50 ? ld('var') : -1` Это выражение возвращает умножение полей a, b, c, d и e, если они больше, чем 50, в другом случае -1. Однако умножение рассчитывается лишь один раз и хранится в локальной переменной среды под названием var.	Object
tableFromCSV(String csv, String header, String delimiter [, String format])	Функция принимает документ CSV как входные данные и преобразует его в таблицу данных 49. Параметр *разделитель* определяет символ, который используется в качестве разделителя столбцов. Строковое значение из одного символа. Формат таблицы может быть определен с помощью параметра *формат*, который должен быть корректным форматом таблицы данных, кодированной в строку 2124. Количество столбцов, определенных *форматом*, должно соответствовать количеству столбцов в файле CSV. Параметр *заголовок* определяет содержание первой строки в файле CSV. Значение параметра зависит от того, определен *формат* или нет. Если *формат* определен, *заголовок* может быть следующим: • **none** Файл CSV не имеет заголовка, первая строка содержит значения. Данные из первой строки файла CSV импортируются как первая запись таблицы данных и так далее. • **skip** Первая строка CSV файла пропускается (считается незначимыми данными). Если *формат* не определен, первая строка CSV файла должна содержать имена или описания полей таблицы данных. В таком случае *заголовок* может быть следующим:	DataTable

	• **names** Имена полей таблицы данных импортируются из первой строки CSV файла. Если имя оказывается пустым, оно заменяется номером соответствующего столбца. Тип полей таблицы данных установлен на строку. • **descriptions** Описания полей таблицы данных импортируются из первой строки CSV файла. Номера соответствующих столбцов используются как имена полей таблицы данных. Тип полей таблицы данных установлен на строку. Функция использует следующие настройки (описаны в опциях импорта/экспорта в CSV 216): • Использовать ограничитель текста: не использовать. • ставка символа ограничителя текста в текст: использовать обратную косую. • Символ комментария: знак номера (#).	
tableFromJSON(String json [, boolean convertUnequalFieldTypesToString])	Функция принимает JSON документ как входные данные и преобразует его в таблицу данных 49 с единственной записью, в которой каждая пара ключ-значение представлена отдельным полем таблицы. Если указан и активирован параметр *convertUnequalFieldTypesToString,* функция преобразуем значения столбцов в строку, если в этих столбцах имеются значения разных типов.	DataTable
tableToJSON(DataTable table)	Функция принимает таблицу данных 49 как входные данные и преобразует ее в JSON документ. Согласно спецификации, объект JSON - это *неупорядоченное* множество пар имя/значение, поэтому нельзя гарантировать соблюдение определенного порядка элементов в результирующем документе JSON.	String
trace(Object value [, String message])	Записывает *значение,* сопровождаемое *сообщением,* как системное событие для включения отладки выражений 140. Возвращает неизмененное *значение*.	Object
user()	Возвращает имя текущего пользователя (например, `admin` или `john`), т.е. учетную запись пользователя 478, чьи права доступа используются во время оценки выражения. В отличие от функции login(), эта функция возвращает имя локального аккаунта пользователя, а не имя логина, использованного при внешней аутентификации (например, через Active Directory или LDAP).	String
xpath(String xml, String query [, Boolean table])	Принимает XML документ в качестве ввода и вычисляет по нему запрос XPath. Запрос может быть оценен в строку (это поведение по умолчанию) или таблицу данных (если параметр *таблица* верен). Табличный режим следует использовать, если запрос вычисляется в набор узлов, не только содержание текста одного узла или значение атрибута. Таблица, возвращенная в табличном формате, имеет три поля: • **Имя**. Имя узла. • **Контекст**. Содержание узла текста. • **Дочерние узлы**. Таблица дочерних узлов данного узла. Имеет такой же формат.	Object

4.8 Комментарии

Выражения могут включать комментарии в одну и более строк.

Комментарий в одну строку начинается с последовательности символов // и продолжается до конца текущей строки.

Вот пример комментария в одну строку:

```
{temperature} > 10 // Low threshold
&&
{temperature} < 30 // High threshold
```

Комментарий в несколько строк начинается с последовательности символов /* и заканчивается последовательностью */.

Вот пример комментария в несколько строк:

```
{temperature} > 10 /* Low threshold */ && {temperature} < 30 /* High threshold
*/
```

4.9 Примеры выражений

Выражения, используемые в определениях Фильтра событий[762] и Тревог[779], должны получать результат типа **Boolean**. Это позволяет AtomMind Server определить, может ли событие пройти через фильтр или должна ли тревога быть активирована по какому-либо событию.

Выражение	**Результат вычисления**	**Тип результата**
`5 + 5 > 7`	true	Boolean
`3 + 2`	5	Integer
`"Red" + " " + "Line"`	"Red Line"	String
`{speed}>5`	true, если ссылка **speed** разрешается в целое число более *5*.	Boolean
`{name} == "admin"`	true, если ссылка **name** разрешается в строку *“admin”.*	Boolean
`{failed[3]} == false`	true, если ссылка **failed[3]** разрешается в *"false"* (Boolean)	Boolean
`{env/context} ~= "^abc.*"`	true, если ссылка **env/context** разрешается в Строку, совпадающую с регулярным выражением "*^abc.**", т.е. начинается с "*abc*".	Boolean
`{int[1]} == {int[2]}`	true,если обе ссылки разрешаются в значения, похожие друг на друга (т.е. два числа или две строки) и эти значения равны.	Boolean
`({field} + 5) * ({field2} - 1) / 2 == {field[1]} + 1`	true или false, в зависимости от значений разрешенных ссылок, которые должны быть числовыми.	Boolean
`substring("smiles", 1, 5)`	"mile"	String
`max(100, {pressure})`	Значение ссылки **pressure**, если оно больше 100 или равно 100.	Number

Расчет расстояния между точками

Предположим, мы владеем небольшой компанией, сдающей в аренду автомобили, и не хотим, чтобы наши автомобили уезжали далеко от стоянки. В целях управления, каждая машина оборудована небольшим контроллером, который читает положение машины с GPS приемника и отправляет его на сервер AtomMind по

сотовой связи при помощи GPRS. В нашей системе, мы имеем координаты двух точек: автомобиля и стоянки. Мы можем рассчитать расстояние между ними при помощи выражения или создать выражение, которое активирует тревогу 779, когда расстояние превысило заданный предел.

Если наши координаты выражены в радианах, мы можем применить следующую формулу для поднятия тревоги, если расстояние между транспортным средством и стоянкой превысило 500 километров:

```
acos(sin({car_latitude}) * sin({base_latitude}) + cos({car_latitude}) *
cos({base_latitude}) * cos({car_longitude} - {base_longitude})) * 6371  > 500
```

`6371` радиус Земли в километрах

`500` разрешенное максимальное расстояние между автомобилем и стоянкой в километрах

Если координаты выражения в градусах (что более вероятно в случаях данных GPS приемника), мы просто конвертируем градусы в радианы, например, записав

```
{car_latitude} / 180 * 2 * 3.1415926
```

вместо

```
{car_latitude}
```

4.10 Отладка выражений

Во многих случаях выражения вычисляются в автономной среде, что серьезно затрудняет их тестирование. Например, когда системный администратор создает выражение для триггера событий 787 тревоги, он не может корректно проверить его в Редакторе выражений 404, так как у него под рукой нет экземпляров запускающего события, которое могло бы предоставить "реальные" данные. Другими словами, таблица по умолчанию 120 не будет определена в среде вычисления 114, а выражение не будет выдавать правильный результат при расчете в визуальном редакторе.

Использование журнала вычислений и ошибок

Действие Просмотр событий 109 дает доступ к журналу вычислений выражения определенного контекста, а также к журналу ошибок вычисления.

Каждое действие по вычислению выражения регистрируется как отдельное событие Вычисление 85. Неудачные попытки вычислить выражение записываются в журнал как события Ошибка вычисления 86.

Вычисление некоторых выражений не привязано к какому-либо серверному контексту. Для таких выражений описываемый способ отладки работать не будет.

Использование функции трассировки

Язык выражений AtomMind включает в себя очень полезную функцию `trace()`, которая была разработана для отладки выражения. Эта функция берет любое значение, регистрирует его (чтобы его было видно в Журнале событий 398) и возвращает неизменным, имея результатом любые другие побочные эффекты.

Способы просмотра значений, вводимых функцией трассировки, различаются:

- Если выражение вычисляется в AtomMind Server, значения можно просматривать путем активации фильтра событий 762 **Трассировка выражения** (который является частью ядра установочного пакета AtomMind)
- Если выражение вычисляется в виджете, значения можно просматривать в журнале событий виджета 983.
- В других случаях значения будут вводиться в стандартном инструменте журналирования 166 (доступ к нему можно получить из консоли, файла и т.п.)

Пример: Предположим, что мы хотим создать тревогу, которая будет появляться при давлении менее 20 единиц и температуре более 50 единиц. Начальное выражение тригерра переменной 782 будет выглядеть следующим образом:

```
{pressure} < 20 && {temperature} > 50
```

Однако если тревога не появляется, сложно выяснить, характеризует ли выражение со значением false давление или температуру.

Для осуществления легкой отладки мы можем менять данное выражение следующим образом:

```
track({pressure}, "Pressure") < 20 && track({temperature}, "Temperature") > 50
```

После этого мы будем видеть следующий вывод в Журнале событий каждый раз, когда тревога производит проверку:

- `Pressure: 25.3`
- `Temperature: 52.7`

Это позволяет быстро находить текущие значения и видеть, почему настройка тревоги неправильная.

Другой способ - это отследить состояние отдельных частей выражения вместо "сырых" значений метрик:

```
track({pressure} < 20, "Pressure Condition") && track({temperature} > 50,
"Temperature Condition")
```

Вывод будет следующий:

- `Pressure Condition: false`
- `Temperature Condition: true`

4.11 Безопасность выражений

Обработка выражения всегда наследует права доступа 477 вызывающей стороны. Например, если триггер тревоги 779 пытается выполнить запрос 829 по группе устройств 497, выражение триггера будет использовать права доступа хозяина тревоги и передаст их в функцию **Выполнить запрос**, позволяя получить доступ только к тем устройствам, которые ему доступны.

Права доступа вызывающей стороны используются для:

- Преобразования ссылок 118 в переменные и функции контекста
- Выполнения функций контекста 124 самого языка выражений

4.12 Производительность выражений

Поскольку AtomMind во многом основан на выражениях, расчет влияния их производительности критичен для планирования производительности всей инсталляции AtomMind.

Влияние загрузки процессора

Некоторые общие замечания по производительности выражения:

- Если выражение ссылается на переменные, функции или контексты, расположенные на локальном сервере, время его расчета будет зависеть только от времени, необходимого локальному серверу для завершения операции. Оно может отличаться в тысячу раз в зависимости от алгоритма чтения переменной и кода реализации функции. Чтение локальной переменной контекста или вызов локальной функции контекста также могут вызвать значительную нагрузку процессора на локальном сервере.
- Если выражение ссылается на переменные 61 или функции 70 удаленных контекстов 41 (например, контекстов на удаленных серверах), его расчет занимает значительное время и может вызывать нагрузку процессора удаленного сервера. Время, необходимое для расчерта выражения, включающего ссылку 117 на удаленный контекст, является суммой:
  - Кругового времени сети, необходимого для отправки команды (т.е. получения/установки переменной или команды вызова функции) на удаленный сервер и получения ответа (обычно приравнивается результату команды `ping`)
  - Времени, необходимого удаленному серверу для обработки команды.
- Во всех других случаях обработка выражения происходит очень быстро и не имеет значительного влияния на нагрузку процессора. Миллионы подсчетов в секунду легко осуществляются любым сервером.

Следующая таблица помогает оценить влияние различных элементов выражения:

Элемент	Производительность

Основные операции (арифметические, сравнительные, логические, условные и т.д.)	Очень высокая. Влияние производительности (загрузки процессора) можно заметить, если выражение рассчитывается миллионы раз в секунду.
Ссылки на таблицу по умолчанию 120 (например, `{field}` или `{temperature[2]}`)	Очень высокая, миллионы операций в секунду не вызовут снижения производительности.
Ссылки контекста (например, `{.:temperature$celsius}` или `{:executeQuery("SELECT * from users.*:info")}`)	Очень высокая или низкая в зависимости от расположения контекста (локального или удаленного) и типа переменной/функции контекста. Производительность разрешения ссылок переменных/функций сервера зависит от: • Времени, необходимого для отправки соответствующего запроса на сервер и получения ответа. Это не относится к выражениям, рассчитываемым внутри сервера (например, в привязках моделей 814), но относится к удаленному доступу (например, для выражений привязки виджета 1295) • Времени и ресурсов (время загрузки процессора, память, ввода/вывод базы данных), необходимых для получения значения вывода данной переменной или функции. Для получения более подробной информации см. производительность переменных 70 и производительность функций 73.
Ссылки на компоненты виджета (например, `{form/textField1:text}`)	Высокая, десятки тысяч операций в секунду не вызовут снижения производительности.
Математические функции	Очень высокая, миллионы операций в секунду не вызовут снижения производительности.
Функции обработки строки	Очень высокая или высокая. Влияние на производительность заметно только при выполнении операций с очень длинными строками (миллионы символов).
Функции обработки даты/времени	Очень высокая, миллионы операций в секунду не вызовут снижения производительности.
Функции обработки цвета	Очень высокая, миллионы операций в секунду не вызовут снижения производительности.
Функции обработки таблицы данных	Очень высокая или средняя, только операции с очень большими таблицами (тысячи записей) могут оказывать значительное влияние на загрузку процессора.
Функции конвертации типов	Очень высокая, миллионы операций в секунду не вызовут снижения производительности.
Функции контекста	Очень высокая или низкая, в зависимости от расположения контекста (локального или удаленного) и типа переменной/функции контекста. Для получения более подробной информации см. производительность переменных 70 и производительность функций 73.

Влияние использования памяти

Выражения не вызывают постоянное использование памяти. Однако расчет выражения может вызвать значительную загрузку памяти, если она относится к некоторым переменным или функциям контекста, чей базовый код "реализации" загружает или генерирует много временных данных (например, загрузку длинного списка исторических событий 75 из базы данных сервера или выполнение запроса 829 по десяткам тысячам устройств).

ВЫражения в распределенной среде

Производительность вычисления выражения в распределенной среде может сильно зависеть от структуры выражения, поэтому архитекторы системы должны обращать внимание на возможность ухудшения производительности. В качестве примера проанализируем выражение, которое используется как часть привязки виджета 1296:

```
addColumns({users.admin.devices.userProvider:userTable}, "<attachedValue><S>",
"callFunction('users.admin.devices.userProvider', 'getUserInformation',
{columnOfUserTable}")
```

Обычно это выражение вычисляется на сетевом хосте, где запущен AtomMind Client или другой пользовательский интерфейс AtomMind Server. Число сетевых запросов, необходимых для вычисления выражения, будет равняться `1 + Number_of_records_in_userTable`, так как отдельный удаленный запрос потребуется для каждого вызова функции `getUserInformation()`. Таким образом, если сетевая задержка (время ответа) составляет 1 секунду, а `userTable` содержит 1000 записей, вычисление выражения займет примерно 1000 секунд, что, скорее всего, недопустимо.

Для решения такой проблемы необходимо "перенести" точку вычисления выражения на другую сторону соединения, т.е. на сторону AtomMind Server. Сделать это можно многими способами. Например, на сервере можно создать отдельную модель 810 **User Provider** с функцией 818 `getExtendedUserTable()` типа **Выражение**. Приведенное выше выражение должно использоваться в параметрах данной функции, а само выражение привязки виджета будет лишь ссылаться на эту функцию, например: `{users.admin.models.userProvider:getExtendedUserTable()}`. В этом случае модель на стороне сервера будет осуществлять вычисление без сетевых вызовов для каждой записи, а выражение привязки будет делать лишь один удаленный вызов для получения предварительно созданной таблицы с сервера. Время вычисления выражения сократится примерно до 1 секунды, т.е. в 1000 раз.

5 Развертывание и администрирование

Эта глава описывает внедрение, конфигурацию и администрирование AtomMind Server . AtomMind Server - межплатформный продукт, разработанный на базе Java и оптимизированный для легкой и быстрой установки.

Дополнительную информацию можно найти в Установка AtomMind Server 152 и Настройка AtomMind Server 177.

5.1 AtomMind Server

В этой главе говорится о AtomMind Server , основном компоненте AtomMind, который отвечает за маршрутизацию и сбор данных, обработку бизнес-правил и т.п.

AtomMind - сложная система, содержащая множество модулей для обработки данных. Взаимодействие между этими модулями, то, как они используются вместе, наделяет AtomMind мощными и широкими возможностями применения. Углубляя свои знания о каждом из этих модулей, Вы сможете сформировать для себя всестороннее понимание системы и придумать решения для любого сценария.

5.2 Лицензирование

AtomMind использует именные лицензии, которые привязаны к серверу или виртуальной машине, на которой запущен AtomMind Server. Если вы приобрели лицензию для AtomMind , Вам необходимо сообщить ваш *Активационный Ключ AtomMind Server* в ТВЭЛ, чтобы мы могли активировать Вашу именную лицензию для успешного начала установки AtomMind.

Ключ активации - это текстовая строка, которая служит уникальным идентификатором вашего экземпляра AtomMind Server.

Так как лицензия AtomMind привязана к определенной инсталляции AtomMind Server, прежде всего необходимо выполнить промышленную установку AtomMind Server. Это означает, что AtomMind Server должен быть установлен на том же ПК, где он будет использоваться при промышленной эксплуатации. Подробнее об этом см. в разделе установка AtomMind Server 152.

Активировать лицензию можно и на уже установленной пробной версии AtomMind. Нет необходимости переустанавливать систему заново.

Получение Активационного Ключа

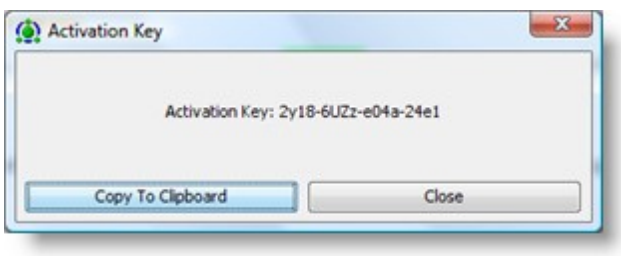
Существует два способа для получения активационного ключа:

- Использование меню панели задач после запуска AtomMind Server
- Использование файла **activation.txt**

ПОЛУЧЕНИЕ АКТИВАЦИОННОГО КЛЮЧА ПРИ ПОМОЩИ МЕНЮ ПАНЕЛИ ЗАДАЧ

Данный метод следует использовать в большинстве случаев, кроме тех, когда AtomMind Server не может быть успешно запущен (например, по истечении срока действия лицензии), или когда программное обеспечение запущено в автономной среде (например, на базе Linux).

1. Запустите 158 AtomMind Server.
2. Кликните правой кнопкой мыши на иконку меню в системной трее 173 и выберите пункт **Просмотреть Активационный Ключ**. Появится диалоговое окно **Активационный Ключ**:



3. Нажмите **Копировать в Буфер Обмена.**

4. Вставьте активационный ключ в текст e-mail сообщения и отправьте его в ТВЭЛ.

ПОЛУЧЕНИЕ АКТИВАЦИОННОГО КЛЮЧА ИЗ ФАЙЛА

При запуске AtomMind Server, программа записывает свой Активационный Ключ в файл **activation.txt,** который находится в установочном каталоге AtomMind Server. Это происходит даже если AtomMind Server не может быть успешно запущен из-за истекшего срока действия лицензии или по каким-либо другим причинам.

Для получения Активационного Ключа запустите AtomMind Server хотя бы один раз, а затем откройте **activation.txt** в любом текстовом редакторе.

Процедура Активации Лицензии

Процедура активации вашей лицензии AtomMind очень проста. Как только вы купили коммерческую лицензию и сообщили в ТВЭЛ Активационный Ключ, мы отправляем вам файл с Именной Лицензией по e-mail. Название файла - **server.license**.

Для активации Лицензии просто поместите данный файл в установочный каталог вашего AtomMind Server (замените существующий файл с Пробной Лицензией) и перезапустите AtomMind Server.

Диспетчер файлов, используемый для копирования файла лицензии под управлением Windows, нужно запускать в режиме с повышенными правами (например, с правами администратора). В случае несоблюдения данного правила файл лицензии может быть записан в ошибочную папку из-за функции "Виртуализация Системы Файла и Регистрации" в последних версиях Windows. В этом случае AtomMind Server не сможет получить доступ к новой лицензии и попытается использовать старую, Истёкшую Лицензию, в результате чего не сможет запуститься или выдаст иные ошибки.

Чтобы открыть Windows Проводник в режиме с повышенными правами, найдите Проводник в меню "Пуск", кликните по нему правой кнопкой мыши и выберите "Запустить на правах Администратора".

СЕРвер лицензий

Большие корпоративные клиенты со множеством развернутых серверов AtomMind могут использовать централизованное лицензирование через сервер лицензий 145.

5.2.1 Сервер лицензий

Большие корпоративные клиенты со множеством развернутых серверов AtomMind могут воспользоваться централизованным управлением лицензиями через специальный *Сервер лицензий*. В этом случае, каждый AtomMind Server подключается при запуске к Серверу лицензий заказчика и получает действующую лицензию вместо того, чтобы загружать ее с локального диска.

КОНФИГУРИРОВАНИЕ СЕРВЕРА ЛИЦЕНЗИЙ

Технически, Сервер лицензий - это обычный AtomMind Server, на котором запущен плагин 207 Сервер лицензий. Этот плагин можно получить в публичном магазине приложений 132 АО "ТВЭЛ".

Общие настройки плагина 207 Сервер лицензий включает таблицу **Лицензии**, которая позволяет настраивать лицензии для использования каждым AtomMind Server в управляемой группе. Эта таблица определяет:

- **Ключ активации** сервера
- **Файл лицензии** для сервера
- **Подробности** о лицензии в Файле лицензии (Владелец лицензии, версия, ограничения и пр.)

АКТИВАЦИЯ ИСПОЛЬЗОВАНИЯ СЕРВЕРА ЛИЦЕНЗИЙ

См. главу Сервер лицензий 195 в разделе **Параметры общих настроек,** чтобы узнать, как настраивать AtomMind Server для проверки лицензий через Сервер лицензий.

5.2.1.1 Централизованное управление шифрованием

Сервер лицензии можно использовать для централизованного управления ключами шифрования, с помощью которых серверы AtomMind шифруют секретные данные, хранящиеся в их базах данных 692.

Плагин общих настроек 207 Сервера лицензии включает таблицу **Ключи шифрования**, которая позволяет настроить ключи шифрования для использования каждым AtomMind Server в управляемой группе. Эта таблица определяет:

- **Ключ активации** для идентификации сервера.
- **Текущий ключ шифрования**, который уже используется сервером (только чтение).
- **Новый ключ шифрования** для сервера. Чтобы установить ключ шифрования для нового сервера, используйте действие **Сгенерировать случайный ключ шифрования** контекста общих настроек плагина Сервер лицензии. Скопируйте сгенерированный ключ в буфер обмена, добавьте новую запить в таблицу **Ключи шифрования** и вставьте ключ в поле **Новый ключ шифрования**. После сохранения таблицы сервер попытается установить новый ключ и перешифровать все данные при следующем запуске.
- **Статус применения ключа -** текущий статус ключа, либо **OK** (установлен), либо **В ожидании** (попытка установки будет совершена после следующего перезапуска сервера).
- **Дата применения ключа** - дата/время успешной установки ключа.
- **Дата последнего запроса** - дата/время последней попытки установить ключ (журналируется даже при неудачной попытке установки ключа).
- **Пробный прогон** - флаг, который активирует тесты шифрования.

5.3 Технические требования

Фактически *AtomMind Server* может быть запущен на любой ОС с поддержкой Java. Мы протестировали нашу платформу и убедились, что она работает на следующих операционных системах:

- *Windows 2000, XP, Vista, 2003 Server, 2008 Server, 7 и 2012 Server (32-bit и 64-bit версии)*
- *Linux (32-bit и 64-bit версии* с поддержкой Java*)*
- *Mac OS X*

Другие ОС, совместимые с AtomMind Server, включают AIX, Solaris, FreeBSD, и т.д.

Требования к аппаратному обеспечению

AtomMind - это универсальная платформа для мониторинга, управления и настройки различных электронных устройств. AtomMind Server с легкостью устанавливается на настольные компьютеры и ноутбуки. При этом следует иметь в виду, что при управлении масштабными сетями или многокомпонентными устройствами, генерирующими значительный объем данных, Вам необходимо внимательнее отнестись к вопросу аппаратного обеспечения и конфигурации используемой системы.

Повлиять на масштабирование системы могут несколько факторов. Первый - число устройств, которые Вы будете контролировать при помощи AtomMind. Когда устройств более 100, может потребоваться некоторая оптимизация. Второй - объём данных, собираемый с каждого из устройств. Третий - интервал опроса. Например, если Вы установите интервал опроса для сбора данных с устройств в одну секунду вместо значения интервала по умолчанию, AtomMind Server'у будет значительно тяжелее работать, и его системные требования заметно возрастут. В конце концов, число одновременно работающих с системой операторов тоже значительно влияет на производительность.

При планировании установки AtomMind помните, что использование CPU и памяти сильно зависит от типов управляемых устройств и их драйверов. В ситуациях, когда под управлением находится множество устройств и их драйверов, рекомендуется использовать выделенный, более быстрый сервер.

Использование сервера RAM зависит от:

- Количества подключенных устройств
- Количества параметров, операций и типов событий в устойстве
- Количества и сложности методов обработки данных (тревог, отчётов и, в особенности, виджетов)
- Количества одновременно выполняемых клиентских соединений

Если у Вас есть дополнительные вопросы, связанные с производительностью, пожалуйста, свяжитесь с ТВЭЛ, чтобы поговорить с одним из наших системных инженеров или посетите сайт AtomMind.

При создании уникального решения на базе AtomMind следует рассматривать платформу больше как "среду разработки", нежели как "коробочный продукт". Исходя из этой точки зрения, очевидно, что фактические требования к аппаратному обеспечению будут полностью зависеть от логики вашего продукта, а не от таких параметров, как количество устройств или число входящих событий в секунду. К сожалению, это означает, что в уникальных случаях может понадобиться окружение для тестирования с целью оценки аппаратных требований к рабочему серверу.

Пример:

Модель 810 с десятью привязками, выполняемыми раз в секунду, практически не будет влиять на процессор и память, если эти привязки просто выполняют какие-то подсчеты, например, оценку KPI отдельного устройства на основе его первичных метрик.

Однако те же привязки вызовут большую нагрузку на процессор и использование большого количества памяти, если каждая из них будет выполнять запрос 829, который извлекает миллионы элементов из унифицированной модели и осуществляет их сортировку/фильтрацию/группировку.

При планировании использования пространства на диске, нужно учитывать, что наибольший объем, используемый AtomMind Server, обычно занимает хранилище событий 73.

Можно определить объем используемого дискового пространства, оценив максимальное количество событий для постоянного хранения (включая исторические данные переменных) 61 и умножив его на средний объем памяти, необходимый для события, т.е. около 1 КБ.

Аппаратные требования AtomMind Server

Мы предлагаем несколько таблиц по конфигурации аппаратных средств, подходящих для систем с низкой, средней и высокой нагрузкой на устройства.

Нижеприведенные требования к аппаратной части приблизительны. Например, единичный контроллер, обеспечивающий десять тысяч каналов ввода-вывода, запрашиваемых с частотой в 10 Гц, может требовать специальный мощный AtomMind Server для соответствующей обработки и хранения данных. С другой стороны, один сервер может поддерживать 100 000 аппаратных Агентов 684, подсоединяющихся к серверу, каждый из которых раз в день рассылает нескольких событий.

Требования к памяти предполагают, что ОС занимает:

- 50% оперативной памяти, если оперативная память сервера менее 2 Gb
- 1 Gb оперативной памяти, если оперативная память сервера более 2 Gb

РЕКОМЕНДУЕМОЕ ОБОРУДОВАНИЕ ДЛЯ СЕРВЕРОВ С ОЧЕНЬ НИЗКОЙ НАГРУЗКОЙ НА КАЖДОЕ УСТРОЙСТВО

Очень низкая нагрузка предполагает в среднем до 30 настроек/операций/типов событий на одно устройство с относительно малым набором простых инструментов для обработки данных (тревоги, отчёты, датчики, модели, виджеты). Каждое устройство генерирует в среднем менее 100 событий и обновлений метрик в день. Требования к дисковому пространству рассчитаны с учетом хранения исторических значений и событий в течение 1 года.

Число устройств	CPU	ядра CPU	RAM	Дисковое пространство	Комментарии
До 10	500 МГц	1	256 Мб	1 Гб	такие конфигурации совместимы со встраиваемыми средами (например, ARM Linux)
До 30	1 ГГц	1	512 Мб	1 Гб	такие конфигурации совместимы со встраиваемыми средами (например, ARM Linux)
До 100	1,5 ГГц	1	1 Гб	2 Гб	
До 300	3 ГГц	2	2 Гб	3 Гб	

До 1000	3 ГГц	4	4 Гб	7 Гб	настоятельно рекомендуется 64-битная система
До 3000	3 ГГц	8	8 Гб	20 Гб	только 64-битная система, выделенный сервер БД
До 10000	3 ГГц	16	16 Гб	70 Гб	64-битная система, выделенный сервер БД
До 30000	3 ГГц	24	32 Гб	200 Гб	64-битная система, выделенный сервер БД, рассматривается использование множества серверов внутри распределенной архитектуры
До 100000	3 ГГц	32	64 Гб	700 Гб	64-битная система, выделенный сервер БД, рассматривается использование множества серверов внутри распределенной архитектуры
Более 100000	3 ГГц	48+	128 Гб и более	2000 Gb	64-битная система, выделенный сервер БД, настоятельно рекомендуется использование множества серверов внутри распределенной архитектуры

РЕКОМЕНДУЕМОЕ ОБОРУДОВАНИЕ ДЛЯ СЕРВЕРОВ С НИЗКОЙ НАГРУЗКОЙ НА КАЖДОЕ УСТРОЙСТВО

Низкая нагрузка предполагает в среднем 30-100 настроек/операций/типов событий на одно устройство и среднее количество инструментов по обработке данных (тревоги, отчеты, датчики, модели, виджеты). Каждое устройство генерирует в среднем менее 1000 событий и обновлений метрик в день. Требования к дисковому пространству рассчитаны с учетом хранения исторических значений и событий в течение 1 года.

Число устройств	**CPU**	**ядра CPU**	**RAM**	**Дисковое пространство**	**Комментарии**
До 10	1 ГГц	1	512 Мб	2 Гб	
До 30	1.5 ГГц	1	1 Гб	2 Гб	
До 100	3 Ггц	2	2 Гб	7 Гб	
До 300	3 ГГц	4	4 Гб	20 Гб	настоятельно рекомендуется 64-битная операционная система
До 1000	3 ГГц	8	8 Гб	70 Гб	только 64-битная операционная система, рекомендуется выделенный сервер БД
До 3000	3 ГГц	16	16 Гб	200 Гб	требуются 64-битная операционная система и выделенный сервер БД
До 10000	3 ГГц	24	32 Гб	700 Гб	требуется 64-битная операционная система и выделенный сервер БД, рассматривается использование множества серверов внутри распределенной архитектуры
До 30000	3 ГГц	32	64 Гб	2000 Гб	требуется 64-битная операционная система и выделенный сервер БД, рассматривается использование множества серверов внутри распределенной архитектуры
Более 30000	3 ГГц	48+	128 Гб и более	7000 Гб	требуется 64-битная операционная система и выделенный сервер БД, настоятельно рекомендуется использование множества серверов внутри распределенной архитектуры

РЕКОМЕНДУЕМОЕ ОБОРУДОВАНИЕ ДЛЯ СЕРВЕРОВ СО СРЕДНЕЙ НАГРУЗКОЙ НА КАЖДОЕ УСТРОЙСТВО

Средняя нагрузка предполагает в среднем 100-300 настроек/операций/типов событий на одно устройство, с большим количеством сложных инструментов по обработке данных (тревоги, отчеты, датчики, модели, виджеты). Каждое устройство генерирует в среднем менее 10000 событий и обновлений метрик в день. Требования к дисковому пространству рассчитаны с учетом хранения исторических значений и событий в течение 1 года.

Число устройств	**CPU**	**ядра CPU**	**RAM**	**Дисковое пространство**	**Комментарии**
До 10	1.5 ГГц	1	2 Гб	7 Гб	
До 30	3 ГГц	2	4 Гб	20 Гб	

До 100	3 ГГц	4	4 Гб	70 Гб	настоятельно рекомендуется 64-битная операционная система
До 300	3 ГГц	8	8 Гб	200 Гб	только 64-битная операционная система, рекомендуется выделенный сервер БД
До 1000	3 ГГц	16	16 Гб	700 Гб	требуются 64-битная операционная система и выделенный сервер БД
До 3000	3 ГГц	24	32 Гб	2000 Гб	требуются 64-битная операционная система и выделенный сервер БД, рассматривается использование множества серверов внутри распределенной архитектуры
До 10000	3 ГГц	32	64 Гб	7000 Гб	требуются 64-битная операционная система и выделенный сервер БД, рассматривается использование множества серверов внутри распределенной архитектуры
Более 10000	3 ГГц	48+	128 Гб и более	20 Тб	требуется 64-битная операционная система и выделенный сервер БД, настоятельно рекомендуется использование множества серверов внутри распределенной архитектуры

РЕКОМЕНДУЕМОЕ ОБОРУДОВАНИЕ ДЛЯ СЕРВЕРОВ С ВЫСОКОЙ НАГРУЗКОЙ НА КАЖДОЕ УСТРОЙСТВО

Высокая нагрузка предполагает в среднем 300-1000 настроек/операций/типов событий на одно устройство, с большим количеством сложных инструментов по обработке данных (тревоги, отчеты, датчики, модели, виджеты). Каждое устройство генерирует в среднем менее 100 000 событий и обновлений метрик в день. Типовыми устройствами такого класса являются контроллеры или аппаратные шлюзы, агрегирующие данные с множества устройств нижнего уровня или датчиков. Требования к дисковому пространству рассчитаны с учетом хранения исторических значений и событий в течение 1 года.

Число устройств	**CPU**	**ядра CPU**	**RAM**	**Дисковое пространство**	**Комментарии**
До 3	3 ГГц	2	4 Гб	20 Гб	
До 10	3 ГГц	4	4 Гб	70 Гб	настоятельно рекомендуется 64-битная операционная система
До 30	3 ГГц	8	8 Гб	200 Гб	только 64-битная операционная система, рекомендуется выделенный сервер БД
До 100	3 ГГц	16	16 Гб	700 Гб	только 64-битная операционная система, рекомендуется выделенный сервер БД
До 300	3 ГГц	24	32 Гб	2000 Гб	требуются 64-битная операционная система и выделенный сервер БД, рассматривается использование множества серверов внутри распределенной архитектуры
До 1000	3 ГГц	32	64 Гб	7000 Гб	требуются 64-битная операционная система и выделенный сервер БД, рассматривается использование множества серверов внутри распределенной архитектуры
От 1000	3 ГГц	48+	128 Гб и более	20 Тб	требуется 64-битная операционная система и выделенный сервер БД, настоятельно рекомендуется использование множества серверов внутри распределенной архитектуры

РЕКОМЕНДУЕМОЕ ОБОРУДОВАНИЕ ДЛЯ СЕРВЕРОВ С ОЧЕНЬ ВЫСОКОЙ НАГРУЗКОЙ НА КАЖДОЕ УСТРОЙСТВО

Очень высокая нагрузка предполагает в среднем 1-3 тысячи настроек/операций/типов событий на одно устройство, с большим количеством сложных инструментов по обработке данных (тревоги, отчеты, датчики, модели, виджеты). Каждое устройство генерирует в среднем менее 10 000 000 событий и обновлений метрик в день. Типовыми устройствами такого класса являются корпоративные системы, интегрированные в AtomMind в качестве источника данных и "экспонирующих" множество управляемых ими устройств/метрик/событий через единственный аккаунт устройства в AtomMind. Требования к дисковому пространству рассчитаны с учетом хранения исторических значений и событий в течение 1 года.

Число устройств	**CPU**	**ядра CPU**	**RAM**	**Дисковое пространство**	**Комментарии**
До 1	3 ГГц	4	4 Гб	700 Гб	настоятельно рекомендуется 64-битная операционная система

До 3	3 ГГц	8	8 Гб	2000 Гб	только 64-битная операционная система, рекомендуется выделенный сервер БД
До 10	3 ГГц	16	16 Гб	7000 Гб	только 64-битная операционная система, рекомендуется выделенный сервер БД
До 30	3 ГГц	24	32 Гб	20 Тб	требуются 64-битная операционная система и выделенный сервер БД, рассматривается использование множества серверов внутри распределенной архитектуры 1332
До 100	3 ГГц	32	64 Гб	70 Тб	требуются 64-битная операционная система и выделенный сервер БД, рассматривается использование множества серверов внутри распределенной архитектуры 1332

РЕКОМЕНДУЕМОЕ ОБОРУДОВАНИЕ ДЛЯ СЕРВЕРОВ С ЭКСТРЕМАЛЬНО ВЫСОКОЙ НАГРУЗКОЙ НА КАЖДОЕ УСТРОЙСТВО

Экстремально высокая нагрузка предполагает в среднем более 3 тысяч настроек/операций/типов событий на одно устройство, с большим количеством сложных инструментов по обработке данных (тревоги, отчеты, датчики, модели, виджеты). Каждое устройство генерирует в среднем менее 1 000 000 000 событий и обновлений метрик в день. Типовыми устройствами такого класса являются высокопроизводительные гейтвеи и корпоративные системы, интегрированные в AtomMind в качестве источника данных и генерирующие большой объем данных. Требования к дисковому пространству рассчитаны с учетом хранения исторических значений и событий в течение 1 месяца.

Число устройств	CPU	Ядра CPU	RAM	Дисковое пространство	Комментарии
До 1	3 ГГц	16	16 Гб	7 Тб	только 64-битная операционная система, рекомендуется выделенный сервер БД
До 3	3 ГГц	24	32 Гб	20 Тб	требуются 64-битная операционная система и выделенный сервер БД
До 10	3 ГГц	32	64 Гб	70 Тб	требуются 64-битная операционная система и выделенный сервер БД, рассматривается использование множества серверов внутри распределенной архитектуры 1332
От 10	3 ГГц	48+	128 Гб и более	200 Тб	требуется 64-битная операционная система и выделенный сервер БД, настоятельно рекомендуется использование множества серверов внутри распределенной архитектуры 1332

Аппаратные требования к внешней базе данных

Если AtomMind Server использует внешнюю базу данных 692, требования к аппаратным средствам для сервера базы данных обычно те же, что требования к компьютеру, самостоятельно запускающему AtomMind Server. Следовательно, требования к аппаратным средствам для сервера базы данных зависят от количества устройств, подключенных к AtomMind Server, и от средней нагрузки на каждое устройство.

AtomMind Server должен быть подключен к серверу базы данных через скоростную выделенную сеть. Выделенный гигабитный Ethernet-канал с временем задержки (время пинга) менее 5 миллисекунд обеспечит достаточную производительность почти в любом случае.

В случае использования внешней базы данных NoSQL, AtomMind Server должен быть оснащен твердотельным накопителем (SSD) емкостью 50 Гб. Этот накопитель будет использоваться для хранения конфигурации. Требования к дисковому пространству в этом случае применяются к серверу внешней базы данных.

Рекомендуемый размер внешней базы данных NoSQL - 12-16 Гб.

Программные требования

AtomMind Server требует для работы следующие программы:

ВИРТУАЛЬНАЯ МАШИНА JAVA (JVM)

JVM - единственное обязательное требование для запуска AtomMind Server. JVM может идти как в комплекте с инсталлятором, так и быть уже установленной в системе (см Установка 153). Минимальная версия JVM для запуска AtomMind Server - 1.8 (Java 8).

РЕЛЯЦИОННАЯ БАЗА ДАННЫХ (RDBMS)

AtomMind Server требует для работы RDBMS. Дистрибутивы ТВЭЛ включают простейший сервер БД по умолчанию. Для больших установок мы рекомендуем использовать выделенные системы реляционных баз данных для промышленной аппаратуры или интегрированную базу данных NoSQL. *AtomMind Server* совместим с широким спектром серверов баз данных:

- MySQL
- PostgreSQL
- Oracle
- Microsoft SQL Server
- Firebird
- и многие другие...

AtomMind Server и RDBMS не обязательно должны быть установлены на одном и том же сервере. Возможно настроить *AtomMind Server* для работы с удаленной базой данных, но в этом случае на производительность также будет влиять состояние сети.

СЕРВЕР DNS

Чтобы использовать возможности динамического DNS, Вам нужен совместимый с RFC-2136 DNS-сервер. Это может быть как локальный, так и удаленно установленный DNS сервер. Спецификация RFC-2136 называется "Динамические обновления в доменной системе имен". Динамические обновления поддерживаются большинством современных DNS-серверов, включая DINS и Windows 2000/2003 DNS.

ПОЧТОВЫЙ СЕРВЕР SMTP

AtomMind Server требует почтового сервера для отправки извещений о тревогах по электронной почте. Теоретически AtomMind Server может "общаться" с любым почтовым сервером SMTP, включая те, которые требуют авторизации.

ТРЕБОВАНИЯ К ПОРТАМ

Номер порта	**Описание**
6460	Сокет *AtomMind Server* для клиентских соединений
6440	Номер порта Net Admin
8443	Безопасные веб-подключения (https) (Настройка сервера -> Веб-приложения)
8080	8080 - Незащищенные веб-соединения (http) (Настройка сервера -> Веб-приложения)
162	Порт прослушивания SNMP-ловушек (Драйверы/Плагины -> SNMP)
6420	Прослушивает подключения пользователей (Драйверы/Плагины -> Распределенная архитектура)
6430	Прослушивает подключения поставщика (Драйверы/Плагины -> Распределенная архитектура)
2055	Порт NetFlow (Драйверы/Плагины -> NetFlow)
6343	Порт sFlow (Драйверы/Плагины -> NetFlow)
514	Порт Syslog (Драйверы/Плагины -> Syslog)
6450	Номер порта для прослушивания соединений сервера устройств (Драйверы/Плагины -> Серверы устройств)
26460	GlobalSat (Драйверы/Плагины -> Датчик)
26461	Maestro MicroTracker (Драйверы/Плагины -> Датчик)
6480	Номер порта для прослушивания AtomMind (Драйверы/Плагины -> Агент)
50000 - 60000	Порты, назначенные для серверов устройств
21	Порт FTP (Драйверы/Плагины -> Сетевое устройство)
22	Порт SSH (Драйверы/Плагины -> Сетевое устройство)
69	Порт TFTP (Драйверы/Плагины -> Сетевое устройство)
5001	Порт базы данных ключевых значений по умолчанию

9042	Порт базы данных NoSQL по умолчанию
47808	Порт локального устройства BACnet
7800	Порт Heartbeat
53	Порт динамического DNS сервера
389	Соответствие групп Active Directory / LDAP пользователям AGG
47808	Настройка локального устройства BACnet
143	IMAP
110	POP3
25	SMTP
1812	RADIUS

5.4 Ветви и версии

Существуют две отдельные ветви AtomMind и версий производных продуктов:

- **Production**. Версии, выпущенные внутри этой ветви, как правило, предназначены для коммерческого развертывания. В этой ветви не добавляется новый функционал, поэтому каждый ее релиз имеет только устраненные ошибки.
- **Программа раннего доступа (EAP)**. Версии этой ветви содержат весь новый функционал, только что добавленный командой разработки. В то же время, EAP версии могут содержать ошибки, и нет гарантий, что каждая новая версия ветви EAP будет более стабильной, чем предыдущая. Обычно это происходит накануне того, как EAP становится ветвью Production (период так называемого "feature freeze" , когда новый функционал больше не добавляется в EAP, и каждый следующий релиз содержит только исправленные ошибки).

Рекомендуем использовать EAP-версии только партнерам, которым нужны новые функции определенной EAP ветви, и которые собираются сделать EAP ветвью Production до окончания своего проекта.

наименование версий

Номер версии AtomMind имеет следующую структуру: A.BC.DD

A значение увеличивается при каждом мажорном релизе раз в 6-12 месяцев

B значение увеличивается при минорном релизе раз в 2-6 месяцев

C значение увеличивается при релизе с исправленными ошибками, который происходит:

- если минорный релиз запускается в производство и
- и если в производственном релизе исправлены серьезные ошибки

DD значение увеличивается при минорном релизе с исправленными ошибками, который выпускается после исправления ошибок или добавления нового функционала, необходимых для инсталляции AggreGate в процессе развертывания

5.5 Установка и обновление

В этой главе описывается установка *AtomMind Server.* Перед тем, как приступить к инсталляции, обязательно ознакомьтесь с установочными требованиями 146 и самой процедурой 153.

Большинство установочных комплектов AtomMind включают AtomMind Server и AtomMind Client 359. Установка Клиента производится по желанию, в то время как компонент Сервер является обязательным.

AtomMind Client также распространяется в виде отдельного дистрибутива.

5.5.1 Установка

ТВЭЛ поставляет подготовленные к инсталляции пакеты для следующих операционных систем:

- *Различные версии Windows*
- Различные версий *Linux*
- *Mac OS X*

Инсталляторы для других поддерживающих Java операционных систем (FreeBSD и прочие UNIX'ы, Solaris, и др.) доступны по запросу.

Чтобы установить продукт, просто запустите установочную программу и следуйте инструкциям. В некоторых системах Вам может потребоваться сделать установщик "исполняемым" перед тем, как запустить его.

Режимы установки

Установка может происходить в **графическом (GUI)**, **консольном** и "**скрытом**" режимах.

Графический (GUI) режим выбирается по умолчанию.

Консольный режим активируется запуском установщика с опцией `'-c'` в командной строке. Режим консоли удобен для систем, не имеющих графического интерфейса (как большинство серверов с *UNIX*).

Установка в "скрытом" режиме активируется запуском установщика с опцией `'-q'` в командной строке. В этом режиме установщик системы использует настройки по умолчанию и не взаимодействует с пользователем.

Выбор БД

По умолчанию AtomMind использует базу данных NoSQL 693 (Apache Cassandra), предназначенную для промышленного использования в сценариях с низкой и средней нагрузкой. Позднее можно переключить 698 сервер на внешний сервер БД NoSQL или на БД другого типа.

Открытие портов

Приложение AtomMind Server принимает подключения/данные на большое количество TCP и UDP портов. Ниже представлен список портов, на которые должны быть разрешены входящие подключения:

- 6460 - подключения AtomMind Client 359 и подключения через AtomMind Server API 1340
- 6470 и 6480 - подключения других серверов через распределенную архитектуру 1332
- 8080 и 8443 - подключения к Web UI 220 и другим веб приложениям, работающим внутри интегрированного веб сервера
- Любые другие порты, требуемые драйверами и плагинами AtomMind Server (такие как UDP порт 162 для приема SNMP ловушек)

Опции командной строки установщика

Опция	Описание
`-manual`	Опция применяется только к Microsoft Windows. Последовательность поиска JVM по умолчанию не будет выполняться, и не будут использоваться связанные JVMs. Такой установщик будет действовать так, будто ни одна JVM не найдена, и отобразит диалог, позволяющий выбрать JVM или загрузить ее, если JVM была связана динамически. Если пользователь локализует JVM, она будет использоваться для установленных приложений. На Unix есть возможность определить переменную среды `INSTALL4J_JAVA_HOME_OVERRIDE` вместо переопределения последовательности поиска JVM по умолчанию.
`-console`	Если установщик исполняется в "скрытом" режиме, и `-console` передается в качестве второго параметра, консоль будет расположена на Windows, отображая выход установщика.
`-overwrite`	В "скрытом" режиме установщик не будет переписывать файлы, чья политика переписывания требует подтверждения пользователем. Если установлена опция `-overwrite`, все такие файлы будут переписаны.
`-dir [directory]`	Устанавливает другой каталог установки для "скрытого" режима установки. Следующим параметром должен быть желаемый каталог установки.

5.5.1.1 Установка на устройствах на базе ARM Linux

Для установки AtomMind на устройство на базе процессора ARM с операционной системой Linux, выполните следующие действия:

- Загрузите версию AtomMind для операционной системы Linux, которая **не включает Java Virtual Machine (JVM)**. Имя установочного файла должно заканчиваться на `-nojvm`. В большинстве случаев должна быть установлена так называемая **Micro Edition** версия AtomMind.
- Загрузите и установите комплект разработчика приложений на языке Java для платформы ARM с официального сайта компании Oracle. Некоторые версии Linux (напр. Raspbian Linux, работающий на плате Raspberry Pi) позволяют устанавливать Oracle Java как пакет программ, например при запуске `sudo apt-get update && sudo apt-get install oracle-java8-jdk`.

Допускается также использование других виртуальных машин Java с поддержкой Java 8.

- Установите Java CommAPI, который является библиотекой для работы с последовательным портом, используемой AtomMind Server. На многих системах, это можно сделать при помощи команды `sudo apt-get install librxtx-java`. Когда установка будет завершена, скопируйте файл `librxtxSerial.so` из папки установки Java CommAPI в подпапку `/lib` папки установки AtomMind Server.
- Запустите установочный пакет AtomMind. Если не удается найти JVM, задайте переменную окружения `INSTALL4J_JAVA_HOME_OVERRIDE` и укажите в ней путь к `/path/to/jdk/jre`.
- Следуйте инструкциям для установки системы.
- Некоторые версии Java SE для платформы ARM не поддерживают "режим сервера" Java VM. Если запускающий файл AtomMind выдает сообщение об ошибке режима сервера, удалите параметр `-server` из Файлов параметров запуска 162 (`*.vmoptions`).
- Скорректируйте ограничения на объем памяти 163 в файлах параметров запуска (`*.vmoptions`) для значений, подходящих для вашей среды.

5.5.1.2 Работа в контейнере Docker

Чтобы инсталлировать AtomMind в Docker, выполните следующие шаги:

1. Установите Docker, основанный на вашей ОС:

```
$ apt-get install docker
```

2. Запустите установщик AtomMind:

```
$ ./AtomMind_full_x.x.x_unix-x64.sh
```

3. Обратитесь к папке AtomMind:

```
$ cd /opt/AtomMind
```

4. Создайте в этой папке файл с названием Dockerfile и впишите туда следующий текст:

```
# Dockerfile
#OS template
FROM centos:7
#Working AtomMind dir in docker image
WORKDIR /AtomMind
ADD . /AtomMind
#Next lines needs to enable working with widgets
```

```
RUN yum install -y xorg-x11-server-Xvfb
RUN yum install -y libXrender.x86_64
RUN yum install -y libXtst.x86_64
RUN yum install -y which
#Ports opened from docker. You could add all needed ports
EXPOSE 6460
EXPOSE 8080
EXPOSE 6480
CMD xvfb-run --auto-servernum ./am_server -r
```

5. Запустите процесс создания docker:

```
$ docker build -t AtomMind .
```

6. Проверьте созданное изображение в локальном реестре docker:

```
root@ubuntu:/opt/AtomMind # docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
AtomMind  latest 32a4dd3d23e8 4 hours ago 1.272 GB
centos 7 328edcd84f1b 2 weeks ago 192.5 MB
```

7. Теперь вы можете запустить приложение с изображения docker:

```
$ docker run AtomMind
```

Также вы можете создать том для хранения данных и смонтировать его в Docker.

5.5.2 Обновление

Обновление - сложная процедура. Убедитесь, что вы создали резервные копии данных согласно инструкции, приведённой ниже, чтобы минимизировать время простоя системы в случае, если обновление не удастся.

ТВЭЛ старается максимально обеспечить обратную совместимость между всеми релизами AtomMind, включая мажорные. Однако в редких случаях нам приходится изменять компоненты платформы, структуры единой модели данных 41 и программные интерфейсы. Из-за этого могут понадобиться незначительные доработки продуктов и сервисов на базе платформы после обновления до новой мажорной версии.

Для обновления AtomMind Server'а до новой версии внимательно следуйте инструкциям, чтобы не допустить повреждения базы данных сервера:

1. Остановите все AtomMind Client
2. Остановите AtomMind Server или его сервис, если сервер запущен в Режиме сервиса Windows 159
3. **Сделайте резервную копию базы данных!** Перейдите в раздел Резервное копирование и восстановление базы данных 173 для получения более подробной информации

Резервное копирование БД - важная операция. Игнорирование этого действия перед обновлением может повлечь за собой потерю данных!

4. **Сделайте резервную копию установочной директории AtomMind Server**. Перейдите в раздел Резервное копирование и восстановление базы данных 173 для получения более подробной информации

Резервное копирование установочной директории сервера - это важная операция, которую нельзя пропускать в производственной среде.

5. Обновите сервер, запустив инсталлятор новой версии
6. Обновите все установленные у Вас AtomMind Client. Убедитесь, что Вы **сделали резервную копию установочных директорий AtomMind Client** перед обновлением.
7. Обновите прошивки всех устройств 497, если необходимо
8. Запустите AtomMind Server
9. Проверьте журнал сервера 166, чтобы убедиться в успешном запуске и что база данных была обновлена

Если обновление не удалось или система после обновления не работает как ожидалось, сделайте откат к предыдущей версии AtomMind Server и AtomMind Client восстановлением их установочных каталогов и базы данных AtomMind Server из резервной копии. Свяжитесь со службой технической поддержки ТВЭЛ для решения Вашей проблемы.

Создание новых ресурсов

Обновлённая версия AtomMind может содержать множество новых встроенных ресурсов (тревоги, отчеты, виджеты и пр.). Эти ресурсы автоматически не создаются в момент обновления, так что если Вы хотите использовать их, запустите действие **Создать ресурсы** ( ) из корневого контекста 1559 и выберите ресурсы, которые нужно создать. См. Управление встроенными ресурсами 208.

5.5.2.1 Удаленное обновление

Можно обновить AtomMind Server до новой версии, загрузив новый установочный пакет через интерфейс управления сервера, не запуская установщик из окружения ОС.

Для удаленного обновления AtomMind Server, просим вас внимательно следовать инструкции, чтобы не повредить базу данных сервера:

1. Скачайте новую версию AtomMind Server. Убедитесь, что вы загрузили версию, подходящую к ОС вашего сервера.
2. Выберите опцию **Обновить сервер** в корневом контексте 1559.
3. Выберите инсталлятор AtomMind Server, который вы загрузили.
4. Выберете мгновенную или отложенную установку.
5. Нажмите "OK".
6. Если вы используете AtomMind Client 359, Обновите все инсталляции клиента, чтобы они соответствовали новой версии сервера

Для удаленного обновления требуется больше ресурсов, прежде всего, оперативной памяти, чем для обновления стандартным методом. При нехватке оперативной памяти обновление будет прекращено, не повлияв на систему.

5.5.3 Деинсталляция

AtomMind Server деинсталлируется запуском исполняемого файла `uninstall` из установочной директории AtomMind Server'a. Перед деинсталляцией рекомендуется остановить сервер.

Деинсталлятор AtomMind Server'a не удаляет файлы, которые были изменены или созданы в процессе работы AtomMind Server (файлы конфигурации, базы данных, журналы, и пр.). Чтобы полностью вычистить данные AtomMind Server'a, удалите установочную директорию после того как деинсталлятор закончит свою работу. Иногда может требоваться "чистая" переустановка.

5.5.4 Руководство по миграции

Этот раздел описывает действия, необходимые для миграции между основными версиями AtomMind.

миграция с версии 5.5X.XX на 5.6X.XX

Данное руководство поможет подготовить ваш AtomMind Server к обновлению с версии 5.5.x до 5.6.x.

Пожалуйста, сделайте резервную копию папки AtomMind Server и всех баз данных, используемых в вашем проекте.

ПОДГОТОВКА ХРАНИЛИЩА СОБЫТИЙ NOSQL (CASSANDRA)

Это гайд поможет вам обновить ваше встроенное хранилище NoSQL до обновления AtomMind Server . AtomMind Server версий 5.4.x, 5.5.x и 5.60.01 использует Cassandra 2.0.12. AtomMind Server использует движок Cassandra версии 3.11, поэтому необходимо обновить версию до актуальной перед первым запуском версии 5.6.x.

Порядок обязательных шагов описан ниже:

1. Настройте переменную среды JAVA_HOME в вашей ОС.
2. Загрузите и извлеките сервер Cassandra версии 2.1.
3. Укажите правильный путь для параметров `hints_directory, data_file_directories, commitlog_directory, saved_caches_directory` в файле конфигурации `cassandra.yaml`. Он должен вести в папку текущей БД Cassandra внутри AtomMind Server (папка **nosql_data**)
4. Запустите сервер Cassandra.
5. Выполните `nodetool upgradesstables` из отдельной папки `bin` сервера Cassandra.
6. Обновите отдельный сервер Cassandra до следующей версии.

Повторите шаги 2-6 для версий 2.2, 3.0 и 3.11 сервера Cassandra.

ОБНОВЛЕНИЕ ВЫРАЖЕНИЙ ПОСЛЕ МИГРАЦИИ НА ВЕРСИЮ 5.6.X

Мы обновили протокол, чтобы уменьшить количество метаданных, и удалили из него дополнительные символы кодирования.

Вы можете увидеть дополнительные `%` символы в некоторых выражениях. В более старых версиях нашей платформы, мы хранили их в БД. Вам нужно удалить дополнительные `%` символы из ваших выражений и сохранить изменения. После этого действия правильные данные сохранятся в БД конфигурации и будут корректно работать в версии 5.6.x.

ОБНОВЛЕНИЕ СКРИПТОВ

Так как в новой версии Java API был изменен, вам также придется обновить ваши скрипты на Java. В особенности нужно будет поменять все ваши команды с классом DataTable (например, DataTable dataTable = new DataTable()) на команды с использованием SimpleDataTable (SimpleDataTable simpleDataTable = new SimpleDataTable()).

5.5.5 Перемещение инсталляции AtomMind Server

Перемещение или репликация инсталляции AtomMind Server с одной машины на другую может быть полезна в следующих случаях:

- для перемещения с инструментального компьютера на рабочий сервер на первых этапах развития проекта
- если рабочий сервер переходит на новую аппаратную конфигурацию или операционную систему

Процесс перемещения включает следующие шаги:

1. Установка AtomMind Server на новой машине
2. Копирование файла конфигурации
3. Репликация базы данных
4. Репликация статистики

Установка нового AtomMind Server

Новая копия AtomMind Server должна быть установлена на новом сетевом копьютере. Версия AtomMind Server, установленная на новой машине, должна быть той же (рекомендуется) или новее, чем версия AtomMind Server перемещенной инсталляции. Языки инсталляций должны совпадать.

Новая инсталляция должна включать тот же набор расширений вертикального рынка (и, следовательно, плагины 207 и драйверы 518 AtomMind Server), как предыдущая инсталляции.

Копирование файла конфигурации

На втором этапе файл общих настроек 178 `server.xml` нужно скопировать из старой инсталляции в новую переопределением файла конфигурации новой инсталляции по умолчанию.

Репликация базы данных

Третий этап - это перемещение базы данных сервера 692 в новую инсталляцию. Существует несколько способов осуществления этого:

1. Установка новой копии движка базы данных на новом сервере и перемещение файлов базы данных вручную соответственно инструкциям по перемещению базы данных сервера.
2. Установка новой копии движка базы данных на новом сервере, создание пустой базы данных, следование инструкции переключение на другой движок базы данных 698 с целью перенесения старой базы данных в новую.
3. Если новая инсталляция AtomMind Server должна использовать ту же автономную базу данных, которая использовалась старой, просто проверьте/измените параметр `databaseUrl` в скопированном файле конфигурации сервера (`server.xml`), убедившись, что IP-адрес или имя хоста сервера базы данных - те самые, которые нужно использовать из нового местоположения сервера.

Если AtomMind Server использует базу данных MySQL или Apache Derby, идущие в комплекте, процесс перемещения базы данных будет очень простым:

- Для базы данных MySQL в комплекте: просто скопируйте подпапку `/mysql` старой установочной папки AtomMind Server на новый сервер. Убедитесь, что оба сервера остановлены во время этого процесса! Когда копия создана, убедитесь, что MySQL автоматически запускается при запуске нового сетевого компьютера. Например, на машине Windows можно использовать команду `mysqld --install`, чтобы установить MySQL как сервис. См. подробности здесь: http://dev.mysql.com/doc/refman/4.1/en/windows-start-service.html
- Для базы данных Apache Derby в комплекте: просто скопируйте подпапку `/db` предыдущей установочной папки AtomMind Server на новый сервер. Убедитесь, что оба сервера остановлены во время этого процесса!

Репликация статистики

Для перемещения кольцевой базы данных, используемой модулем статистического контроля процессов 718:

- Убедитесь, что новый и старый серверы остановлены
- Скопируйте или переместите папку `/statistics` из старой инсталляции в новую. Поверьте параметр `statisticsFolder` старого конфигурационного файла (`server.xml`), чтобы разместить папку статистики в старой инсталляции (она расположена в установочной папке AtomMind Server по умолчанию). Проверьте или измените параметр `statisticsFolder` нового файла конфигурации инсталляции, чтобы узнать точное нахождение скопированной папки статистики.

5.6 Запуск и остановка

В этой главе описываются запуск и остановка AtomMind Server.

В большинстве случаев, сервер запускается в фоновом режиме сервиса 159. Однако, ручной запуск 160 также может использоваться, особенно на начальных стадиях развертывания, настройки и тестирования системы.

Разные способы остановки сервера описываются в главе Остановка сервера 161.

AtomMind Server может нуждаться в доступе к привилегированным портам, то есть портам с номерами ниже 1024. Например, он слушает SNMP-прерывания 162 UDP порта. Таким образом, если сервер запущен на Linux или Mac OS, он получает права привилегированного пользователя.

5.6.1 Режим сервиса операционной системы

AtomMind Server можно запускать в *режиме сервиса операционной системы*. Сервис работает независимо от авторизованных пользователей и может быть запущен, даже если нет ни одного авторизованного пользователя. Сервис не зависит от того, открыта ли консоль, и не может самостоятельно открывать окна. Режим сервиса и его поведение зависит от используемой ОС.

Сервис AtomMind Server регистрируется инсталлятором. Его конфигурация для авто-загрузки задается во время загрузки операционной системы.

Реализция режима сервиса зависит от операционной системы:

- Для Microsoft Windows режим сервиса представлен *сервисом Windows*
- Для Linux режим сервиса представлен *фоновым (daemon) процессом*

Microsoft Windows

Установленным сервисом можно управлять через *Start* > *Control Panel* > *Administration* > *Services*.

УСТАНОВКА И ДЕИНСТАЛЛЯЦИЯ СЕРВИСА

Сервис можно также установить из командной строки, передав `/install` выполняемому сервису (`am_server_service.exe`). Чтобы избежать автоматическую загрузку сервиса при перезагрузке компьютера, следует передать аргумент `/install-demand`.

В качестве второго параметра после параметра `/install` можно опционально передать имя сервиса. Таким способом можно присвоить сервису новое имя вместо того, что было дано по умолчанию.

Деинсталляцию сервиса можно выполнить, передав выполняемому сервису параметр `/uninstall`.

Для всех переключателей командной строки также указывают тире в виде префикса, а не косую черту (т.е., `-uninstall`) или двойное тире (т.е., `--uninstall`).

ЗАПУСК И ОСТАНОВКА СЕРВИСА ИЗ КОМАНДНОЙ СТРОКИ

Для запуска и остановки сервиса доступны элементы выбора `/start` и `/stop`. Кроме того, аргумент `/status` показывает, запущен ли уже сервис. Код выхода для команды статуса, когда сервис запущен, - 0, когда сервис не запущен - 3, и 1 - когда состояние статуса невозможно определить (например, когда сервис не установлен на Windows).

АВТОМАТИЧЕСКИЙ ПЕРЕЗАПУСК СЕРВИСА

Сервис Windows будет автоматически перезапущен при выходе с кодом выхода, отличным от нуля, или в случае сбоя. По сути это означает, что сервис всегда будет доступен до явной остановки через любой пользовательский интерфейс AtomMind Server или через контрольную панель сервиса Windows.

Linux/Unix

Разные версии Linux используют разные подходы для сервисов автозапуска и автовореспауна (демонов), так что инсталлятор AtomMind Server делает все возможное для конфигурирования поведения любого автозапуска и автоматического перезапуска по умолчанию. Вот как это технически организовано:

AtomMind Server инсталляция Linux содержит несколько скриптов, которые можно использовать для автозапуска и автоматического перезапуска сервера:

- скрипт `am_server` запускает сервер и ждет до его остановки. Он возвращает код выхода AtomMind Server, который соответствует `0`, если был запрос на остановку сервера, и `1`, если был запрос на перезапуск сервера. Другие ненулевые коды могут быть возвращены в случае остановки сервера по причине критической ошибки.
- скрипт `am_server_service` запускает сервер как демон. Доступны аргументы `start`, `stop`, `restart` или `status` для запуска/остановки/перезапуска сервера или, соответственно, для отчета о его статусе.

АВТОЗАПУСК И АВТОРЕСПАУН ДЛЯ СИСТЕМ С ДЕМОНОМ ИНИЦИАЛИЗАЦИИ UPSTART

Для автозапуска и автореспауна сервера на системах, использующих демон инициализации `Upstart`, инсталлятор AtomMind Server копирует файл конфигурации `am_server.conf` в директорию `/etc/init`.

Для запуска сервера впервые, выполните команду `initctl start am_server`.

АВТОЗАПУСК И АВТОРЕСПАУН ДЛЯ СИСТЕМ С ДЕМОНОМ ИНИЦИАЛИЗАЦИИ SYSTEMD

Для автозапуска и автореспауна сервера на системах, использующих демон инициализации `systemd`, инсталлятор AtomMind Server копирует файл конфигурации `am_server.service` в директорию `\etc\systemd\system`.

Чтобы активировать автозапуск и автореспаун сервиса, выполните команду `systemctl enable am_server.service`.

Для запуска сервера впервые, выполните команду `systemctl restart am_server.service`.

Обратите внимание, что инсталлятор AtomMind Server не активирует автоматический автозапуск и автореспаун на основе сервиса `systemd`. Для перезагрузки демона systemctl выполните команду `systemctl daemon-reload`

Для проверки работоспособности сервиса выполните команду `systemctl status am_server_service.service -l`.

АВТОЗАПУСК БЕЗ АВТОРЕСПАУНА

В очень редких случая может понадобиться активировать автозапуск AtomMind Server при запуске ОС, но без перезапуска, если перезапуск запрашивается через пользовательский интерфейс сервера.

Для активации автозапуска демона AtomMind Server при запуске ОС, но без перезапуска даже при соответствующем запросе через пользовательский интерфейс, используйте скрипт `am_server_service`:

- Создайте символьную ссылку на него в директории инициализации сервисов (`/etc/init.d` или `/etc/rc.d/init.d`), или
- Выполните команду `update-rc.d am_server_service defaults`(для Ubuntu и схожих версий Linux), или
- Выполните команды `chkconfig --add am_server_service` и `chkconfig am_server_service on` (для RedHat и LSB-совместимых версий Linux)

ОС Mac

Инсталлятор AtomMind Server создает LauncherDaemon, который должен обеспечивать автозапуск сервера при запуске ОС.

5.6.2 Ручной запуск

AtomMind Server можно запустить вручную при помощи специальной *Программы запуска,* созданной во время процесса инсталляции. Тип этой программы зависит от операционной системы:

Инсталлятор AtomMind Server создает три программы запуска, чье расширение зависит от операционной системы:

- `am_server` для обычного запуска
- `am_server_console` для запуска сервера и отображения его вывода в окне консоли (отметим, что журналирование 166 в файлы и другие места в этом случае не отключается)
- `am_server_service` для запуска сервера как сервиса 159

Ручной запуск в Windows

В *Windows* программы запуска AtomMind Server доступны через *Меню Старт* (*Старт* > *Программы* > *AtomMind* > *Сервер)* или ярлык на рабочем столе:

Для просмотра журнала запуска сервера в окне консоли запустите приложение `am_server_console.exe`.

Ручной запуск в Linux

Программы запуска AtomMind Server Linux - это выполняемые shell-скрипты. Для запуска сервера вручную запустите одну из следующих команд:

- `am_server_console` для синхронного запуска, который показывает журнал сервера в текущем окне консоли. Отметим, что сервер будет остановлен, например, при вводе клавишной команды `Ctrl-C`.
- `am_server_service start` для запуска сервера как демона. Отметим, что сервер не будет автоматически перезапускаться, если программа перезапуска запускается через любой пользовательский интерфейс (такой как AtomMind Client 359).
- `service am_server_service start` для запуска сервера как демона, если его скрипт/символьная ссылка автозапуска доступны в `/etc/init.d` или `/etc/rc.d/init.d`

Ручной запуск в Mac OS

В Mac OS X AtomMind Server можно запустить из командной строки со следующей командой (при условии, что приложение установлено в папке `/Application/AtomMind/`):

```
sudo /Applications/AtomMind/am_server.app/Contents/MacOS/JavaApplicationStub
```

Проблемы запуска сервера

Если ваш AtomMind Server не запускается или работает некорректно, пожалуйста, обратитесь к следующим разделам:

- Проблемы запуска AtomMind Server 212
- Ошибки во время работы AtomMind Server 214
- AtomMind Server не отвечает 216

5.6.3 Остановка сервера

Есть несколько способов остановить AtomMind Server:

- использовать приложение AtomMind Client 359 во время подключения к серверу с соответствующими правами.
- использовать Web UI 220, войдя в систему с соответствующими правами.
- использовать меню System Tray Icon 173.
- использовать интерфейс *Net Admin* 1448.
- Остановить сервис Windows/Linux, если он запущен в сервисном режиме 159.
- Вручную прекратить все процессы сервера, если сервер не отвечает на попытки остановить процессы.

В некоторых случаях попытка остановки и записи кэшированных данных в базу может занять у AtomMind Server несколько минут. Обрывание процессов во время остановки с использованием средств операционной системы может привести к потере данных.

При остановке виртуальной машины, на которой запущен AtomMind Server, автоматически выполняется и остановка AtomMind Server.

Методы Остановки Сервера

Все методы остановки AtomMind Server описаны ниже.

ИСПОЛЬЗОВАТЬ ПРИЛОЖЕНИЕ ATOMMIND CLIENT

Остановка сервера через соединение AtomMind Client:

- Найдите узел сервера ( ) в Системном Дереве 370
- Кликните по нему правой кнопкой мыши
- выберите **Остановить Сервер** из контекстного меню

Вам необходимо иметь разрешение Администратора 486 в Корневом контексте 1559, чтобы удалённо остановить сервер.

ИСПОЛЬЗОВАТЬ WEB UI

Остановка сервера при помощи Web UI 220:

- Найдите узел сервера ( ) в Системном Дереве 370
- Кликните по нему правой кнопкой мыши
- выберите **Остановить Сервер** из контекстного меню

Вам необходимо иметь разрешение Администратора 486 в Корневом контексте 1559, чтобы удалённо остановить сервер.

ИСПОЛЬЗОВАНИЕ TRAY MENU

Выберите элемент **Остановить Сервер** в меню панели задач 173 AtomMind Server.

Меню панели задач недоступно, если сервер запущен в сервисном режиме 159.

ОСТАНОВИТЬ СЕРВИС ОПЕРАЦИОННОЙ СИСТЕМЫ

Остановить AtomMind Server, если он запущен в сервисном режиме 159:

- В Windows следуйте **Меню Пуск > Панель Управления > Инструменты Администрирования > Сервисы**, найдите сервис `AtomMind Server`, кликните по нему правой кнопкой мыши и выберите **Стоп**.
- В Linux, откройте консоль и наберите `service am_server_service stop`.

ИСПОЛЬЗОВАТЬ ИНТЕРФЕЙС NET ADMIN

Войдите в интерфейс *Net Admin* 1448 через `telnet` в порт `6440` и используйте его, чтобы остановить (команда `S`) или перезапустить (команда `R`) сервер.

ПРЕКРАТИТЬ ПРОЦЕССЫ ВРУЧНУЮ

Этот метод не рекомендуется, поскольку он может прекратить другие процессы *Java,* запущенные на хосте. Его следует использовать только в случае, если сервер не отвечает на все другие попытки его остановить.

В *Windows* следует прекратить процессы **am_server.exe**, **am_server_service.exe**, **am_server_console.exe**. Прекратите все экземпляры данных процессов при помощи *Диспетчера Задач* (`Ctrl+Shift+Esc` в *Windows*).

В *Unix* следует прекратить процессы **am_server**, **am_server_service**, **am_server_console** и **java**. Одновременно может выполняться несколько процессов на Java. Прекратите процесс с **наименьшим ИНП** (идентификационным номером процесса). Это важно, потому что прекращение другого процесса не устранит проблему.

Существуют особые случаи, когда нужно прекратить другой процесс, а не тот, который имеет наименьший ИНП, но это особые случаи, которые зависят от специальных настроек сервера. Прекращение процесса с наименьшим ИНП, скорее всего, устранит проблему.

5.6.4 Файлы свойств Java VM

Файлы свойств Java VM используются для передачи дополнительных опций виртуальной машине Java, запускающей AtomMind Server. Эти файлы находятся в одной директории с самими запускающими модулями, т.е. в папке установки AtomMind. Их названия соответствуют названиям запускающих модулей и имеют расширение `.vmoptions`. Например, дополнительные опции для запускающего модуля `am_server.exe` передаются при помощи Файла Свойств Запускающего Модуля `am_server.vmoptions`. В этом файле каждая строка интерпретируется как единичный параметр виртуальной машины.

Необходимо добавить новый символ перевода строки (т.е., нажать *ENTER,* когда закончится ввод текста строки) в конце каждой строки файла .vmoptions. Если в конце строки нет нового символа перевода строки, она не будет учитываться!

Например, если у вас имеется однострочный файл .vmoptions, строка должна заканчиваться новым символом перевода строки:

`-classpath/a jdbc-driver.ja`*<newline_character>*

Изменение языка

Файл свойств запускающего модуля по умолчанию содержит опцию `-Duser.language=X`, которая определяет язык, используемый AtomMind Serverом. Язык по умолчанию выбирается в процессе установки AtomMind Servera.

Если язык был изменен после первого запуска AtomMind Servera, база данных 692 будет содержать в языке данные (описание ситемного ресурса, события и т.д.), которые были активны во время предыдущих запусков.

Изменение пути к классу

Следующие опции нужно добавить к файлам *.vmoptions, чтобы изменить *путь к классу* AtomMind Server'a Виртуальной Машины Java:

`-classpath [classpath]`	Замена путь к классу сгенерированного запускающего модуля.
`-classpath/a [classpath]`	Добавка в конец пути к классу сгенерированного запускающего модуля.
`-classpath/p [classpath]`	Вставка в начало пути к классу сгенерированного запускающего модуля.

Важно поставить один пробел между параметром пути к классу и его значением.

НОВЫЙ ТЕРМИН: *Путь к классу* - это имя каталога, указывающее, где в локальной системе находятся компилируемые файлы *Java*.

Опция Файла Свойств Запускающих Модулей **Добавка в конец пути к классу** нужна в случае создания новых библиотек Java, доступных для AtomMind Servera. Например, путь к файлу драйвера Соединение с Базой Данных Java следует добавить в конец пути к классу, когда переходите на другой механизм базы данных 698.

Настройка использования памяти

Следующие опции можно добавить к файлам *.vmoptions, чтобы контролировать объём памяти, используемый AtomMind Serverом:

`-Xms`	Исходный объем памяти, используемый AtomMind Serverом. За этим параметром должно следовать количество мегабайт и символ `m`, без пробелов. Например, параметр `-Xms200m` даёт указание AtomMind Servery использовать 200 мегабайт памяти при запуске. Значение по умолчанию - 100 мегабайт.
`-Xmx`	Максимальный объем памяти, который может использоваться AtomMind Serverом. За этим параметром должно следовать количество мегабайтов и символом `m`, без пробелов. Например, параметр `-Xmx1024m` даёт AtomMind Servery команду использовать максимум 1 гигабайт. Значение по умолчанию - 800 мегабайт. **Слишком высокое значение параметра -Xmx может привести к сбою запуска AtomMind Servera, потому что Виртуальной Машине Java требуется данный объем памяти, имеющийся в одном непрерывном блоке. В этом случае перейдите в 64-битную операционную систему и установите 64-битную версию AtomMind Servera.**
`-Xss`	Размер стека одиничного потока сервера - 128 килобайт по умолчанию. Размер стека нужно уменьшить, если сбой при создании потока вызвал ошибку **Нехватка Памяти**, появившуюся в журнале регистрации AtomMind Servera. Увеличьте размер стека, если драйверы устройств или системные компоненты приводят к ошибке **Переполнение Стека**.

Настройка прокси сервера

Следующие свойства можно добавить к файлам *.vmoptions, чтобы включить использование прокси сервера для подключения AtomMind Server к интернету:

-Dhttp.proxyHost	Имя хоста или адрес прокси сервера с поддержкой HTTP
-Dhttp.proxyPort	Номер порта прокси сервера с поддержкой HTTP
-Dhttp.proxyUser	Имя пользователя на случай, если HTTP прокси сервер запрашивает аутентификацию
-Dhttp.proxyPassword	Пароль на случай, если HTTP прокси сервер запрашивает аутентификацию
-Dhttp.nonProxyHosts	Указывает хосты, доступ к которым не осуществляется через прокси. Обычно, это внутренние хосты. Значение этого свойства - список хостов, разделенных символом '|'. Кроме этого может использоваться специальный символ '*' для соответствия шаблону. Например, Dhttp.nonProxyHosts="*.foo.com|localhost" будет показывать, что доступ по всем хостам в домене foo.com и на локальной машине должен осуществляться напрямую, даже если указан прокси сервер. Обратите внимание, что обработчик протокола HTTPS будет использовать то же свойство nonProxyHosts, что и для HTTP протокола.
-Dhttps.proxyHost	Имя хоста или адрес прокси сервера с поддержкой HTTPS
-Dhttps.proxyPort	Номер порта прокси сервера с поддержкой HTTPS
-Dhttps.proxyUser	Имя пользователя на случай, если HTTPS прокси сервер запрашивает аутентификацию
-Dhttps.proxyPassword	Пароль на случай, если HTTPS прокси сервер запрашивает аутентификацию
-Dsocks.proxyHost	Имя хоста или адрес прокси сервера с поддержкой SOCKS
-Dsocks.proxyPort	Номер порта прокси сервера с поддержкой SOCKS
-Dsocks.proxyUser	Имя пользователя на случай, если SOCKSv5 сервер запрашивает аутентификацию
-Dsocks.proxyPassword	Пароль на случай, если SOCKSv5 сервер запрашивает аутентификацию

SOCKS позволять осуществлять туннелирование на более низком уровне, т.к. работает на уровне TCP. Если в настройках стоит SOCKS прокси сервер, все TCP соединения пойдут через прокси, если не указаны другие прокси.

Изменение опций использования сокетов

Следующие опции можно добавить к файлам *.vmoptions, чтобы изменить сетевые параметры, используемые AtomMind Server:

-Dsun.net.maxDatagramSockets	Устанавливает максимальное количество сокетов датаграмм. Опция позволяет увеличить число UDP-сокетов, которые может использовать виртуальная машина Java, например, если при попытке синхронизировать большое количество устройств вы столкнетесь с ошибкой SNMP "достигнуто максимальное количество сокетов датаграмм".

5.6.5 Параметры командной строки

AtomMind Server можно запускать с любыми из следующих параметров командной строки:

Параметр	Описание
-a	Активировать безопасный режим 172.
-c	Создать структуру базы данных 692. Эта опция дает указание AtomMind Server попытаться (пере)создать схему БД, включая все таблицы и индексы. Во время этой операции не будут

	затронуты существующие таблицы или индексы.
-e <filename>	Выполнить файл скрипта 165 <filename> во время загрузки.
-f *filename* >	Использовать файл глобальной конфигурации 178 *filename* >.
-h	Печатать список настроек командной строки и выйти.
-o *directory* >	Установить в качестве домашней директории *directory* >.
-p *user* > *pass* >	Установить новый пароль *pass* > для учётной записи 478 *user* >. **Например:** %program files%\AtomMind\Server\AtomMind_server.exe -p admin 12345 Эта команда запустит AtomMind Server и изменит пароль для **администрирующего** пользователя на **12345**.
-s *filename* >	Выполнить SQL команды 166, содержащиеся в файле *filename* > со время загрузки.
-u	Обновить структуру базы данных 692 после обновления AtomMind Server до новой версии. Эта опция заставляет AtomMind Server создавать недостающие в БД таблицы, добавлять недостающие поля в существующие таблицы. Обратите внимание, что во время этой операции индексы не создаются. При переходе на новый движок БД 698 для создания всех таблиц с индексами необходимо всегда использовать опцию командной строки -c.

5.6.6 Скрипты запуска

AtomMind Server может выполнять два типа скриптов во время запуска: командные скрипты и скрипты SQL.

Командные скрипты

Командный скрипт может содержать команды, аналогичные командам Net Admin 1448, одна команда на строчку. Эти команды описаны в Руководстве по командам управления и запуска 1449. Команды, не применимые во время запуска сервера (т.е. "*stop server*", "*restart server*"), не выполняются. Любой вывод исполняющей команды не принимается во внимание.

Командные скрипты можно использовать для отладки поврежденных инсталляций AtomMind Server или для предварительного создания системных объектов (Фильтров событий 762, Предупреждений об ошибке 779 и т.п.) в особых случаях. Командные скрипты могут включать в себя *комментарии*: строки, начанающиеся с **#,** не будут обработаны.

Чтобы выполнить командный скрипт, запустите AtomMind Server с опцией командной строки -e <filename>, (где filename -- это имя скрипта файла).

ПРИМЕРЫ КОМАНДНЫХ СКРИПТОВ

```
# Создать новый фильтр событий с названием "Keytroller Events" для всех пользователей системы

C/users.*.filters/create/keytroller/Keytroller Events

# Показать дополнительные поля в журнале событий

S/users.*.filters.keytroller/additionalFields/description/Description/Keytroller Events/1/timestamp/Description/Keytroller Events/1/userid/Description/Keytroller Events/1/username/Description/Keytroller Events/1/machine/Description/Keytroller Events/1/data/Description/Keytroller Events/1

# Не показывать названия событий в журнале

F/users.*.filters.keytroller/shownFields/event/0

# Не показывать данные событий в журнале

F/users.*.filters.keytroller/shownFields/data/0

# Не показывать события подключений в журнале

F/users.*.filters.keytroller/shownFields/ack/0
```

```
# Установить драйвер Keytroller по умолчанию
F/config/general/defaultDevicePlugin/com.tibbo.AtomMind
Server.plugin.device.keytroller
```

Скрипты SQL

Скрипты SQL содержат команды SQL, которые выполняются непосредственно в БД AtomMind Server во время запуска сервера, один скрипт на одну строку. Чтобы выполнить командный скрипт, запустите AtomMind Server с опцией командной строки `-s <filename>` (где `filename` -- это имя файла скрипта). Скрипты SQL могут включать в себя *комментарии*: строки, начинающиеся с **#,** не будут обработаны.

Выполнение команд SQL может испортить вашу БД AtomMind Server. Создавайте резервную копию перед тем, как выполнять любые скрипты SQL.

ПРИМЕР СКРИПТОВ SQL

```
# Срочное удаление аккаунта пользователя 'john' и всей связанной с ним информации:
# (может быть полезно, если аккаунт был поврежден)
# Удаление аккаунта пользователя
delete from ag_users where ag_username = 'john'
# Удаление аккаунта из Сервера устройств
delete from ag_deviceservers where ag_owner = 'john'
# Удаление всех хранимых свойств во всех контекстах
delete from ag_properties where ag_context like 'users.john.%'
# Удаление всех хранимых событий в контекстах пользовалетя John
delete from ag_events where ag_context like 'users.john.%'
```

5.7 Журналирование

Как и в большей части серверного ПО, в AtomMind Server важным методом журналирования событий является запись в лог-файлы. AtomMind Server имеет расширенные средства логирования, которые можно легко настраивать через текстовый конфигурационный файл. Настройку журналирования можно производить в процессе работы AtomMind Server, без перезапуска.

Наиболее важные операци сервера также журналируются как события 73 платформы, для гибкой проверки защиты. В разделе Управление событиями 761 описывается, как просматривать потоки событий в реальном времени и анализировать исторические события.

Для вывода записей журнала AtomMind Server использует библиотеку Apache Log4J. Это гибкое средство логирования позволяет разделять события из различных источников на множество уровней приоритета и перенаправлять вывод логов в:

- Консоль
- Текстовые лог-файлы
- Файлы формата XML
- Журналы Windows
- Системные журналы UNIX
- Базы данных
- Другие серверы в сети
- Сообщения электронной почты
- Служба сообщений Java (JMS)
- и многие другие...

Файл настройки журналирования в AtomMind Server располагается в основной директории инсталляции под именем `logging.xml`. Формат этого файла кратко описан в разделе Файл настроек журналирования 167. В нем также содержится список категорий логирования, используемых в различных компонентах AtomMind для вывода логов.

По умолчанию в файле настроек журналирования 167 определены два направления журналирования: **консоль** и **лог-файл** `server.log`, находящийся в подкаталоге `/logs` директории инсталляции AtomMind Server. Журналируются события на уровне 171 **Info**.

5.7.1 Файл настроек журналирования

В этом разделе описывается конфигурационный файл журналов с настройками по умолчанию и особенности журналирования, типичные для AtomMind Server.

Вот конфигурация по умолчанию:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Configuration monitorInterval="10">
  <Appenders>
     <Console name="console" target="SYSTEM_OUT">
       <ThresholdFilter level="debug"/>
       <PatternLayout pattern="%d{HH:mm:ss,SSS} %-5p %-25c %m     -     [%t] %C.%M (%F:%L)%n%throwable{full}"/>
     </Console>
     <RollingFile name="file" fileName="logs/server.log" append="true" filePattern="logs/server.%d{yyyy-MM-dd}.log">
       <ThresholdFilter level="debug"/>
       <PatternLayout pattern="%d{dd.MM.yyyy HH:mm:ss,SSS} %-5p %-25c %m     -     [%t] %C.%M (%F:%L)%n%throwable{full}" charset="UTF-8"/>
       <Policies>
         <TimeBasedTriggeringPolicy interval="1" modulate="true"/>
       </Policies>
     </RollingFile>
  </Appenders>

  <Loggers>
    <Logger name="ag" level="warn" additivity="false">
       <AppenderRef ref="file"/>
       <AppenderRef ref="console"/>
     </Logger>

     <Logger name="ag.alerts" level="info"/>

     <!-- Other logging categories are skipped -->

     <Root level="warn">
       <AppenderRef ref="console"/>
       <AppenderRef ref="file"/>
     </Root>
```

```
    </Loggers>
</Configuration>
```

Настройки журналирования по умолчанию представлены корневым элементом конфигурации, двумя аппендерами (содержат два направления журналирования) и несколькими именованными категориями журналирования. Элемент `<root>` определяет общий уровень журналирования событий как **предупреждение** (`<Root level="warn">`) (более подробно см. уровни журналирования 171). Эта настройка применяется ко всем сообщениям, приходящим от всех использующихся внутри сервера библиотек.

Сообщения от самого сервера регистрируются с уровнем `warn`. Это определяется тегом `level="warn"` внутри секции `<Logger name="ag"...>`. Чтобы понизить уровень до отладочного для некоторой категории 168, измените этот тег на `<level="debug"/>`. Элементы `<Logger>` определяют, для какого именно компонента AtomMind Server Вы настраиваете уровень логирования.

В этой конфигурации также определены два **аппендера** - `console` и `file`. Аппендер - это направление для введения данных журнала. Это могут быть текстовые файлы, сообщения электронной почты и пр.

`RollingFile` требует набора политик для определения, когда необходимо выполнить ротацию файла лога. `TimeBasedTriggeringPolicy` используется по умолчанию, то есть ротация будет выполняться раз в день. Одновременно могут использоваться различные политики.

Для более подробной информации изучите документацию по Log4j 171.

5.7.2 Категории журналирования

В этом разделе рассказывается о категориях журналирования, используемых в AtomMind Server, AtomMind Client 359 и прочих компонентах AtomMind.

Категория	Описание
ag.alerts	Сообщения, относящиеся к Оповещениям 779.
ag.autorun	Сообщения, относящиеся к Автозапуску 941.
ag.bindings	Сообщения, относящиеся к Привязке данных 735. Большая часть сообщений этой категории появляется при **отладке**.
ag.commands	Запись процесса обмена командами между различными компонентами системы. Большая часть сообщений этой категории появляется при **отладке**.
ag.commands.device_server	Команды Сервера устройств 2087 журналируются при **отладке**.
ag.commands.agent	Команды Агента 684 журналируются при **отладке**.
ag.commands.client	Команды Клиента 359 журналируются при **отладке**.
ag.commands.distributed	Команды потребитель-провайдер распределенной архитектуры 1332 журналируются при **отладке**.
ag.commondata	Сообщения, относящиеся к Общим данным 2176.
ag.classes	Сообщения, относящиеся к Классам 885.
ag.clients	Сообщения, относящиеся к AtomMind Clients 359.
ag.cluster	Сообщения, относящиеся к Отказоустойчивому кластеру и балансировщику нагрузки 1326.
ag.config	Сообщения, относящиеся к Общей Настройке 177 AtomMind Server.
ag.context	Сообщения, относящиеся к Контекстам 41. Большая часть сообщений этой категории появляется при **отладке**.
ag.context.info	Информационные 77 события в контексте AtomMind Server записываются в эту категорию на **информационном** уровне.

ag.context.children	Сообщения, относящиеся к управлению деревом контекстов. Большая часть сообщений этой категории появляется при **отладке**.
ag.context.variables	Сообщения, относящиеся к контексту Переменные (Свойства) 61. Большая часть сообщений этой категории появляется при **отладке**.
ag.context.functions	Сообщения, относящиеся к контексту Функции 70. Большая часть сообщений этой категории появляется при **отладке**.
ag.context.events	Сообщения, относящиеся к контексту События 73. Большая часть сообщений этой категории появляется при **отладке**.
ag.context.actions	Сообщения, относящиеся к контексту Действия 87. Большая часть сообщений этой категории появляется при **отладке**.
ag.context.*	Другие сообщения от различных плагинов 207 контекста.
ag.core	В эту категорию попадают сообщения из ядра AtomMind Server и AtomMind Client. Различные сообщения, которые сложно отнести к какой-то конкретной категории, тоже попадают сюда.
ag.core.thread	Сообщения о потоках выполнения.
ag.dashboards	Сообщения, относящиеся к Информационным панелям 912.
ag.database	Взаимодействие с базой данных. Большая часть сообщений этой категории появляется при **отладке**.
ag.device	Базовые сообщения, относящиеся к взаимодействию AtomMind Server с устройствами.
ag.device.discovery	Сообщения, относящиеся к обнаружению устройств.
ag.device.sync	Сообщения, относящиеся к синхронизации 514 AtomMind Server с устройствами.
ag.device.agent	Сообщения, относящиеся к Agent.
ag.device.*	Прочие сообщения от различных драйверов устройств 518.
ag.device_browser	Сообщения, относящиеся к компоненту Навигатор устройств 475 из AtomMind Client.
ag.device_server	Сообщения, относящиеся к Серверам устройств 2087.
ag.device_server.inbands	Сообщения, относящиеся к командам, проходящим в потоке данных Серверов устройств 2087 (при **отладке**).
ag.data_table	Сообщения, относящиеся к Таблицам данных 49. Большая часть сообщений этой категории появляется при **отладке**.
ag.data_table.filter	Сообщения, относящиеся к фильтрам Таблиц Данных 49 (используется Фильтром событий 762). Большая часть сообщений этой категории появляется при **отладке**.
ag.data_table_editor	Сообщения, относящиеся к компоненту Редактор таблиц данных 382 из AtomMind Client.
ag.dns	Сообщения, относящиеся к взаимодействию с серверами DNS. В основном используется драйвером Динамического DNS 2094.
ag.entity_selector	Сообщения, относящиеся к компоненту Селектор сущностей 402 из AtomMind Client.
ag.eventfilters	Сообщения, относящиеся к Фильтрам событий 762.
ag.event_log	Сообщения, относящиеся к компоненту Журнал событий 398 из AtomMind Client.

ag.expression_builder	Сообщения, относящиеся к компоненту Редактор выражений 404 из AtomMind Client.
ag.expressions	Сообщения, относящиеся к обработке Выражений 112.
ag.favourites	Сообщения, относящиеся к Закладкам 2186.
ag.groups	Сообщения, относящиеся к Группам 751.
ag.gui	Базовые сообщения, относящиеся к Graphics User Interface (GUI).
ag.gui_builder	Сообщения, относящиеся к компоненту Конструктор GUI 423 (из AtomMind Client).
ag.guide	Сообщения, относящиеся к Интерактивным руководствам 476.
ag.mail	Сообщения, относящиеся к обработке исходящих и входящих сообщений электронной почты.
ag.models	Сообщения, относящиеся к Моделям 810.
ag.net_admin	Сообщения, относящиеся к Администратору сети 1448.
ag.performance	Сообщения, относящиеся к вопросам производительности: какие операции выполняются дольше, чем ожидалось. В режиме **отладки** в эту категорию попадают также сообщения и о небольших потерях в производительности. Более серьезные проблемы записываются с более высокими значениями приоритета.
ag.plugins	Сообщения, относящиеся к Плагинам 207.
ag.properties_editor	Сообщения, относящиеся к компоненту Редактор свойств 377 из AtomMind Client.
ag.protocol	Сообщения, относящиеся к Коммуникационным протоколам AtomMind 2119. Большая часть сообщений этой категории появляется при **отладке**.
ag.protocol.caching	Сообщения, относящиеся к кэшированию в Коммуникационных протоколах AtomMind 2119. Большая часть сообщений этой категории появляется при **отладке**.
ag.queries	Сообщения, относящиеся к Запросам 829.
ag.reports	Сообщения, относящиеся к Отчетам 928.
ag.resource	Сообщения, относящиеся к управлению ресурсами. Большая часть сообщений этой категории появляется при **отладке**.
ag.scheduler	Сообщения, относящиеся к Запланированным заданиям 823.
ag.scripts	Сообщения, относящиеся к Скриптам 879 и Скриптам виджетов 1308.
ag.security	Сообщения, относящиеся к безопасности и правам доступа. Большая часть сообщений этой категории появляется при **отладке**.
ag.statistics	Сообщения, относящиеся к сбору статистики (Statistical Process Control).
ag.stdout	Стандартный вывод различных библиотек и компонент AtomMind перенаправляются в эту категорию. Журналирование в этой категории происходит при **отладке**.
ag.stderr	Стандартный вывод ошибок различных библиотек и компонент AtomMind перенаправляются в эту категорию. Журналирование в этой категории происходит на уровне **информационный**.
ag.store	Сообщения, относящиеся к Магазину.
ag.system_tree	Сообщения, относящиеся к компоненту Дерево свойств системы 370 из AtomMind Client.

ag.trackers	Сообщения, относящиеся к Датчикам 218.
ag.users	Сообщения, относящиеся к Пользователям 478.
ag.view	Сообщения, относящиеся к Просмотрам 888.
ag.web	Базовые сообщения, относящиеся к Web UI 354 AtomMind Servera.
ag.widgets	Сообщения, относящиеся к Виджетам 943.
ag.workflows	Сообщения, относящиеся к Процессам 895.

5.7.3 Уровни журналирования

AtomMind Server использует 5 заранее определенных уровней журналирования:

Фатальный ("fatal") уровень используется только для наиболее важных сообщений. Обычно после обнаружения и записи фатальной ошибки AtomMind Server останавливает свою работу.

После возникновения сообщений уровня **Ошибка** ("error") AtomMind Server не прекращает свое выполнение, однако появление в логах сообщений подобного уровня важности говорит о том, что при работе сервера возникла ситуация, когда не обойтись без вмешательства оператора.

Предупреждения ("warn") показывают, что что-то происходит не так, но в большинстве случаев AtomMind Server способен самостоятельно обработать проблемы подобного уровня и немедленного вмешательства оператора обычно не требуется.

В **Информационных** ("info") сообщениях AtomMind Server отображается обычная деятельность. Сообщения подобного рода используются администратором для отслеживания системы в рабочем режиме.

Отладочные ("debug") сообщения необходимы тогда, когда в AtomMind Server что-то происходит не так, как ожидает оператор. По умолчанию сообщения этого уровня отключены. Включение отладочных сообщений для всех категорий может привести к появлению огромного количества информации в логах и значительно понизить производительность AtomMind Server. Советуем включать этот уровень только *временно* и только если Вы уверены, что это действительно необходимо. Обычно этот уровень включают по совету от команды технической поддержки ТВЭЛ.

По умолчанию журналирование происходит на уровне **Информационных** сообщений. После установки и настройки системы с целью повышения производительности и уменьшения количества ненужной информации в журналах мы рекомендуем повысить уровень журналирования до **Предупреждений** и опускать до **Отладочного** при возникновении проблем и лишь для некоторых категорий.

Чтобы изменить уровень журналирования, выполните следующие действия:

- Откройте в редакторе файл logging.xml.
- Найдите в нем секцию, описывающую категории:

```
<Logger name="ag.category_name " level="info"/>
```

- Поменяйте в ней значение приоритета (см. информацию выше) на интересующий Вас уровень журналирования.
- Обычно изменения применяются в течении 20 секунд с момента сохранения файла на диск. В некоторых случаях может потребоваться перезапустить AtomMind Server, чтобы активировать изменения.

Строка `priority value=` может находиться в разных частях кофигурационного файла. Главное, что значение приоритета находится в секции `</root>`. Изменение значения в секции `root` (особенно для таких высоких значений уровня, как `debug`) может *значительно* повлиять на производительность, привести к зависанию **AtomMind Server** или даже его полной остановке.

5.7.4 Документация к библиотеке журналирования

В качестве подсистемы для ведения журналов в *AtomMind Server* используется **Log4j**. Это хорошо масштабируемая, устойчивая и гибкая библиотека. Полное описание её свойств выходит за рамки этого руководства. Исчерпывающую информацию о Log4j можно получить из следующих книг и веб-ресурсов:

- Официальный сайт Log4j

- Исчерпывающее руководство по Log4j
- Журналирование в Java с использованием JDK 1.4 Logging API и Apache log4j

5.8 Безопасный режим

Безопасный режим используется для обхода различных ошибок при запуске, запуска частично функционирующего AtomMind Server и удаления и/или починки ресурсов 208, вызывающих ошибки при запуске.

AtomMind Server можно запустить в *безопасном режиме*, добавив в командной строке параметр `-a`.

Безопасный режим отличается от обычного тем, что:

- Устройства 497 не синхронизируется с сервером
- Триггеры тревог 779 не обрабатываются, и оповещения не возникают
- Планировщик задач 823 не запускается, и отложенные задания не выполняются
- Датчики 218 значений не вычисляются
- Автозапуск 941 не выполняется
- Выполнение скриптов 879 запрещено
- Привязки моделей 810 не обрабатываются, и модели, по сути, неактивны
- Большинство других активностей сервера также не запущены

Безопасный режим независим от режима обслуживания 172. Можно использовать оба режима одновременно.

5.9 Режим обслуживания

Режим обслуживания позволяет администраторам AtomMind временно закрыть систему для использования обычными пользователями. Это полезно при запланированных периодах простоя, когда необходимо выполнить серьезную перенастройку систему.

Режим обслуживания управляется двумя действиями из корневого контекста 1559:

- Начать режим обслуживания
- Закончить режим обслуживания

Режим обслуживания независим от безопасного режима 172. Можно использовать оба режима одновременно.

ВКЛЮЧЕНИЕ РЕЖИМА ОБСЛУЖИВАНИЯ

После включения режима обслуживания, AtomMind Server выполняет следующие операции:

- Завершает все пользовательские сессии, кроме той, которая использовалась для запуска режима обслуживания
- Сохраняет статус режима обслуживания в файле конфигурации сервера 178 для возобновления обслуживания после перезапуска сервера.

ОПЕРАЦИИ В РЕЖИМЕ ОБСЛУЖИВАНИЯ

Режим обслуживания вносит некоторые изменения в нормальную работу системы:

- Никакие пользователи, кроме администратора по умолчанию 479, не могут войти в систему
- Невозможно подключить никакие устройства
- Статус соединения для всех устройств устанавливается как "Обслуживание"
- Любые API вызовы от сторонних систем запрещены (кроме API сессий, авторизованных как администратор по умолчанию 479)
- Все пользователи веб-интерфейса перенаправляются на специализированную страницу "Обслуживание"

5.10 Меню в системном трее

После запуска AtomMind Server в системном трее возникает новая иконка, которая выглядит так: 

Меню трея недоступно, если сервер запущен в сервисном режиме 159.

Клик по этой иконке вызывает появление меню со следующими вариантами выбора:

- **Открыть Центр Управления AtomMind Server**. Открывает в окне браузера Центр Управления AtomMind Server 354.
- **Открыть страницу загрузки AtomMind Клиента.** Перенаправляет на страницу загрузки AtomMind Client 359.
- **Открыть страницу помощи в браузере**. Открывает ссылку на это руководство.
- **Посмотреть ключ активации**. Позволяет увидеть и скопировать ключ активации 144.
- **Просмотреть журнал сервера**. Открывает окно журнала событий 398, показывая сообщения от журнала сервера 166. Технически показывает Журнал 145 событий контекста.
- **Перезапустить сервер**. Перезапускает сервер. Обращаем Ваше внимание, что перезапуск сервера возможен, только если он запущен в сервисном режиме 159.
- **Остановить сервер**. Останавливает работу AtomMind Server.

5.11 Резервное копирование и восстановление

AtomMind используется для критически важных приложений, поэтому оченьт важно обеспечить целостность данных и возможность быстрого восстановления после сбоев.

AtomMind Server разработан с возможностью легкого создания резервных копий как базы данных 692, так и конфигурации сервера. Настоятельно рекомендуется регулярно выполнять резервирование системы.

Полные резервные копии виртуального сервера

Если сервер и его база данных запущены на вирутальной машине (или машинах), самые простой и надежный способ резервного копирования - это сделать моментальные снимки всей виртуальной машины.

Настоятельно рекомендуем отдавать предпочтение резервному копированию на уровне виртуальной машины, по сравнению с другими способами.

Частичное резервное копирование

Резервное копирование пошаговой установки AtomMind может включать следующие этапы:

- **Резервное копирование установочной папки сервера**. Периодическое резервное копирование настоятельно рекомендуется, когда инсталляция AtomMind Server не дублируется отказоустойчивым сервером 1326.
- **Резервное копирование файла конфигурации сервера**. Рекомендуется, если полное резервное копирование установочной папки сервера не производится.
- **Резервное копирование статистики**. Рекомендуется, если полное резервное копирование установочной папки сервера не производится.
- **Резервное копирование встроенной базы данных**. Рекомендуется, если полное резервное копирование установочной папки сервера не производится.
- **Резервное копирование внешней базы данных**. Настоятельно рекомендуется, если внешняя база данных не дублируется, см. отказоусточивость базы данных 710.

Если планируется обновление сервера, необходимо выполнить следующие операции по резервному копированию до обновления:

- **Резервное копирование установочной папки сервера**
- **Резервное копирование внешней базы данных**

Резервное копирование установочной папки сервера

В процессе установки[152] AtomMind Server не происходит никаких модификаций в системных конфигурационных файлах или реестре Windows. Никакие файлы не сохраняются за пределами установочной директории. Сам AtomMind Server также сохраняет файлы, созданные при работе в нормальном режиме, в этой же директории.

Таким образом, чтобы создать полную резервную копию инсталляции AtomMind Server, достаточно просто сделать полную копию установочной директории сервера (т.е. `%program files%/AtomMind` в Microsoft Windows) и архивировать её. В случае серьезных проблем (таких, как порча диска или повреждение базы данных неправильно написанным скриптом[879]), восстановить архивированную инсталляцию можно простым извлечением последнего архива резервной копии на ту же машину, что и раньше, либо даже на совсем другой сервер. При этом не потребуется никаких дополнительных действий: сервер может быть запущен сразу после завершения процесса распаковки.

Резервная копия установочной папки сервера уже включает:

- Резервное копирование конфигурационного файла сервера (убедитесь, что используется конфигурационный файл по умолчанию, см. информацию ниже)
- Резервное копирование статистики (убедитесь, что используется местоположение статистики по умолчанию, см. информацию ниже)
- Резервное копирование встроенной базы данных (убедитесь, что используется встроенная база данных по умолчанию, см. информацию ниже)

Однако резервная копия установочной папки сервера **не включает резервную копию внешней базы данных**[174].

При восстановлении резервной копии на другой машине может понадобиться получить новую лицензию (см. раздел Лицензирование[144]).

Резервное копирование внешней базы данных

Крупные инсталляции AtomMind Server для множества управляемых устройств обычно сохраняют данные на внешнем сервере баз данных, таком как Apacha Cassandra, MySQL или Oracle. Такие инсталляции требуют, чтобы резервное копирование внешней базы данных выполнялось вдобавок к Резервному копированию установочной папки сервера[173], чтобы получить полную копию инсталляции AtomMind Server.

Сервер хранит все свои данные в нескольких таблицах в одной базе данных (см. разделы База данных[692] и Настройка базы данных[182], чтобы понять, какая база данных используется для сохранения данных сервера). В процессе создания резервной копии сервера необходимо выполнить и резервное копирование базы данных. Способ создания резервной копии базы данных зависит от разновидности движка базы данных, используемого инсталляцией AtomMind Server, и выходит за рамки этой статьи.

ПОдробные инструкции по резервному копированию

В этом разделе описаны инструкции по созданию резервных копий отдельных компонентов AtomMind Server.

РЕЗЕРВНОЕ КОПИРОВАНИЕ КОНФИГУРАЦИОННОГО ФАЙЛА СЕРВЕРА

Конфигурационный файл[178] сервера содержит основные настройки AtomMind Server, такие как имя инсталляции и настройки доступа к базе данных. Чтобы создать резервную копию конфигурационного файла сервера, создайте копию файла `server.xml`, расположенного в установочной папке AtomMind Server, и сохраните эту копию в безопасном месте.

Сервер может использовать конфигурационный файл с именем, указанным как параметр командной строки[164]. Убедитесь, что резервируете нужный файл.

РЕЗЕРВНОЕ КОПИРОВАНИЕ СТАТИСТИКИ

Каналы статистики[718] сохраняют данные в подпапке `/statistics` установочной папки AtomMind Server. Нужно создать резервную копию этой папки, чтобы сохранить статистические архивы.

Местоположение папки хранения статистики можно изменить в общих настройках сервера[182]. Убедитесь, что создаете резервную копию нужной папки.

РЕЗЕРВНОЕ КОПИРОВАНИЕ ВСТРОЕННОЙ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

Движок встроенной реляционной БД[697], включенный во все комплекты AtomMind Server, хранит данные базы данных в поддиректории /db инсталляции сервера.

Чтобы создать резервную копию всей базы данных, сделайте следующее:

1. Прекратите работу сервера
2. Скопируйте поддиректорию /db установочной директории AtomMind Servera в место хранения резервной копии.

Сжатие содержимого папки /db уменьшит размер резервной копии в 5-10 раз.

Местоположение встроенной базы данных можно изменить в общих настройках сервера[182]. Убедитесь, что создаете резервную копию нужной папки.

РЕЗЕРВНОЕ КОПИРОВАНИЕ И ВОССТАНОВЛЕНИЕ БАЗЫ ДАННЫХ POSTGRESQL

Если AtomMind Server использует базу PostgreSQL, можно создать ее резервную копию, совершив следующие действия:

- Чтобы создать резервную копию целой базы данных в файле, запустите следующую команду:

```
pg_dump -U USER -W PASSWORD DATABASE > database.sql
```

где *USER* и *PASSWORD* - это ваши учетные данные для входа в PostgreSQL, а *DATABASE* - это имя базы данных, используемое AtomMind Server.

- Чтобы восстановить целую базу данных из файла, запустите следующую команду:

```
psql -U USER -W PASSWORD DATABASE < database.sql
```

РЕЗЕРВНОЕ КОПИРОВАНИЕ БАЗЫ ДАННЫХ MYSQL

Если AtomMind Server использует базу данных MySQL, можно создать ее резервную копию, совершив следующие действия:

- Чтобы создать резервную копию целой базы данных в файле, запустите следующую команду:

```
mysqldump -u USER -p PASSWORD --skip-lock-tables DATABASE > database.sql
```

где *USER* и *PASSWORD* - это ваши учетные данные для входа в MySQL, а *DATABASE* - это имя базы данных, используемое AtomMind Server.

- Чтобы избежать копирования большого количества некритических событий, вы можете ограничить резервное копирование несколькими основными таблицами[706]:

```
mysqldump -u USER -p PASSWORD --skip-lock-tables DATABASE ag_properties ag_events
ag_data ag_alert > database.sql
```

СЖАТИЕ, ЗАГРУЗКА И ПЛАНИРОВАНИЕ РЕЗЕРВНЫХ КОПИЙ В LINUX

- Чтобы сжать конечный файл database.sql и сохранить его на удаленном безопасном сервере резервного копирования, скопируйте его через FTP:

```
mysqldump -u USER -p PASSWORD --skip-lock-tables DATABASE ag_properties ag_events
ag_data ag_alert | /home/LINUX_USER /db_pipe.sh
```

Вот текст скрипта db_pipe.sh:

```
sfx=`/bin/date +%Y-%m-%d`

/bin/gzip -c > /home/LINUX USER/db_backup.${sfx}.sql.gz

/usr/bin/lftp -e "put /home/LINUX USER/db_backup.${sfx}.sql.gz; bye" -u BACKUP_SERVER_USER,BACKUP_

/bin/rm -rf /home/LINUX_USER/db_backup.${sfx}.sql.gz
```

- Чтобы запланировать вышеобозначенное резервное копирование, добавьте следующую строку в настройку **крона** Linux/Unix:

```
0 0 * * * mysqldump -USER -PASSWORD --skip-lock-tables DATABASE ag_properties ag_events ag_data ag_alert | /home/LINUX_USER /db_pipe.sh
```

Данная конфигурация будет выполнять резервное копирование каждую полночь.

СЖАТИЕ, ЗАГРУЗКА И ПЛАНИРОВАНИЕ РЕЗЕРВНЫХ КОПИЙ В WINDOWS

- Скачайте и установите архиватор WinRar
- Скачайте и установите утилиту cURL
- Подготовьте командный файл backup.bat со следующим контентом:

```
"%AtomMind%\mysql\bin\mysqldump" -uUSER -pPASSWORD --skip-lock-tables DATABASE ag_properties ag_ev

rar.exe a sqlBackup d:\backup\MySQL.sql

del /q d:\backup\MySQL.sql

curl.exe -T d:\backup\sqlBackup.rar -u FTP_USER:FTP_PASSWORD ftp://BACKUP_SERVER

exit
```

- Запланируйте запуск этого командного файла, используя Планировщик Заданий Windows

Чтобы создать резервную копию на машине Linux с использованием SCP, воспользуйтесь утилитой WinSCP вместо cURL:

```
winscp.exe /console /script=D:\BackUp\transfer.txt
```

Содержимое transfer.txt:

```
open sftp://SSH_USER:SSH_PASSWORD@BACKUP_SERVER

cd /home/SSH_USER

option transfer binary

put d:\backup\sqlBackup.rar

close

exit
```

6 Настройка и решение проблем

Этот раздел посвящен настройке AtomMind.

6.1 Конфигурирование

Существует ряд способов конфигурирования AtomMind Server:

- используя утилиту Конфигуратор сервера 177.
- используя программное обеспечение AtomMind Client (см. Конфигурирование из клиента 177).
- через Web UI 220, используя стандартный браузер, например, Internet Explorer или Mozilla Firefox.
- редактируя конфигурационный файл сервера (см. Файловая конфигурация 178). Это позволяет выявить неисправности и может быть использовано для правки одной или двух настроек, когда AtomMind Server не может запуститься, а другие способы решения неисправности недоступны.

6.1.1 Конфигуратор сервера

Конфигуратор сервера -- это небольшая утилита, сопровождающая AtomMind. Она открывает основные настройки сервера в Редакторе свойств 377 и позволяет их менять. Настройки загружаются и сохраняются в конфигурационном файле AtomMind (`server.xml` по умолчанию).

Конфигуратор сервера позволяет вносить изменения лишь в основные настройки сервера, которые хранятся в файлах 179 настройки сервера, но не в те, которые хранятся в БД. Таким образом, только небольшое количество настроек сервера можно изменить при помощи этой утилиты. Это по большей части настройки, необходимые для запуска сервера, например, настройки связи с БД.

Для того, чтобы внести изменения в другие настройки, используйте AtomMind Client или Web UI.

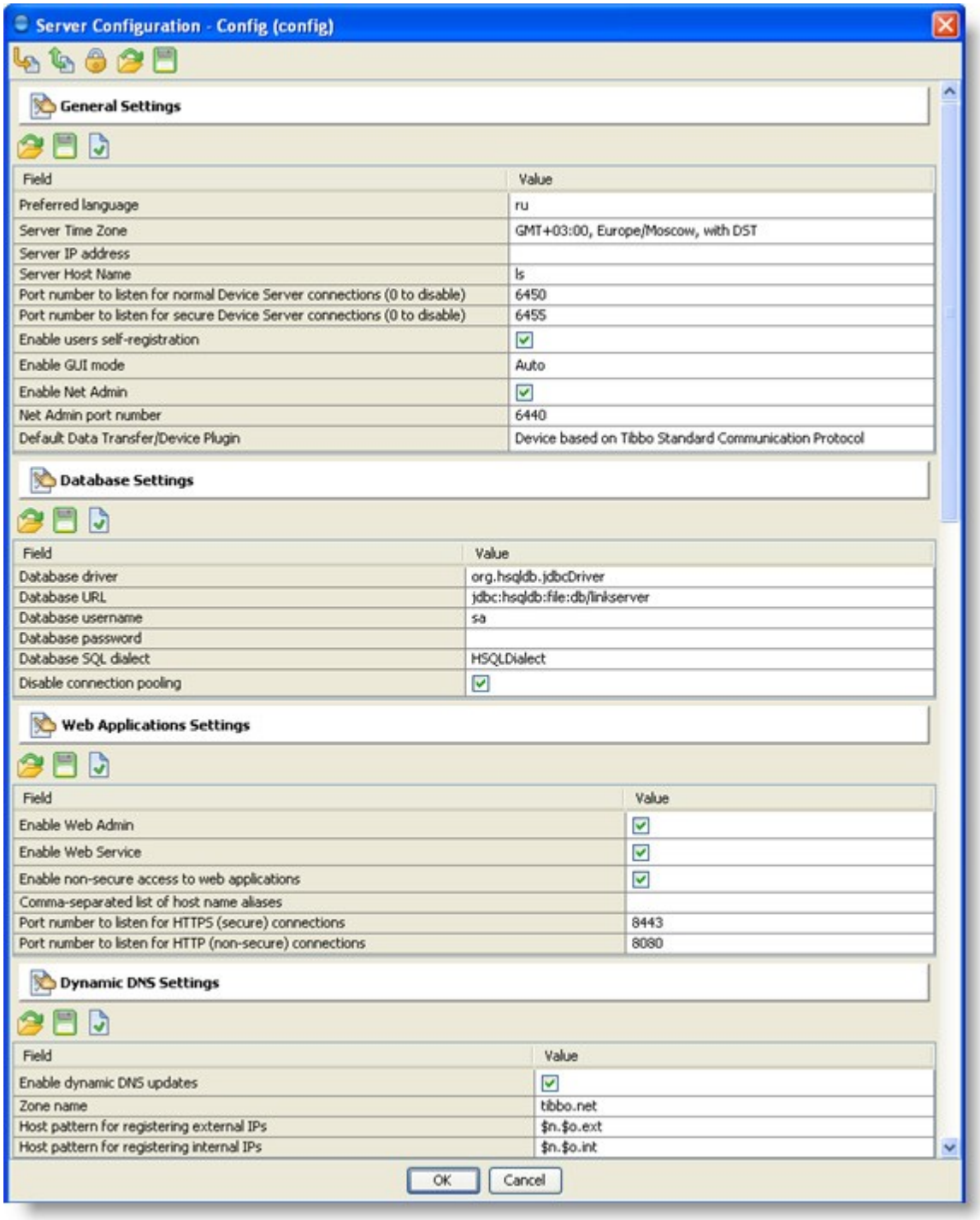
Утилита конфигуратора сервера принимает опцию одной командной строки: путь конфигурационного файла сервера. Если эта опция пропущена, редактируется файл по умолчанию (`server.xml`).

6.1.2 Настройка из клиента

Когда запущен AtomMind Server, можно с легкостью получить доступ к основным настройкам из AtomMind Client 359, запустив Конфигурировать сервер 1559 ( ) из контекстного меню для Узла сервера 376 (в Системном дереве) 370. Дополнительную информацию можно найти здесь 1559.

После того, как вы сохраните новые настройки, конфигуратор сервера запросит подтверждение, а затем перезагрузит AtomMind Server. Некоторые настройки критичны для системы, и если в них внести неправильные значения, это может воспрепятствовать запуску AtomMind Server после перезагрузки. В этом случае конфигурирование через клиент станет невозможным. Вам придется вручную редактировать конфигурационный файл, чтобы исправить в нем ошибку (см. файловая конфигурация 178). Поэтому убедитесь, что вы понимаете, что делаете, когда редактируете основные настройки.

Так выглядит диалоговое окно с основными настройками AtomMind Server в AtomMind Client:



6.1.3 Настройка через файлы

Настройку через файлы конфигурации следует использовать, если AtomMind Server не запускается, а другие способы недоступны. Конфигурационный файл по умолчанию - `server.xml`. Альтернативное имя файла можно выбрать, использовав опцию командной строки 164 '`-f <filename>`'.

Обратите внимание, что AtomMind Server не только загружает конфигурационный файл во время старта, но и сохраняет его настройки во время остановки. Выполненные вручную изменения в файле во время работы AtomMind Server будут перезаписаны.

Только несколько свойств глобальной конфигурации AtomMind Server сохраняются в конфигурационный файл. Это в основном свойства, используемые до установление соединения с базой данной.

Об этом прямо говорится в документации по глобальным свойствам, хранящимся в конфигурационном файле. Значения всех остальных глобальных настроек сохраняются в базу данных 692 сервера.

Конфигурационный файл имеет формат XML (eXtensible Markup Language). Вот образец файла:

```
<config>
  <firstLaunch>false</firstLaunch>
  <previousVersion>31001</previousVersion>
  <locale>ru</locale>
  <deviceServerPort>6450</deviceServerPort>
  <deviceServerSslPort>6455</deviceServerSslPort>
  <usersSelfRegistration>true</usersSelfRegistration>
  <configGuiMode>auto</configGuiMode>
  <netAdminPort>6440</netAdminPort>
  <netAdminEnabled>true</netAdminEnabled>
  <databaseDriver>org.hsqldb.jdbcDriver</databaseDriver>
  <databaseUrl>jdbc:hsqldb:file:db/AtomMind Server</databaseUrl>
  ...
</config>
```

AtomMind Server создает файл автоматически, нет необходимости добавлять настройки вручную. При устранении ошибок вам, возможно, потребуется изменить значения некоторых настроек.

Подробное описание доступных настроек можно найти в разделе Общие настройки 179.

6.1.4 Параметры общих настроек

Эта глава об общих настройках AtomMind Server. Если общие настройки были изменены, сервер следует перезапустить для того, чтобы активировать новые настройки.

Общие настройки можно редактировать, используя Конфигуратор сервера 177, Web UI 220 или AtomMind Client 359. Дополнительную информацию о том, как получить доступ к этим настройкам, можно найти в разделе Конфигурирование AtomMind Server 177.

6.1.4.1 Общие настройки

Общие настройки - это свойство глобальной конфигурации сервера, которое определяет глобальные параметры AtomMind Server, не принадлежащие ни одной из подкатегорий. Поля этого свойства сопоставляются с файлом конфигурации сервера 178.

Описание экземпляра сервера

Имя переменной в конфигурационном файле: serverDescription

Тип значения: String

Возможные значения: любая печатная строка

Значение по умолчанию: текстовое описание экземпляра AtomMind Server.

Временная зона сервера

Имя переменной в конфигурационном файле: timezone

Тип значения: String

Возможные значения: список временных зон, поддерживаемых виртуальной машиной Java

Значение по умолчанию: временная зона устанавливается в ОС, под которой запущен AtomMind Server, или выбирается **GMT,** если временную зону ОС невозможно определить. За дополнительной информацией обращайтесь к разделу Временные зоны 911.

IP-адрес сервера

Имя переменной в конфигурационном файле: serverIp

Тип значения: String

Возможные значения: любой действительный IP-адрес или пустая строка

Значение по умолчанию: "" (пустое значение)

Если этот параметр приписан действительному IP адресу, AtomMind Server "слушает" входящие соединения с Device, AtomMind Client и Web UI 220 только с этого IP адреса. Этот параметр может быть полезен для запуска множественных экземпляров AtomMind Server на одном сервере с несколькими IP адресами. В этом случае каждый экземпляр AtomMind Server будет принимать входящие соединения на одних и тех же портах, но с разными IP адресами.

Задайте эти настройки только для непустого значения, если есть большая необходимость принимать соединения только на одном IP или сетевом интерфейсе. Например, если IP сервер установлен на 192.168.1.5, то AtomMind Client, запущенный на той же машине, не сможет подключиться к серверу, используя адрес **localhost**.

Другие системные сервисы, принимающие входящие сетевые соединения, такие как SNMP-уловитель 644, используют этот параметр, чтобы "слушать" соединения только на определенных IP адресах.

Этот IP адрес тоже прописывается как *целевой IP-адрес* для Внешних серверов устройств 2096, которые подключаются 2098 к AtomMind Server.

Имя хоста сервера

Имя переменной в конфигурационном файле: serverHostName

Тип значения: String

Возможные значения: любое действительное имя хоста или пустая строка

Значение по умолчанию: "" (пустое значение)

Если у этого параметра есть значение, его используют в качестве дополнительного псевдонима 354 встроенного веб-сервера.

Значение данного параметра либо не должно быть указано вообще, либо должно соответствовать DNS имени машины, на которой запущен AtomMind Server.

Кроме того, его используют и для доступа к Device Servers, работающим через HTTP прокси 2095.

Неверно указанное имя хоста сервера может стать причиной различных проблем сервера при включении и в процессе работы.

Разрешить пользователям выполнять регистрацию самостоятельно

Имя переменной в конфигурационном файле: usersSelfRegistration

Тип значения: Boolean

Возможные значения: true или false

Значение по умолчанию: true

Когда разрешена саморегистрация, каждый, кто имеет доступ к Web UI 220 или установленному AtomMind Client 359, может регистрировать учетную запись пользователя, войти в систему и использовать *AtomMind Server*. Когда эта опция выключена, лишь зарегистрированные пользователи с соответствующим разрешением могут добавлять учетные записи новых пользователей 484.

Режим GUI

Имя переменной в файле конфигурации: configGuiMode

Тип значения: String

Возможные значения: да, нет, авто

Значение по умолчанию: авто

Эта опция определяет, будет ли AtomMind Server выполнять отрисовку GUI (графического интерфейса пользователя). GUI включает: *всплывающее окно программы* в начале запуска, иконку 173 в *Системном трее* и окна с сообщениями об ошибке. Когда эта опция выключена, отрисовка GUI осуществляться не будет.

Номер порта клиента

Имя переменной в конфигурационном файле: clientPort

Тип значения: Integer

Возможные значения: 1-65535

Значение по умолчанию: 6460

Эта опция определяет порт, на котором принимаются соединения клиентов.

Включить незащищенную связь клиента

Имя переменной в конфигурационном файле: nonSecureClientCommunicationEnabled

Тип значения: Boolean

Возможные значения: true или false

Значение по умолчанию: false

Позволяет клиентам соединяться с AtomMind Server с помощью простых сокетов без SSL шифрования. Обычно отключается из соображений безопасности.

Незащищенный порт клиента

Имя переменной в конфигурационном файле: nonSecureClientPort

Тип значения: Integer

Возможные значения: 1-65535

Значение по умолчанию: 6461

Эта опция определяет порт, на котором принимаются незазищенные соединения клиентов.

максимальная длина очереди исходящих событий

Имя переменной в конфигурационном файле: clientEventQueueLength

Тип значения: Integer

Значение по умолчанию: 100000

Эта опция определяет длину очереди клиента (события, которые ожидают отправки в клиент). Когда число событий в очереди достигает данного значения, Ошибка доставки события возникает в корневом контексте сервера. Все последующие события будут отбрасываться до тех пор, пока размер очереди не уменьшится.

Включить Net Admin

Имя переменной в конфигурационном файле: netAdminEnabled

Тип значения: Boolean

Возможные значения: true или false

Значение по умолчанию: true

Эта опция позволяет активировать или деактивировать локальный доступ через Telnet к интерфейсу *Net Admin* 1448 . Рекомендуется отключить эту опцию, если у сомнительных пользователей есть доступ (физическии или через командный интерпретатор) к хосту с AtomMind Server.

Номер порта Net Admin

Имя переменной в конфигурационном файле: netAdminPort

Тип значения: Integer

Возможные значения: 1-65535

Значение по умолчанию: 6440

Эта опция определяет порт, на котором принимаются соединения с *Net Admin* 1448.

Время ожидания HTTP сессии

Имя переменной в конфигурационном файле: httpSessionTimeout

Тип значения: Long

Возможные значения: any

Значение по умолчанию: 3600000

Таймаут для сессий HTTP (в миллисекундах). Используется встроенным веб сервером 354.

Папка данных статистики

Имя переменной в конфигурационном файле: statisticsFolder

Тип значения: String

Возможные значения: любое разрешённое имя папки

Значение по умолчанию: статистика

Абсолютный или относительный путь к папке, которая используется для хранения данных канала статистики 718.

6.1.4.2 Базы данных

База данных - это свойство глобальной конфигурации сервера, которое определяет, как AtomMind Server сохраняет данные в свою базу данных 692. Поля этого свойства сопоставляются с файлом конфигурации сервера 178.

Персонализация хранилища

Настройки в этой секции определяют, как хранить конфигурации устройств, события и двоичные данные.

Доступно пять режимов:

Значение	Описание
0	**Неактивный**. Данные не будут храниться.
1	**Реляционная база данных**. Данные будут храниться в реляционной БД. Для конфигурации базы данных используйте настройки раздела "Реляционная база данных".
2	**Хранилище ключ-значение**. Данные будут храниться в хранилище ключ-значение. Для конфигурации базы данных используйте настройки раздела "Хранилище ключ-значение".
3	**Хранилище NoSQL**. Данные будут храниться в хранилище NoSQL. Для конфигурации базы данных используйте настройки раздела "Хранилище NoSQL".
4	**Файловое хранилище**. Данные будут храниться в файлах в двоичном формате.

ХРАНИЛИЩЕ КОНФИГУРАЦИЙ

Имя переменной в конфигурационном файле: databaseConfigurationStorage

Тип значения: Integer

Возможные значения: 1, 2, 3, 4

Значение по умолчанию: 3

Эта настройка определяет, где хранятся конфигурации устройств.

ХРАНИЛИЩЕ СОБЫТИЙ

Имя переменной в конфигурационном файле: databaseEventHistoryStorage

Тип значения: Integer

Возможные значения: 0, 1, 3

Значение по умолчанию: 3

Эта настройка определяет, где хранятся события системы.

ХРАНИЛИЩЕ ДВОИЧНЫХ ДАННЫХ

Имя переменной в конфигурационном файле: databaseBinaryDataStorage

Тип значения: Integer

Возможные значения: 1, 2, 4

Значение по умолчанию: 4

Эта настройка определяет, где хранятся двоичные данные.

реляционная база данных

ВКЛЮЧИТЬ КЛАСТЕРИЗАЦИЮ БАЗЫ ДАННЫХ

Имя переменной в конфигурационном файле: databaseCluster

Тип значения: Boolean

Возможные значения: true или false

Значение по умолчанию: false

Если эта опция не включена, AtomMind Server работает с одной БД, т.е. речь идет о ее "классическом" поведении.

Если опция включена, сервер работает с несколькими базами данных и реплицирует все операции записи в каждую БД, а также распределяет нагрузку между всеми операциями чтения. В этом случае:

- Параметр URL базы данных 183 не активирован.
- Таблица баз данных кластеров 184 активирована, что позволяет конфигурировать базы данных, включенные в кластер.

Более подробно см. Отказоустойчивый кластер БД 710.

ДРАЙВЕР БАЗЫ ДАННЫХ

Имя переменной в конфигурационном файле: databaseDriver

Тип значения: String

Возможные значения: любое имя *Java* -класса, относящееся к драйверу JDBC

Значение по умолчанию: `org.apache.derby.jdbc.EmbeddedDriver`

Эта опция определяет, какой драйвер базы данных JDBC (*Java* Database Connectivity) будет использован. Технически это имя основного Java-класса драйвера. Например, для хранения данных в MySQL установите эту опцию в `com.mysql.jdbc.Driver`. Для определения нужного значения обратитесь к документации по драйверу JDBC.

Чтобы разрешить AtomMind Server загружать сторонние драйверы базы данных JDBC, файл JAR (*Java* Archive), содержащий этот драйвер, должен быть добавлен к *classpath* сервера в поддиректории `/jar` установочной директории AtomMind Server'a, или же можно использовать Файл свойств загрузчика 162 AtomMind Server.

URL БАЗЫ ДАННЫХ

Имя переменной в конфигурационном файле: databaseUrl

Тип значения: String

Возможные значения: строка пути, зависящая от версии БД

Значение по умолчанию: `jdbc:derby:database;create=true`

Это строка, зависящая от версии БД, которая определяет тип базы данных, путь системного файла (хранящегося локально или в сети) к базе данных, содержащей таблицы данных AtomMind Server и любые дополнительные настройки. Чтобы определить правильное значение для выбранного JDBC-драйвера базы данных, пожалуйста, обратитесь к его документации. Значение по умолчанию для этой опции заставляет AtomMind Server использовать встроенную базу данных Apache Derby для сохранения данных в обычных текстовых файлах в папке `database/` установочной директории AtomMind Server'a.

БАЗЫ ДАННЫХ В КЛАСТЕРЕ

Имя переменной в конфигурационном файле: N/A (отсутствует), значение хранится в Конфигурационном файле кластера БД 715

Тип значения: Data Table

Возможные значения: N/A

Значение по умолчанию: N/A

Таблица баз данных кластера позволяет просматривать статус и настраивать все базы данных в отказоустойчивом кластере БД 713, используемом AtomMind Server. Каждая база данных в кластере конфигурируется согласно следующим настройкам:

- **ID базы данных**. Уникальный определенный пользователем строковый идентификатор базы данных в кластере.
- **URL базы данных**. Адрес базы данных. Дополнительную информацию можно найти в разделах URL базы данных 183 (настройки для некластеризованной БД) и заметки об особенностях БД 698.
- **Вес**. Чем больше вес у базы данных в кластере, тем больше запросов на чтение она получит.
- **Локальный**. Этот флаг следует установить, если база данных располагается на одном и том же сервере, что и экземпляр AtomMind Server.

Сохранение баз данных кластера. При сохранении таблицы баз данных кластера сервер выполняет несколько процедур для новых и измененных записей БД:

- Каждая только что добавленная база данных активируется 714, т.е. соединяется с рабочим кластером
- Тестируется соединение с каждой базой данных
- Инициируется синхронизация данных между новой/изменённой базой данных и другими базами данных в кластере

Ручное редактирование баз данных кластера. Базы данных кластера хранятся в конфигурационном файле кластера БД 715. Возможно редактировать файл напрямую 715, без использования конфигуратора сервера 177.

ИМЯ ПОЛЬЗОВАТЕЛЯ БД

Имя переменной в конфигурационном файле: databaseUsername

Тип значения: String

Возможные значения: любое имя пользователя, подходящее для сервера БД

Значение по умолчанию: sa

Эта опция определяет, какое имя пользователя использовать для регистрации на сервере базы данных. Значение по умолчанию позволяет соединиться со встроенным в AtomMind Server сервером БД.

ПАРОЛЬ БАЗЫ ДАННЫХ

Имя ключа в файле конфигурации: databasePassword

Тип значения: String

Возможные значения: любой пароль, подходящий для сервера базы данных

Значение по умолчанию: "" (пустое значение)

Эта опция определяет, какой пароль используется при регистрации на сервере базы данных. Значение по умолчанию позволяет подключаться к встроенному в AtomMind Server серверу БД.

SQL ДИАЛЕКТ БАЗЫ ДАННЫХ

Имя переменной в конфигурационном файле: databaseSqlDialect

Тип значения: String

Возможные значения:

Значение	Сервер БД

Cache71Dialect	InterSystems Cache
DB2Dialect	DB2
DB2400Dialect	DB2 AS/400
DB2390Dialect	DB2 OS390
DerbyDialect	Apache Derby (v10.7 or above)
FirebirdDialect	Firebird
FrontbaseDialect	FrontBase
H2Dialect	H2
HSQLDialect	Hypersonic SQL
InformixDialect	Informix
IngresDialect	Ingres
InterbaseDialect	Interbase
JDataStoreDialect	JDataStore
MckoiDialect	Mckoi SQL
MimerSQLDialect	Mimer SQL
MySQL5InnoDBDialect	MySQL 5
MySQLInnoDBDialect	MySQL
OracleDialect	Oracle (старая версия)
Oracle9Dialect	Oracle 9/10g
Oracle 10g/11g	Oracle 10g/11g
PointbaseDialect	Pointbase
PostgreSQLDialect	PostgreSQL
ProgressDialect	Progress
SAPDBDialect	SAP DB
SQLServerDialect	Microsoft SQL Server
Sybase11Dialect	Sybase 11
SybaseDialect	Sybase
SybaseAnywhereDialect	Sybase Anywhere

Значение по умолчанию: DerbyTenSevenDialect

Эта опция определяет имя *Java*-класса для диалекта SQL базы данных. Например, следует использовать MySQLDialect, если Вы используете для хранения базу данных MySQL. Если Ваш сервер базы данных не указан в приведенной выше таблице, пожалуйста, обратитесь в службу технической поддержки.

МИНИМАЛЬНЫЙ РАЗМЕР ПУЛА СОЕДИНЕНИЙ

Имя переменной в конфигурационном файле: databaseMinimumPoolSize

Тип значения: Integer

Возможные значения: 1 или более

Значение по умолчанию: 3

Минимальное количество соединений с базой данных в пуле.

МАКСИМАЛЬНЫЙ РАЗМЕР ПУЛА СОЕДИНЕНИЙ

Имя переменной в конфигурационном файле: databaseMaximumPoolSize

Тип значения: Integer

Возможные значения: 1 или более

Значение по умолчанию: 200

Максимальное количество соединений с базой данных в пуле.

Максимальное число соединений в пуле должно быть меньше максимально позволенного числа соединений в настройках базы данных. В противном случае возможно снижение скорости работы или возникновение неожиданных ошибок БД.

ТАЙМАУТ ПОЛУЧЕНИЯ СОЕДИНЕНИЯ ИЗ ПУЛА

Имя переменной в конфигурационном файле: databaseCheckoutTimeout

Тип значения: Long

Возможные значения: 0 или более

Значение по умолчанию: 30000

Определяет, как долго (в миллисекундах) будет ждать сервер для соединения с каждой базой данных, полученной при групповом соединении. Нулевое значение обозначает неопределенный срок ожидания. Нулевое значение рекомендуется для производственной среды, а значение по умолчанию определяет относительно короткий лимит времени для быстрого определения проблемы в базе данных при внедрении системы.

ТАЙМАУТ ПОТЕРЯННОГО СОЕДИНЕНИЯ

Имя переменной в конфигурационном файле: databaseUnreturnedConnectionTimeout

Тип значения: Long

Возможные значения: 0 или более

Значение по умолчанию: 0

Определяет, как долго соединение может оставаться прерванным. При установленном ненулевом значении, прерванные соединения, превышающие лимит таймаута, будут удалены, а затем заменены в пуле соединения. Убедитесь, что для этого параметра установлено достаточно большое значение, чтобы хватало времени для завершения всех запланированных операций с потерянными соединениями.

Используйте данный параметр только в качестве временного решения при работе с ненадежными приложениями на базе AtomMind, которые выдают ошибку при отключении соединения. Изменение значения данной опции на ненулевое поможет предотвратить утечки соединения в производственных средах.

РАЗМЕР ПАКЕТА (НОЛЬ ДЛЯ ОТКЛЮЧЕНИЯ ГРУППОВЫХ ОБНОВЛЕНИЙ)

Имя переменной в конфигурационном файле: databaseBatchSize

Тип значения: Integer

Возможные значения: 0 или более

Значение по умолчанию: 50

Максимальное количество запросов в пакете обновления данных, например, размер пакета JDBC2.

НЕ ИСПОЛЬЗОВАТЬ ГРУППОВЫЕ ПОДКЛЮЧЕНИЯ

Имя переменной в конфигурационном файле: databaseDisablePooling

Тип значения: Boolean

Возможные значения: true или false

Значение по умолчанию: false

Когда флаг включен, все ошибки соединения базы данных протоколируются 166. Это помогает найти и устранить неисправность с подключением к БД, например, при переключении на новый сервер базы данных 698.

Эту опцию следует включать только временно, когда необходимо найти и устранить неполадку, поскольку она сильно снижает производительность сервера.

СВЯЗНОСТЬ

Тип значения: Reference

Возможные значения: N/A

При нажатии запускается тестовое подключение к базе данных с использованием текущих настроек. Если настройка кластера БД активна, все БД кластера будут тестироваться независимо друг от друга с использованием текущего имени пользователя и пароля.

Эта опция недоступна в свойстве Конфигуратор Сервера.

Хранилище ключ-значение

РОЛЬ В КЛАСТЕРЕ

Имя переменной в конфигурационном файле: databaseKvClusterRole

Тип значения: Integer

Возможные значения: 0 для None, 1 для Master и 2 для Failover

Значение по умолчанию: 0

Определяет роль узла ключ-значение в отказоустойчивом кластере БД.

Эта настройка может повлиять на работу AtomMind Server. Роль кластера БД ключ-значение должна быть идентична роли кластера AtomMind Server, за исключением конфигурации кластера модели Active / Active. Также, переключение БД в режим кластера включает в себя переключение в режим транзакций, который влияет на производительность. Обратное переключение не возвращает БД в нетранзакционный режим. IP адрес БД берется из параметра IP адрес сервера.

Для сброса группы репликации до единственного участника при открытой реплицированной среде, установите Роль в кластере на None.

ПОРТ ОСНОВНОЙ БАЗЫ ДАННЫХ

Имя переменной в конфигурационном файле: databaseKvClusterPrimaryDbPort

Тип значения: Integer

Возможные значения: Порт должен быть вне диапазона "Хорошо известные порты" (от нуля до 1023).

Значение по умолчанию: 5001

Порт, используемый данным узлом. Используется для взаимодействия узлов отказоустойчивого кластера. IP адрес БД берется из параметра IP адрес сервера.

АДРЕСА БАЗ ДАННЫХ В КЛАСТЕРЕ КЛЮЧ-ЗНАЧЕНИЕ

Имя переменной в конфигурационном файле: databaseKvClusterHelperUrl

Тип значения: String

Возможные значения: hostname[:port][,hostname[:port]]*

Значение по умолчанию:

Определяет другие хранилища ключ-значение в отказоустойчивом кластере по парам "адрес:порт". Номер порта должен соответствовать **Порту первичной базы данных**, определенной на других узлах.

Внимание! Если есть несколько знаков сети на других узлах, убедитесь, что показанные IP-адреса те же самые, что и выбранные другими узлами базы данных.

ПРИОРИТЕТ БАЗЫ ДАННЫХ В КЛАСТЕРЕ

Имя переменной в конфигурационном файле: databaseKvClusterPriority

Тип значения: Integer

Возможные значения: 0 и больше

Значение по умолчанию: 100

Приоритет текущего узла отказоустойчивого кластера. Узел с более высоким приоритетом будет избран мастером кластера. Нулевой приоритет означает, что узел никогда не станет мастером кластера.

ОБЪЕМ ПАМЯТИ

Имя переменной в конфигурационном файле: databaseKvCacheSize

Тип значения: Integer

Возможные значения: 0 и больше

Значение по умолчанию: 100 Мб

Настраивает объем доступной для хранилища памяти в байтах.

МИНИМАЛЬНЫЙ ПРОЦЕНТ АКТУАЛЬНЫХ ДАННЫХ В БД

Имя переменной в конфигурационном файле: confDatabaseKvMinUtilization

Тип значения: Integer

Возможные значения: от 0 до 50

Значение по умолчанию: 30

Определяет процент файлового пространства журнала базы данных, которое должно быть использовано для используемых записей. Если процент пространства, используемого записями, слишком низок, то процесс очистки базы данных удаляет устаревшие записи, пока этот порог не будет достигнут.

КОЛИЧЕСТВО ПОТОКОВ ОЧИСТКИ

Имя переменной в конфигурационном файле: databaseKvCleanerThreads

Тип значения: Integer

Возможные значения: от 1 до Integer.MAX_VALUE

Значение по умолчанию: 5

Количество потоков, выделенных очисткой для обработки файла журнала. Если очередь очистки становится большой, попробуйте увеличить это значение.

АКТИВАТОР ОЧИСТКИ

Имя переменной в конфигурационном файле: databaseKvCleanerActivator

Тип значения: String

Возможные значения: databaseKvCleanerByteThreshold или databaseKvCleanerWakeupInterval

Значение по умолчанию: databaseKvCleanerWakeupInterval

Определяет, когда запускается очистка - при превышении порогового значения использования диска, либо периодически.

ПОРОГ АКТИВАЦИИ ОЧИСТКИ

Имя переменной в конфигурационном файле: databaseKvCleanerByteThreshold

Тип значения: Integer

Возможные значения: от 1 до Integer.MAX_VALUE

Значение по умолчанию: 40 Мб

Проверяет использование диска каждый раз, когда пользователь записывает в журнал многобайтовую информацию.

ПЕРИОД АКТИВАЦИИ ОЧИСТКИ

Имя переменной в конфигурационном файле: databaseKvCleanerWakeupInterval

Тип значения: Long

Возможные значения: от 0 до Long.MAX_VALUE

Значение по умолчанию: 0

Проверяет, необходима ли очистка, если в течение данного периода времени не было произведено операций записи.

ИНТЕРВАЛ ЗАПИСИ ТРАНЗАКЦИЙ

Имя переменной в конфигурационном файле: databaseKvTransactionsCommitInterval

Тип значения: Integer

Возможные значения: от 0 до Integer.MAX_VALUE

Значение по умолчанию: 0

Временной интервал, в течение которого транзакции могут быть сгруппированы для амортизации стоимости записи.

РАЗМЕР ГРУППЫ ТРАНЗАКЦИЙ

Имя переменной в конфигурационном файле: databaseKvTransactionsCommitThreshold

Тип значения: Integer

Возможные значения: от 0 до Integer.MAX_VALUE

Значение по умолчанию: 0

Пороговое значение влияет на количество транзакций, которые могут быть сгруппированы для уменьшения затрат на запись. Только для транзакционного режима.

МАКСИМАЛЬНЫЙ РАЗМЕР СООБЩЕНИЯ ПРИ РЕПЛИКАЦИИ

Имя переменной в конфигурационном файле: confDatabaseKvMaxReplicationMessageSize

Тип значения: Long

Возможные значения: от 262144 до Long.MAX_VALUE

Значение по умолчанию: 1048576

Максимальный размер сообщения (в байтах), который будет принят этим узлом. Используется для предотвращения DOS атак.

АКТИВАТОР СОЗДАНИЯ КОНТРОЛЬНЫХ ТОЧЕК

Имя ключа в файле конфигурации: databaseKvCheckpointerActivator

Тип значения: String

Возможные значения: databaseKvCheckpointerWakeupInterval или databaseKvCheckpointerWriteInterval

Значение по умолчанию: databaseKvCheckpointerWakeupInterval

Режим активации контрольных точек.

ИНТЕРВАЛ ВРЕМЕНИ СОЗДАНИЯ КОНТРОЛЬНЫХ ТОЧЕК

Имя ключа в файле конфигурации: databaseKvCheckpointerWakeupInterval

Тип значения: Long

Возможные значения: от 0 до Long.MAX_VALUE

Значение по умолчанию: 3000

Интервал создания контрольной точки в микросекундах, т.е. временной интервал между двумя соседними записями.

ИНТЕРВАЛ ДАННЫХ СОЗДАНИЯ КОНТРОЛЬНЫХ ТОЧЕК

Имя переменной в конфигурационном файле: databaseKvCheckpointerWriteInterval

Тип значения: Long

Возможные значения: от 1 до Integer.MAX_VALUE

Значение по умолчанию: 20000000

Количество незаписанных данных, которое активирует создание контрольной точки.

МАКСИМАЛЬНЫЙ ПРЕДЕЛ ИСПОЛЬЗОВАНИЯ ДИСКА

Имя переменной в конфигурационном файле: databaseKvMaxDisk

Тип значения: Long

Возможные значения: от 0 до Long.MAX_VALUE

Значение по умолчанию: 2147483648

Верхний предел количества байтов, используемых для хранения данных. Если лимит превышен, операции записи будут запрещены. Если установлено значение ноль, ограничение не применяется.

ПРЕДЕЛ СВОБОДНОГО ДИСКОВОГО ПРОСТРАНСТВА

Имя переменной в конфигурационном файле: databaseKvFreeDisk

Тип значения: Long

Возможные значения: от 0 до Long.MAX_VALUE

Значение по умолчанию: 100000000

Нижний предел количества байтов свободного пространства для хранения на томе. Если лимит превышен, операции записи будут запрещены. Если установлен ноль, ограничение свободного места не применяется. Это не рекомендуется.

NoSQL хранилище

ИСПОЛЬЗОВАТЬ ВСТРОЕННУЮ СЛУЖБУ

Имя переменной в конфигурационном файле: databaseCassandraUseEmbeddedService

Тип значения: Boolean

Возможные значения: true или false

Значение по умолчанию: true

Когда этот параметр включен, встроенная служба Cassandra будет запущена на AtomMind Server и будет использоваться в качестве основной базы данных. Если установлен параметр IP адрес сервера 179, он будет использоваться в качестве встроенного сервиса Cassandra, в противном случае он будет рассчитан автоматически

Если опция не активирована, укажите опцию Адрес сервера 190 для подключения к внешней БД Cassandra.

ИСПОЛЬЗОВАТЬ ВНЕШНИЙ YAML-ФАЙЛ КОНФИГУРАЦИИ

Имя переменной в конфигурационном файле: databaseCassandraUseYamlConfiguration

Тип значения: Boolean

Возможные значения:true или false

Значение по умолчанию: false

Когда этот параметр включен, встроенная служба Cassandra будет использовать настройки не из конфигурации AtomMind Server (по умолчанию - файл `server.xml`), а из отдельного файла `cassandra.yaml` в домашней папке сервера. Это может быть полезно для более тонкой настройки встроенной службы Cassandra, когда необходимо изменить параметры БД, не доступные из конфигурации сервера.

АДРЕС СЕРВЕРА

Имя переменной в конфигурационном файле: databaseCassandraHost

Тип значения: String

Возможные значения: любой действительный IP адрес

Значение по умолчанию: `""` (пусто)

IP адрес сервера Cassandra.

ПОРТ СУБД

Имя переменной в конфигурационном файле: databaseCassandraNativePort

Тип значения: Integer

Возможные значения: 0 или больше

Значение по умолчанию: 9042

Порт, используемый сервером Cassandra для связи с клиентом.

ПУТЬ К ПАПКЕ ХРАНИЛИЩА

Имя ключа в файле конфигурации: databaseCassandraStorageDirectory

Тип значения: String

Возможные значения: любой действительный путь

Значение по умолчанию: "" (пусто)

Путь к папке хранилища основных данных пространства ключей. Если путь не указан, директория будет размещена в папке установки AtomMind.

ПУТЬ К ПАПКЕ COMMITLOG

Имя переменной в конфигурационном файле: databaseCassandraCommitlogDirectory

Тип значения: String

Возможные значения: любой действительный путь

Значение по умолчанию: "" (пусто)

Путь к папке, где хранятся commitlogs. Если путь не указан, директория будет размещена в папке установки AtomMind.

ПУТЬ К ПАПКЕ С КЭШЕМ

Имя переменной в конфигурационном файле: databaseCassandraCachesDirectory

Тип значения: String

Возможные значения: любой действительный путь

Значение по умолчанию: "" (пусто)

Путь к папке, где хранятся кэшированные данные. Если путь не указан, директория будет размещена в папке установки AtomMind.

ПРОСТРАНСТВО КЛЮЧЕЙ ХРАНИЛИЩА КОНФИГУРАЦИЙ

Имя переменной в конфигурационном файле: databaseCassandraConfigurationKeyspace

Тип значения: String

Возможные значения: Любое действительное имя пространства ключей.

Значение по умолчанию: aggregate

Пространство ключей, используемое Cassandra для хранения конфигураций. Используется для гарантии корректной работы единичного экземпляра ДБ Cassandra с несколькими серверами AtomMind.

ПРОСТРАНСТВО КЛЮЧЕЙ ХРАНИЛИЩА СОБЫТИЙ

Имя переменной в конфигурационном файле: databaseCassandraEventHistoryKeyspace

Тип значения: String

Возможные значения: Любое действительное имя пространства ключей.

Значение по умолчанию: aggregate

Пространство ключей, используемое Cassandra для хранения истории событий. Используется для гарантии корректной работы единичного экземпляра ДБ Cassandra с несколькими серверами AtomMind.

ПРОСТРАНСТВО КЛЮЧЕЙ ХРАНИЛИЩА ДВОИЧНЫХ ДАННЫХ

Имя переменной в конфигурационном файле: databaseCassandraBinaryDataKeyspace

Тип значения: String

Возможные значения: Любое действительное имя пространства ключей.

Значение по умолчанию: aggregate

Пространство ключей, используемое Cassandra для хранения двоичных данных. Используется для гарантии корректной работы единичного экземпляра ДБ Cassandra с несколькими серверами AtomMind.

ПРОСТРАНСТВО КЛЮЧЕЙ ХРАНИЛИЩА СТАТИСТИКИ

Имя переменной в конфигурационном файле: databaseCassandraStatisticsKeyspace

Тип значения: String

Возможные значения: Любое действительное имя пространства ключей.

Значение по умолчанию: aggregate

Пространство ключей, используемое Cassandra для хранения статистики 718. Используется для гарантии корректной работы единичного экземпляра ДБ Cassandra с несколькими серверами AtomMind.

ВНУТРЕННЕЕ ПРОСТРАНСТВО КЛЮЧЕЙ

Имя переменной в конфигурационном файле: databaseCassandraInternalKeyspace

Тип значения: String

Возможные значения: Любое действительное имя пространства ключей.

Значение по умолчанию: aggregate

Пространство ключей, используемое Cassandra для хранения системных данных AtomMind Server. Используется для гарантии корректной работы единичного экземпляра ДБ Cassandra с несколькими серверами AtomMind.

ФАКТОР РЕПЛИКАЦИИ

Имя переменной в конфигурационном файле: databaseReplicationFactor

Тип значения: Integer

Возможные значения: 1 или больше

Значение по умолчанию: 1

Фактор репликации описывает, как много копий данных будет сделано. Установите это значение больше одного, если вы хотите разделить данные между узлами кластера NoSQL.

СИДЫ КЛАСТЕРА NOSQL

Имя переменной в конфигурационном файле: databaseSeeds

Тип значения: String

Возможные значения: Список IP адресов, разделенных запятой

Значение по умолчанию: "" (пусто)

Список узлов, отвечающих за хранение и обработку конфигурации кластера NoSQL.

РАЗМЕР ПАКЕТА ДАННЫХ

Имя переменной в конфигурационном файле: databaseCassandraBatchSize

Тип значения: Integer

Возможные значения: 0 или выше

Значение по умолчанию: 0

Максимальное количество изменений в обновлении пакета. Нулевое значение деактивирует обновления пакета.

ПРЕДЕЛЬНЫЙ РАЗМЕР ПАМЯТИ ПАКЕТА

Имя ключа в файле конфигурации: databaseCassandraBatchSizeThreshold

Тип значения: Integer

Возможные значения: 0 или выше

Значение по умолчанию: 100000

Максимальный размер обновления пакета в байтах.

ИСПОЛЬЗОВАТЬ АУТЕНТИФИКАЦИЮ

Имя переменной в конфигурационном файле: databaseCassandraUseAuthentication

Тип значения: Boolean

Возможные значения: true или false

Значение по умолчанию: false

Контролирует, используется ли аутентификация для соединений с кластером Cassandra.

ИМЯ ПОЛЬЗОВАТЕЛЯ

Имя ключа в файле конфигурации: databaseCassandraLogin

Тип значения: String

Возможные значения: любое имя пользователя, подходящее для БД Cassandra

Значение по умолчанию: "" (пусто)

Имя пользователя для входа в хосты Cassandra.

ПАРОЛЬ

Имя переменной в конфигурационном файле: databaseCassandraPassword

Тип значения: String

Возможные значения: любей пароль, подходящий для БД Cassandra

Значение по умолчанию: "" (пусто)

Пароль для входа в хосты Cassandra.

РАЗМЕР COMMITLOG

Имя переменной в конфигурационном файле: databaseCassandraCommitlogSize

Тип значения: Integer

Возможные значения: 8,16,32

Значение по умолчанию: 32

Общее пространство, используемое под commitlogs. Меньший размер commitlog, как правило, увеличивает количество сбрасываний на диск в наименее активных таблицах.

УРОВЕНЬ СОГЛАСОВАННОСТИ

Имя переменной в конфигурационном файле: databaseCassandraConsistencyLevel

Тип значения: String

Возможные значения: ONE, TWO, THREE, QUORUM, ALL, LOCAL_QUORUM, EACH_QUORUM, LOCAL_ONE

Значение по умолчанию: LOCAL_ONE

Уровень согласованности Cassandra.

КОНТАКТНЫЕ ТОЧКИ

Имя переменной в конфигурационном файле: databaseCassandraContactPoints

Тип значения: String

Возможные значения: Список IP адресов или названий хостов, разделенных запятой

Значение по умолчанию: ""

Контактные точки Cassandra, т.е. IP адреса узлов.

ПОЛИТИКА БАЛАНСИРОВКИ НАГРУЗКИ

Имя переменной в конфигурационном файле: databaseCassandraLoadBalancing

Тип значения: Data Table

Возможные значения: Round Robin, Latency Aware, DC Aware Round Robin

Значение по умолчанию: Round Robin

Политика балансировки нагрузки БД Cassandra.

ПОЛИТИКА ПЕРЕПОДКЛЮЧЕНИЯ

Имя переменной в конфигурационном файле: databaseCassandraReconnectionPolicy

Тип значения: Data Table

Возможные значения: Экспоненциальное или постоянное для Политики переподключения, 1 или более миллисекунд для Начальной задержки, 1 или более миллисекунд для Максимальной задержки, 1 или более миллисекунд для Задержки

Значение по умолчанию: Экспоненциальное

Политика пепреподключения БД Cassandra.

ТАЙМАУТ ЗАПРОСА НА ЧТЕНИЕ

Имя переменной в конфигурационном файле: databaseCassandraReadRequestTimeout

Тип значения: Integer

Возможные значения:

Значение по умолчанию: 30000

Таймаут чтения БД Cassandra на хост в миллисекундах.

Устанавливая данное значение, учитывайте следующее:

- Значение не должно превышать настройки таймаута, используемые на стороне БД Cassandra
- Таймаут запроса на чтение приблизительный и контролирует только таймаут для одного хоста БД Cassandra, а не для запроса в целом

6.1.4.3 Кластер

Кластер - это свойство глобальной конфигурации сервера, отвечающее за функционирование данного AtomMind Server в составе отказоустойчивого кластера 1326. Поля этого свойства сопоставляются с файлом конфигурации сервера 178.

Роль в кластере

Имя переменной в конфигурационном файле: clusterRole

Тип значения: целое число

Возможные значения: 0 для Отсутствует, 1 для Главная и 2 для Дублирующая

Значение по умолчанию: 0

Определяет роль установки AtomMind Servera в отказоустойчивом кластере.

Режим дублирования

Имя переменной в конфигурационном файле: clusterFailoverReadonly

Тип значения: логическое значение

Возможные значения: true или false

Значение по умолчанию: false

Определяет, работает ли дублирующий узел в режиме "только для чтения", т.е. когда операции записи в базу данных запрещены.

Время обнаружения отказа узла

Имя переменной в конфигурационном файле: clusterFailureDetectionTime

Тип значения: длинное целое

Возможные значения: от 4000 до 86400000

Значение по умолчанию: 20000

Время в миллисекундах, необходимое для узлов кластера, чтобы отметить определенный узел "отключенным", если он не выполняет обновления базы данных.

- Дублирующие узлы активируются самостоятельно и переключатся на Ведущий режим, если ведущий узел будет неактивен дольше времени обнаружения отказа узла
- Ведущий узел сгенерирует предупреждающее событие, если дублирующие узлы не среагируют по истечении времени обнаружения отказа узла

Порт Heartbeat

Имя переменной в конфигурационном файле: clusterHeartbeatPort

Тип значения: Целочисленное

Возможные значения: 0 или более

Значение по умолчанию: 7800

Количество порта для прослушивания других узлов кластера heartbeat

Heartbeat адреса других узлов

Имя переменной в конфигурационном файле: clusterHeartbeatHelperUrls

Тип значения: Целочисленное

Возможные значения: Список IP адресов с номерами портов, разделенными запятыми, например, `address:port, address:port`.

Значение по умолчанию: `""` (пусто)

IP-адрес интерфейса Heartbeat

Имя переменной в конфигурационном файле: clusterHeartbeatInterfaceAddress

Тип значения: Строка

Возможные значения: Интерфейс, чтобы слушать, когда heartbeat включен. Если у устройства сервера есть больше, чем один интерфейс сети, должен быть выбран определенный интерфейс.

Значение по умолчанию: Первый найденный IP-адрес интерфейса сети

6.1.4.4 Сервер лицензий

Сервер лицензий - это свойство глобальной конфигурации сервера, позволяющее экземпляру данного AtomMind Server вадилировать его лицензии через Сервер Лицензий. Поля этого свойства сопоставляются с файлом конфигурации сервера 178.

Более подробно о централизованном управлении лицензиями см. в разделе Сервер лицензий 145.

активация сервера лицензий

Имя переменной в конфигурационном файле: licenseServerEnabled

Тип значения: Boolean

Возможные значения: true или false

Значение по умолчанию: false

Определяет, активирован ли Сервер Лицензий. Когда опция отключена, система будет искать файл с лицензией (`server.license`) из корневого каталога своего сервера.

Адрес

Имя переменной в конфигурационном файле: licenseServerAddress

Тип значения: String

Возможные значения: любой действительный IP адрес, либо пустая строка

Значение по умолчанию: "" (пустая строка)

Определяет IP адрес сервера, к котором будет подключен ваш текущий сервер для получения лицензии. В случае ошибки соединения систему также будет искать файл с лицензией из корневого каталога своего сервера.

порт

Имя переменной в конфигурационном файле: licenseServerPort

Тип значения: Integer

Возможные значения: 1-65535

Значение по умолчанию: 6460

Определяет порт сервера, к которому будет подключен ваш текущий сервер для получения лицензии.

Login

Имя переменной в конфигурационном файле: licenseServerLogin

Тип значения: String

Возможные значения: любая строка

Значение по умолчанию: отсутствует

Определяет логин для аутентификации пользователя.

пароль

Имя переменной в конфигурационном файле: licenseServerPassword

Тип значения: String

Возможные значения: любая строка

Значение по умолчанию: отсутствует

Определяет пароль для аутентификации пользователя

6.1.4.5 Обработка событий

Эта глава о настройках обработки событий в AtomMind Server.

6.1.4.5.1 Правила обработки событий

Таблица правил обработки событий определяет:

- какие события будут отклонены без обработки, маршрутизации или хранения;
- как система обнаруживает дубликаты событий и агрегирует их, уменьшая общее количество событий;
- как долго хранимые события 73 будут оставаться в истории событий 75.

Формат таблицы правил обработки событий:

Маска контекстов	Каждая запись в таблице определяет правила обработки события в каждом контексте, соответствующем данной маске 44. Например, если мы устанавливаем **время хранения**

	1 месяц для события "login" (в столбце **событие**) и устанавливаем **маску контекстов** в значение "users.admin" (соответствует контексту пользователь 1608 **администратора**), информация о логинах системного администратора будет храниться в течение одного месяца. Если **маска контекстов** установлена в значение "users.*", успешные логины *всех* пользователей будут храниться в течение выбранного периода. Если маска не задана, (т.е. NULL), правило применимо к событиям во всех контекстах.
Событие	Имя события, чьи правила обработки определяются этой записью.
Префильтр	Выражение 112, используемое для отклонения событий, которые не соответствуют определенным критериям. Отклоненные события не сохраняются в базе данных, не передаются слушателям и не обрабатываются никакими модулями системы. При заданном выражении префильтра, события будут отклонены, если выражение возвращает FALSE. Среда выполнения 114 выражения префильтра: • Контекст по умолчанию 119 — Контекст события. • Таблица данных по умолчанию 120 — Таблица данных, содержащая данные для разных типов событий 74. • Ряд по умолчанию 119 — 0 • Переменные среды 123 — Только стандартные 123 переменные.
Выражение идентификатора дедупликации	Выражение 112, используемое для расчета *идентификаторов дедупликации* событий. Выражение преобразуется в строку. Если очередь событий в оперативной памяти (чей размер определяется параметром **количество событий для хранения в оперативной памяти**) содержит событие с тем же идентификатором дедупликации, текущее событие обрабатывается так же, как его дубликат: • Число дубликатов предыдущего события увеличится на 1. • Время создания предыдущего события будет обновлено до соответствия времени создания дубликата. Таким образом, сохраняется время создания самого последнего дубликата. • Событие будет отправлено слушателям только при включенном параметре **диспетчеризация дубликатов.**
Количество событий для хранения в оперативной памяти	Количество событий для хранения в памяти и использования в целях поиска дубликатов согласно **выражению идентификатора дедупликации**.
Диспетчеризация дубликатов	Определяет, будет ли событие с обнаруженными в оперативной памяти дубликатами в дальнейшем маршрутизировано и доставлено подписчикам (слушателям).
Время хранения	Продолжительность постоянного хранения. Продолжительность по умолчанию - 100 дней, после чего события навсегда удаляются из истории. Можно специально настроить продолжительность хранения ("время существования") для различных событий. Заметим, что на практике события могут сохраняться в истории немного дольше определенного этими настройками срока, потому что процесс очистки (удаления событий с истекшим сроком хранения) запускается с интервалом в несколько минут. Заданное время хранения, равное нулю, блокирует хранение событий, определенных правилом.
Обогащения	Таблица, определяющая, какие обогащения 76 добавляются к событиям, определенным правилом. Таблица содержит два столбца: • **имя** обогащения • **выражение** значения обогащения Как только появляется вновь созданное событие, соответствующее указанному правилу, каждое **выражение** значения преобразуется в строку. К экземпляру события

	добавляется новое обогащение с **именем** из данной таблицы и значение, равное результату **выражения**.

Среда выполнения 114 выражения обогащения:	
Контекст по умолчанию 119	Контекст события.
Таблица данных по умолчанию 120	Таблица данных, содержащая данные для разных типов событий 74.
Ряд по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

При возникновении события AtomMind Server обрабатывает таблицу правил обработки события от первой до последней записи. Если контекст, в котором произошло событие, соответствует маске, определенной записью, сервер применяет определяемые этой записью параметры для хранения события и прекращает поиск по списку, т.е. применяется первое соответствие, даже если маски в строках ниже также соответствуют контексту. Таким образом, правила с более узкими масками контекста должны предшествовать правилам с более широкими масками.

Если контекст не соответствует ни одной из масок, определенных в таблице, для хранения и обработки события будут применены параметры по умолчанию, определенные в дескрипторе события.

Пример: допустим, имеется следующая таблица правил обработки событий:

Маска контекстов	Событие	Время хранения
users.admin	login	1000 часов
users.*	login	200 часов

Согласно этой таблице, AtomMind Server будет хранить логины пользователя "admin" в течение 1000 часов, логины всех других пользователей будут храниться 200 часов.

Пример: допустим, имеется следующее правило обработки событий:

Маска контекстов	Собы-тие	Префильтр	Выражение идентификатора дедупликации	Количество событий для хранения в оперативной памяти	Время хранения
users.admin.devices.virtual1	event1	{int} < 10	{int}	10	3 месяца

Согласно этой таблице, AtomMind Server будет обрабатывать событие **event1**, поступившее с определенного виртуального устройства 654 через путь контекста **users.admin.devices.virtual1**. Это событие имеет в том числе целочисленное **int** поле. Упомянутое правило будет работать следующим образом:

- Все события, имеющие значение поля **int** большее или равное 10, будут отклонены;
- Для остальных событий, будет рассчитан идентификатор дедупликации. Этот идентификатор будет соответствовать значению поля **int**.
- Согласно правилу выше, события с одинаковым значением поля **int** будут отфильтровываться как дубликаты первого события с таким значением поля **int**.
- Как только количество событий в памяти превысит **10**, самые старые события удалятся и больше не будут участвовать в поиске дубликатов. При этом копии этих событий, хранящиеся в базе данных (и сохраняющие информацию о дубликатах), останутся нетронутыми.

6.1.4.5.2 Настраиваемые самописцы событий

Таблица настраиваемых записей событий конфигурирует то, какие события[73] будут храниться в настраиваемых назначениях, таких как БД или CSV файл в дополнение к обычной хронологии событий[73].

Формат таблицы настраиваемых записей событий:

Тип хранилища	Тип настраиваемого хранилища. В настоящий момент поддерживается два типа: **database** и **CSV**-файл.
Имя	Имя хранилища. Использование имени зависит от типа хранилища. Например, это имя БД для хранилища БД или имени файла (без расширения) при использовании файлового хранилища CSV.
Маска контекста	Маска[44] контекстов для отслеживания событий, которые необходимо сохранить в настраиваемом хранилище.
Имя события	Имя хранимого события.

Настраиваемые записи событий

Обычно AtomMind Server хранит исторические события в базе данных[692] в своем собственном формате. Таблица событий (имеющая имя `ag_events`) полностью управляется сервером и не должна быть доступна из сторонних приложений.

Настраиваемое хранилище событий БД предоставляет метод для хранения избранных событий в специальной таблице. Эта таблица имеет несколько отличий от стандартного хранилища событий:

- События добавляются в таблицу по мере их возникновения. AtomMind Server не удаляет устаревшие события из этой таблицы. Никаких дополнительных настроек существующих событий не производится.
- Сервер создает специальную колонку в таблице для каждого поля, описанного в формате[73] события.

Любое стороннее приложение может осуществлять выборку запросов модификации в этой таблице для получения доступа к событиям AtomMind Server. Возможен доступ к данной таблце даже во время работы AtomMind Server.

Структура настраиваемой таблицы событий:

Поле	Тип	Комментарии
ag_id	Длинное (bigint)	ID события
ag_datetime	Временная отметка (datetime)	Временная отметка события
ag_context	Строка (varchar)	Контекст события
ag_event	Строка (varchar)	Имя события
ag_level	Целое (int)	Уровень[73] события (числовое значение)
ev_*	Другой тип	Настраиваемые специфичные поля

Хранилище событий CSV

Это хранилище добавляет избранные события в CSV (Character-Separated Values) файл. Это простой текстовый файл, который может быть интерпретирован любым приложением.

Поля CSV-файла:

- ID события
- Временная отметка события (отформатированная согласно образцу[2146] `yyyy-MM-dd HH:mm:ss.SSS`)
- Контекст события
- Имя события
- Уровень[73] события (числовое значение)
- Настраиваемые специфичные поля

6.1.4.5.3 Хранилища событий

Таблица хранилищ событий описывает логические разделы, используемые для хранения различных хронологических событий[73]. Настоящая реализация хранилищ событий зависит от типа базы данных сервера[692]. Например, каждое хранилище события - это таблица в реляционной БД или пространство ключей в БД NoSQL.

AtomMind Server создает, удаляет и настраивает хранилища событий автоматически. В большинстве случаев создание или изменение хранилищ событий вручную не требуется.

Параметры хранилища событий:

Таблица	Имя хранилища событий. Обычно соотносится с именем подходящих данных БД.
Контекст	Путь или маска контекстов событий, которые должны быть в хранилище.
Событие	Имя события экземпляров, которые должны быть в хранилище.
Привязки	Постоянные привязки[735], которые будут вычислены, когда событие сохраняется. Каждая привязка определяет целевую *ключевую колонку*, которая будет хранить данные события, такие как поле события таблицы данных. Значения ключевых полей могут быть использованы, чтобы загрузить события из БД, используя критерии пользовательского поиска.  **Пример:** Хранилище, которое содержит события изменения[84], имеет постоянную привязку, которая хранит имя измененной переменной в отдельной ключевой колонке. Это позволяет загрузить изменения только выборочной переменной, вместо загрузки всех событий изменения в контексте. Свойства постоянной привязки: **Имя** — Имя ключевого поля. **Тип** — Тип ключевого поля. **Индекс** — Имя индекса БД для создания ключевого поля. При значении null индекс не будет создан. **Выражение** — Выражение[112], которое возвращает значение ключевого поля. Вычисляется в момент сохранения события. Среда вычисления[114] выражения постоянной привязки: Контекст по умолчанию[119] — Контекст события. Таблица данных по умолчанию[120] — Таблица данных, которая содержит данные состояния[74]. Ряд по умолчанию[119] — 0 Переменные среды[123] — Только стандартные[123] переменные.
Сохранить контекст	Определяет, должен ли контекст события быть в хранилище.
Сохранить имя	Определяет, должно ли имя события быть в хранилище.
Сохранить время хранения	Определяет, должно ли время хранения события быть в хранилище.
Сохранить уровень	Определяет, должен ли уровень события быть в хранилище.
Сохранить права доступа	Определяет, должны ли права доступа к событию быть в хранилище.
Сохранить количество	Определяет, должно ли количество ограничителей события быть в хранилище.
Сохранить подтверждение	Определяет, должно ли подтверждение события быть в хранилище.
Сохранить дополнения	Определяет, должны ли дополнения события быть в хранилище.

Сохранить формат	Определяет, должен ли формат события быть в хранилище.
Сохранить данные	Определяет, должны ли данные события быть в хранилище.

6.1.4.6 Безопасность

Эта глава о настройках безопасности AtomMind Server.

6.1.4.6.1 Настройки паролей

Опции в этой группе описывают правила, которым должен следовать каждый пользователь AtomMind Server при создании или изменении пароля.

Обратите внимание, что эти настройки хранятся в базе данных, а не в файле конфигурации.

Минимальная длина пароля

Тип значения: целочисленное

Возможные значения: 0 или больше

Значение по умолчанию: 0

Минимальное количество символов, которые должен содержать пароль. Игнорируется, когда значение равно 0.

Время действия пароля (в днях)

Тип значения: целочисленное

Возможные значения: 0 или больше

Значение по умолчанию: 0

Описывает, как долго будет актуален новый пароль . После того, как срок действия пароля истекает, система автоматически предлагает изменить пароль. Игнорируется, когда значение равно 0.

Требование наличия заглавных букв

Тип значения: логического типа

Возможные значения: true или false

Значение по умолчанию: false

Укажите, должен ли пароль содержать заглавные буквы.

Требование наличия чисел

Тип значения: логического типа

Возможные значения: true или false

Значение по умолчанию: false

Укажите, должен ли пароль содержать числа.

Требование наличия специальных символов

Тип значения: логического типа

Возможные значения: true или false

Значение по умолчанию: false

Укажите, должен ли пароль содержать специальные символы.

Специальные символы:

!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	:	;	<	=	>	?	@	[	\	]	^	_	`	{	|	}	~

количество запрещенных предыдущих паролей

Тип значения: Integer

Возможные значения: 0 or more

Значение по умолчанию: 0

Количество ранее использованных паролей, которые будут храниться в профиле пользователя. Система запрещает пользователю устанавливать использованный ранее пароль.

6.1.4.6.2 Права доступа по умолчанию

Это свойство определяет права доступа 477 по умолчанию и доступность ресурсов для всех новых пользователей 478 AtomMind Server. Оно используется при построении таблицы прав 488 и таблице ресурсов 482 при создании нового пользователя. Каждая запись в Правах Доступа по Умолчанию определяет наличие определенного контекста 41 контейнера и доступ к этому контейнеру:

- Еслии флаг **Доступен** является непроверяемым в какой-либо записи, контейнер ресурсов в этой записи не будет доступен пользователям AtomMind Servera.
- Если флаг **Разрешен** является непроверяемым в какой-либо записи, новый пользователь не получит доступ к ресурсам, указанным в этой записи. При этом другие пользователи смогут иметь к ним доступ.

Уровень 486 доступа к ресурсам, определяемый из Прав Доступа по Умолчанию при регистрации пользователя, может быть изменен позднее редактированием таблицы прав 487 данного пользователя.

6.1.4.6.3 Расширенные права доступа для новых пользователей

Это свойство определяет список дополнительных настроек, которые будут добавлены в таблицу прав 488 новых пользователей 478 AtomMind Servera. Используется для создания таблицы прав доступа при создании новой учетной записи. Таблица содержит два поля:

- **Маска** 44 **контекста**, определяющая контексты, к которым применима данная запись
- **Уровень** 486 **доступа** данного пользователя к вышеупомянутому контексту

Маска контекста может содержать спецсимвол %. Он будет заменен на имя создаваемого пользователя во время создания записи.

Пример: Если **маска контекста** таблицы расширенных прав доступа содержит строку **users.%.alerts** и имя создаваемого пользователя **john**, его таблица прав доступа будет содержать запись **users.john.alerts**.

Таблица Расширенных Прав Доступа очень похожа на таблицу Прав Доступа По Умолчанию 202. Различие в том, что **Права Доступа По Умолчанию** позволяют быстро настроить права для новых пользователей, просто щелкая несколько чекбоксов, в то время как **Расширенные Права Доступа** предоставляют более тщательный контроль над правами на основе непосредственных данных записей в таблице прав пользователей.

Таблица Расширенных Прав Доступа имеет приоритет над таблицей Прав Доступа По Умолчанию 202. Записи из расширенной таблицы будут вноситься в таблицу прав нового пользователя **до** записей в Правах Доступа По Умолчанию.

6.1.4.6.4 Аутентификация

Опции в этой группе контролируют процесс аутентификации в системе.

Внешняя аутентификация

Имя раздела в файле конфигурации: externalAuthentication

Тип значения: String

Возможные значения: `Disabled` (другие значения зависят от активных плагинов 205)

Значение по умолчанию: `Disabled`

Контролирует внешнюю аутентификацию пользователей AtomMind Server 490, таких как аутентификация с помощью LDAP / Microsoft Active Directory.

URL внешней регистрации

Имя раздела в файле конфигурации: externalRegisterLink

Тип значения: String

Возможные значения: `Any valid URL`

Значение по умолчанию: `""`

URL, к которому будет перенаправлен пользователь для внешней регистрации через веб интерфейс. Использование этой настройки позволяет пользователю зарегистрироваться в сторонней системе, а затем применить опцию внешняя аутентификация 490 для получения доступа к AtomMind Server.

Режим пользовательских подключений

Имя раздела в файле конфигурации: userConnectionMode

Тип значения: Integer

Возможные значения: `0` для Разрешать одновременные подключения, `1` для Отсоединить последнее и `2` для Ограничить одновременные подключения

Значение по умолчанию: `0`

Определяет политики подключения разных пользователей.

- Режим **Разрешать одновременные подключения** позволяет вам входить в систему множество раз под одним пользователем
- Когда включен режим **Отсоединить последнее**, каждое новое подключение к серверу отключает старую сессию с этим же пользователем
- Режим **Ограничить одновременные подключения** позволяет войти в систему, используя пользователя, который уже подключен

Некоторые интерфейсы пользователя, созданные в Web UI 220, могут устанавливать более одного одновременного подключения с сервером. Например, каждый компонент веб виджета 348 поддерживает свое собственное подключение.

Такие интерфейсы с большим количеством подключений будут несовместимы с режимами **Отсоединить последнее** и **Ограничить одновременные подключения**.

Выражение активации нового пользователя

Имя раздела в файле конфигурации: newUserActivationExpression

Тип значения: String

Возможные значения: Expression

Значение по умолчанию: `Null`

Позволяет выполнить пользовательское выражение во время регистрации пользователя. Если результат вычисления `null`, то пользователь будет активирован, иначе аккаунт будет отключен. Контекст для выражения по умолчанию является контекстом нового пользователя.

число попыток входа в систему

Имя раздела в файле конфигурации: numberLoginAttempts

Тип значения: Integer

Возможные значения: `1 or more`

Значение по умолчанию: `3`

Допустимое количество неудачных попыток входа в систему до того, как аккаунт будет заблокирован на период времени, указанный в **Длительность блокировки аккаунта**.

длительность блокировки аккаунта

Имя раздела в файле конфигурации: accountLockoutDuration

Тип значения: Integer

Возможные значения: 1 or more

Значение по умолчанию: 30000

Период времени, в течение которого пользователю будет запрещен вход в систему после нескольких неудачных попыток входа.

Тайм-аут перед следующей попыткой входа

Имя раздела в файле конфигурации: timeoutBeforeNextLoginAttempt

Тип значения: Integer

Возможные значения: 1 or more

Значение по умолчанию: 5000

Минимальный период времени, через который будет разрешена попытка входа в систему. Если в этот период совершить попытку входа, она окажется неудачной даже в случае правильного ввода идентификационных данных.

6.1.4.6.5 Настройки саморегистрации

Настройки саморегистрации - это свойство глобальной конфигурации сервера, которое определяет видимые и обязательные поля для формы саморегистрации пользователя 484.

Значения полей

Каждое поле имеет **Описание**, значения **Видимость** а **Обязательно для заполнения**:

- **Описание -** имя поля.
- **Видимость -** значение определяет, отображается ли данное поле в форме саморегистрации пользователя.
- **Обязательно для заполнения** - значение определяет, отмечается ли поле в форме саморегистрации пользователя как обязательное, и можно ли завершить регистрацию без их заполнения.

Поля, определенные одновременно как невидимые и обязательные, обрабатываются как необязательные для заполнения.

Значения по умолчанию

Имя пользователя и **Пароль -** поля, обязательные по умолчанию.

поле Captcha

Пожалуйста, обратите внимание, что **CAPTCHA** требует установленного подключения AtomMind Server к интернету.

6.1.4.6.6 Шифрование

Опции в этой группе контролируют шифрование данных в базе данных 692 сервера.

Зашифровать защищенные поля внутренним ключом

Имя опции в конфигурационном файле: internalEncryptionEnabled

Тип значения: Boolean

Возможные значения: `true` или `false`

Значение по умолчанию: `false`

Определяет, должен ли AtomMind Server использовать внутренний ключ шифрования или внешние ключи шифрования, предоставляемые Сервером лицензий в рамках централизованного управления шифрованием 146.

Защищенные поля в базе данных не будут зашифрованы, если данная опция отключена, и Сервер лицензий не предоставил ключ шифрования для данного сервера.

6.1.4.7 Устройства

Эта глава предоставляет информацию о **настройках синхронизации по умолчанию** для соединенных устройств и помогает понять, как контролировать количество параллельных сессий вход/выход устройства группами устройств или для всех устройств, управляемых AtomMind с помощью **виртуальных сетей устройств**.

6.1.4.7.1 Опции синхронизации по умолчанию

Эти табличные свойства в целом используются, чтобы установить опции синхронизации 502 по умолчанию для настроек Device.

Эти опции используются для настроек заново добавленных устройств и для настроек, которые никогда не сохраняли свои опции синхронизации 502. Существующие устройства используют свои собственные копии опций синхронизации и не наследуют значения из глобальной таблицы.

6.1.4.7.2 Статистика

Статистика - это свойство глобальной конфигурации сервера, которое используется для централизованной настройки каналов статистики 718 для новых устройств.

Эти каналы создаются для новых устройств и могут пересоздаваться при перезагрузке драйвера устройства 1495. Уже существующие устройства используют собственные копии настроек каналов статистик 719 и не наследуют изменения настроек из этой общей таблицы.

6.1.4.7.3 Сети виртуальных устройств

Эта таблица конфигурирует Сети виртуальных устройств 517, используемые AtomMind Server.

6.1.4.8 Активные плагины

Это свойство определяет, какие плагины 207 включены на сервере. Все плагины устанавливаются в каталог `/plugins` установочной директории AtomMind Server'a и по умолчанию включены.

Плагин не может быть отключен, если другие включенные плагины зависят от его работы.

Отключение плагинов может негативно повлиять на работу системы. Например, при отключении плагина драйвера устройств 518 будут отключены все Devices**, использующие этот драйвер.**

6.1.4.9 Магазин

Активные плагины - это свойство глобальной конфигурации сервера, определяющее настройки для магазинов приложений 132, которыми пользуется данный сервер.

Список магазинов

Данное свойство определяет список магазинов.

Имя свойства: **stores**

Тип свойства: **Data Table**

Каждая запись в это таблице определяет магазин:

- **Address**. Адрес магазина.
- **Port**. Имя порта магазина.
- **Description**. Имя магазина. Это имя будет отображаться в диалоговом окне выбора магазина.
- **Login**. Имя пользователя для авторизации на сервере магазина.
- **Password**. Пароль для авторизации.

Конфигурация

Эта группа содержит опции для работы с магазинами.

ПОКАЗЫВАТЬ МАГАЗИН ПРИ ЗАПУСКЕ

Этот флаг определяет, отображается ли диалог выбора магазина при запуске клиента.

Имя свойства: **showStoreOnStartup**

Тип свойства: **Boolean**

6.2 Продукты, плагины и ресурсы

Каждый экземпляр AtomMind приобретает ценность для бизнеса благодаря наследованию функционала, предоставляемого:

- **Продуктами**, или Решениями
- **Плагинами**, или Модулями
- **Ресурсами**, которые фактически являются преднастроенными Контекстами

Продукты

Продукты 1788, или Решения, разрабатываются для комплексного решения определенных бизнес-задач.

Пример: ТВЭЛ предлагает такие коробочные продукты, как Сетевое управление и мониторинг 1788 и SCADA/HMI 1971.

Технически, каждый продукт - это набор плагинов и связанных файлов.

Продукты могут корректировать базовую Платформу, если их включить в определенный установочный пакет AtomMind. Другой способ получить продукты - это установить их из магазина приложений 1321.

Установка нового продукта (решения) автоматически разворачивает все включенные в него плагины (модули), но позднее некоторые из этих плагинов можно отключить.

Плагины

Плагины - это встраиваемые модули AtomMind Server или AtomMind Client. Каждый плагин доставляет определенный функционал базовой Платформе, например, поддержку нового коммуникационного протокола или какой-либо метод анализа данных.

Пример: продукт Сетевое управление и мониторинг 1788 включает драйвер устройств SNMP 637, а также модуль (плагин) Мониторинг IP SLA 1934.

Каждый плагин расширяет кодовую базу Платформы, добавляя функционал, реализованный в коде на языке Java. Некоторые плагины могут также предоставлять новые Ресурсы, т.е. преднастроенные контексты.

Сразу после установки нового плагина, включенные в него ресурсы не *создаются*, т.е. не активируются автоматически. Чтобы эти ресурсы заработали, необходимо создать их вручную, как описано в разделе управление ресурсами 208.

Более подробно см. раздел Плагины 207.

Ресурсы

Ресурсы - это контексты 41, заранее настроенные для совместной работы в составе Продукта или Решения. Доступные типы ресурсов зависят от набора установленных Плагинов/Модулей. Ресурсами могу быть тревоги 779, модели 810, отчеты 928, инструментальные панели 912, фильтры событий 762, планировщики задач 823 и т.д.

Пример: продукт Сетевое управление и мониторинг 1788 включает инструментальную панель `Network Traffic Overview`, доступную как ресурс, включенный в плагин `Network Management`.

Ресурсы могут быть жестко закодированы в плагины (например, плагины, предлагаемые ТВЭЛ, или плагины, разработанные нашими партнерами с использованием комплекта разработки плагинов 1342 с открытым исходным кодом).

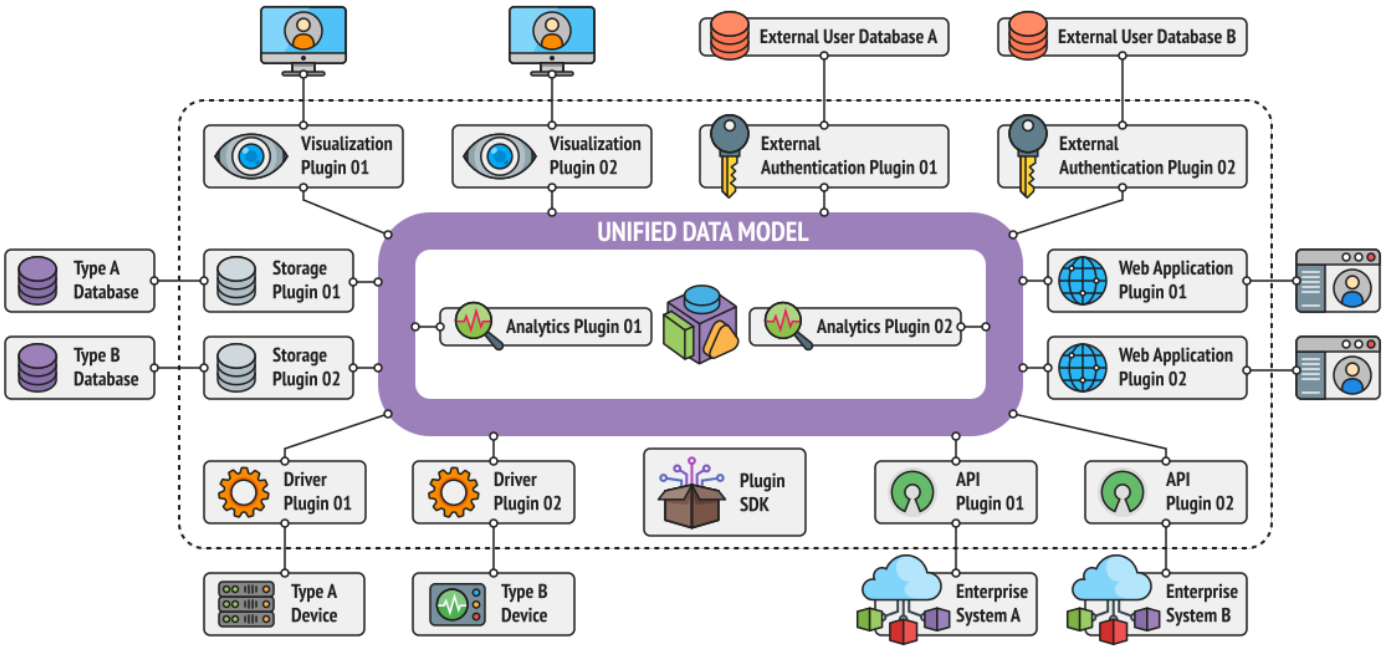
Ресурсы могу быть также упакованы в приложение 1318, что упрощает их распространение между серверами AtomMind.

Более подробно см. раздел Ресурсы 208.

6.3 Плагины

Некоторые функции AtomMind Server и AtomMind Client выполняются при помощи *плагинов*. AtomMind Server и AtomMind Client поддерживают несколько типов плагинов для взаимодействия с различными устройствами 497, добавления новых контекстов 41 в дерево контекстов сервера, активации новых возможностей хранения данных, реализации новых схем аутентификации пользователя и т.д.

Плагин - это модуль компьютерной программы, взаимодействующий с главным приложением для выполнения определенных, обычно очень специфичных, функций по запросу.



Все плагины загружаются при запуске сервера.

Изначальный набор доступных плагинов зависит от выбранного установочного пакета AtomMind Server или плана подписки на облачные услуги. Новые плагины можно установить из магазина приложений 1321.

настройка плагинов

Каждый плагин имеет до двух уровней настроек:

- **Глобальные настройки**. Влияют на поведение плагинов по умолчанию. Могут изменяться только пользователями с достаточными правами доступа.
- **Настройки уровня пользователя**. Влияют на поведение плагинов, только когда их действия относятся к определенной учетной записи пользователя 478.

У многих плагинов есть только глобальные настройки или вообще нет никаких настроек.

Администрирование плагинов

Для администрирования плагинов используются два контекста: общий контекст Конфигурация драйверов/плагинов 1505, который служит в качестве контейнера, и контекст Конфигурация драйвера/плагина 1506, который содержит конфигурацию только одного плагина.

Существуют два типа контейнеров конфигурации плагинов: *глобальный контейнер* для глобальных настроек и *личный пользовательский контейнер* для хранения настроек плагина, относящихся к определенной учетной записи пользователя 478.

Включение/Отключение плагинов

Можно включать или отключать отдельные плагины, используя опцию глобальной конфигурации сервера Активные плагины 205.

Ручная установка плагинов

Каждый плагин представляет собой отдельный файл. Файлы плагинов расположены в подпапке `/plugins` папки установки.

Чтобы установить новый плагин, просто скопируйте архив плагина в соответствующую подпапку в папке `/plugins` и перезапустите AtomMind Server. При установке новой версии существующего плагина, полученного от ТВЭЛ, перепишите архив нового плагина поверх старого.

6.4 Ресурсы

Плагины 207 и приложения 1318 AtomMind Server в основном состоят из *ресурсов,* таких как тревоги 779, модели 810, отчеты 928, инструментальные панели 912, фильтры событий 762, запланированные задачи 823 и другие инструменты, заранее настроенные для решения задач по аналитике и визуализации в конкретных отраслях прямо "из коробки". Ресурсы, включенные в плагины и приложения, называются *упакованными ресурсами*.

Например, наш продукт Сетевое управление и мониторинг включает инструментальную панель `Network Traffic Overview`, а продукт SCADA/HMI поставляется вместе с демонстрационным HMI виджетом `Filter Plant`.

Пакеты установки продуктов, разработанных партнерами ТВЭЛ на базе Платформы, обычно включают в себя плагины с собственными наборами ресурсов. Эти ресурсы смешиваются с ресурсами AtomMind в каждой отдельной продуктовой инсталляции.

В каждой инсталляции ресурсы могут *создаваться* по запросу.

Созданный упакованный ресурс становится контекстом 41 ресурса.

Как только упакованные ресурсы созданы, т.е. стали контекстами сервера, системные операторы могут вносить в них изменения. В то же время, новые версии продуктов содержат обновленные наборы ресурсов. При этом следует учитывать, что обновление любых ресурсов, измененных локально, приведет к конфликтам и потенциальной потере этих изменений.

Чтобы избежать этого, сервер AtomMind Server поддерживает большое количество способов управления ресурсами:

- **Группировка ресурсов**. Каждый ресурс входит в определенную группу (например, Управление сетью) и опционально может входить в подгруппу (например, Управление базами данных). Это облегчает поиск и выборку ресурсов во время операций по их созданию/обновлению/удалению.
- **Отслеживание изменений**. Как только ресурс меняется оператором локальной системы, он помечается и больше не обновляется автоматически при появлении более новой версии. Однако, можно обновить его вручную, минуя предупреждение о потере изменений, например, после создания клона.
- **Управление версиями ресурсов**. Каждый ресурс имеет свою историю изменений, помогающую локальным администраторам решать, нужно ли обновлять локальную копию.
- **Отслеживание зависимостей**. Если для создания выбирается инструментальная панель, включающая десять виджетов и журнал событий, операторам будут предоставлены подсказки по созданию всех зависимых ресурсов. Это гарантирует функциональную согласованность.

создание и удаление ресурсов

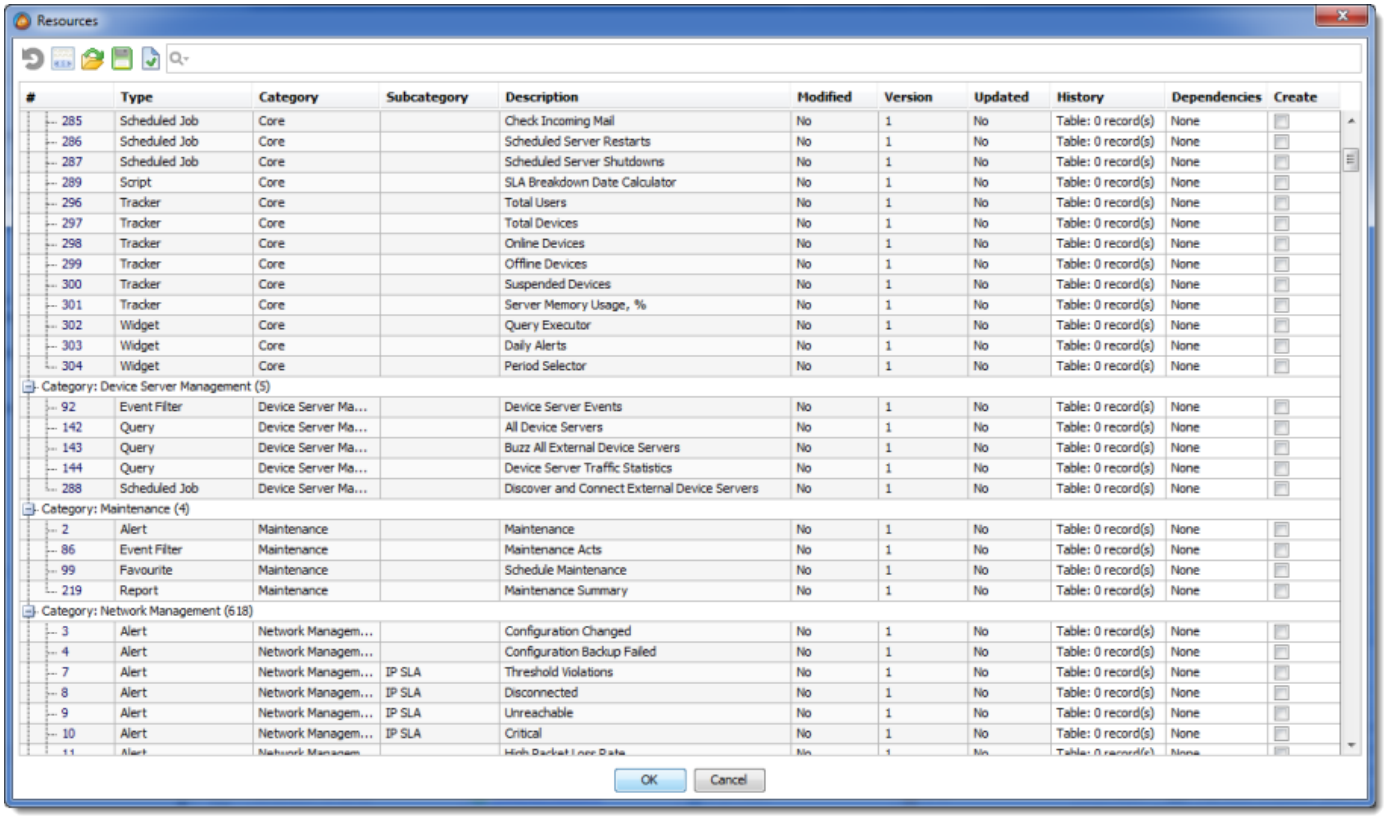
По умолчанию, некоторые встроенные ресурсы создаются при первом запуске сервера. Может понадобиться создать ресурсы вручную в следующих случаях:

- Если метод создания ресурса, выбранный при установке AtomMind Server, не привел к созданию всех ресурсов
- Если ресурс был по ошибке удален или поврежден некорректным редактированием
- Если AtomMind Server был обновлен, и доступны новые версии ресурсов

Если при установке AtomMind Server использован **режим создания ресурсов,** отличный от **None,** и все ресурсы были удалены, набор выбранных при установке ресурсов будет создан заново при запуске сервера.

СОЗДАНИЕ/ОБНОВЛЕНИЕ РЕСУРСОВ ВРУЧНУЮ

Чтобы создать ресурсы, предоставляемые ядром или плагинами AtomMind, запустите действие **Создать/Обновить ресурсы** ( ) из корневого контекста 1559. Вам будет предложено выбрать ресурсы для создания или обновления:



#	Type	Category	Subcategory	Description	Modified	Version	Updated	History	Dependencies	Create
285	Scheduled Job	Core		Check Incoming Mail	No	1	No	Table: 0 record(s)	None	
286	Scheduled Job	Core		Scheduled Server Restarts	No	1	No	Table: 0 record(s)	None	
287	Scheduled Job	Core		Scheduled Server Shutdowns	No	1	No	Table: 0 record(s)	None	
289	Script	Core		SLA Breakdown Date Calculator	No	1	No	Table: 0 record(s)	None	
296	Tracker	Core		Total Users	No	1	No	Table: 0 record(s)	None	
297	Tracker	Core		Total Devices	No	1	No	Table: 0 record(s)	None	
298	Tracker	Core		Online Devices	No	1	No	Table: 0 record(s)	None	
299	Tracker	Core		Offline Devices	No	1	No	Table: 0 record(s)	None	
300	Tracker	Core		Suspended Devices	No	1	No	Table: 0 record(s)	None	
301	Tracker	Core		Server Memory Usage, %	No	1	No	Table: 0 record(s)	None	
302	Widget	Core		Query Executor	No	1	No	Table: 0 record(s)	None	
303	Widget	Core		Daily Alerts	No	1	No	Table: 0 record(s)	None	
304	Widget	Core		Period Selector	No	1	No	Table: 0 record(s)	None	
Category: Device Server Management (5)										
92	Event Filter	Device Server Ma...		Device Server Events	No	1	No	Table: 0 record(s)	None	
142	Query	Device Server Ma...		All Device Servers	No	1	No	Table: 0 record(s)	None	
143	Query	Device Server Ma...		Buzz All External Device Servers	No	1	No	Table: 0 record(s)	None	
144	Query	Device Server Ma...		Device Server Traffic Statistics	No	1	No	Table: 0 record(s)	None	
288	Scheduled Job	Device Server Ma...		Discover and Connect External Device Servers	No	1	No	Table: 0 record(s)	None	
Category: Maintenance (4)										
2	Alert	Maintenance		Maintenance	No	1	No	Table: 0 record(s)	None	
86	Event Filter	Maintenance		Maintenance Acts	No	1	No	Table: 0 record(s)	None	
99	Favourite	Maintenance		Schedule Maintenance	No	1	No	Table: 0 record(s)	None	
219	Report	Maintenance		Maintenance Summary	No	1	No	Table: 0 record(s)	None	
Category: Network Management (618)										
3	Alert	Network Managem...		Configuration Changed	No	1	No	Table: 0 record(s)	None	
4	Alert	Network Managem...		Configuration Backup Failed	No	1	No	Table: 0 record(s)	None	
7	Alert	Network Managem...	IP SLA	Threshold Violations	No	1	No	Table: 0 record(s)	None	
8	Alert	Network Managem...	IP SLA	Disconnected	No	1	No	Table: 0 record(s)	None	
9	Alert	Network Managem...	IP SLA	Unreachable	No	1	No	Table: 0 record(s)	None	
10	Alert	Network Managem...	IP SLA	Critical	No	1	No	Table: 0 record(s)	None	

Список ресурсов может быть отсортирован, отфильтрован или сгруппирован по типам ресурсов, категориям и подкатегориям.

Как только ресурсы будут созданы, Вы сможете легко модифицировать их. Если какой-то ресурс был поврежден, и у вас нет его резервной копии, просто удалите этот ресурс и снова запустите действие Создать ресурсы, чтобы восстановить его.

УДАЛЕНИЕ РЕСУРСОВ

Возможно удалить встроенные ресурсы, как любые другие ресурсы AtomMind Server.

Но чтобы облегчить пакетное удаление нескольких ресурсов, используйте действие **Удалить ресурсы** ( ) из корневого контекста 1559.

Если выбранные для удаления ресурсы были модифицированы в местной инсталляции AtomMind Server, система выдаст оператору предупреждение о возможной потере этих изменений.

Управление версиями и параметры ресурсов

Система управления версиями ресурсов помогает:

- Обновлять локально созданные ресурсы, если они были обновлены в новой версии AtomMind Server
- Отслеживать изменения ресурсов, выполненные системными операторами
- Предотвращать потерю локальных изменений во время обновлений ресурсов
- Просматривать историю версий ресурсов

См. раздел Сохранение истории изменений объекта 1695 для информации о том, как сохранять и загружать историю изменений ресурсов, сделанных операторами системы.

Диалоговое окно создания ресурсов имеет несколько столбцов, относящихся к управлению версиями ресурсов:

- **Модифицировано**. Этот столбец показывает, был ли ресурс модифицирован оператором локальной системы. Если ресурс был модифицирован, его удаление или обновление до версии, идущей в комплекте с новым дистрибутивом AtomMind Server, вызовет потерю этих изменений!
- **Версия**. Показывает версию ресурса, идущую в комплекте с установленной версией AtomMind Server. Она может отличаться или совпадать с версией ресурса, созданной ранее в локальной инсталляции.
- **Обновлено**. Показывает, был ли обновлен ресурс, т.е. его версия, доступная в установленном AtomMind Server, новее, чем версия, созданная ранее в локальной инсталляции.
- **История**. Показывает историю версий ресурса и их описания.

Обновленные ресурсы, которые не были модифицированы локально, отбираются в первую очередь для обновления в диалоговом окне создания/обновления ресурса.

ЗАВИСИМОСТИ РЕСУРСОВ

В поле **Зависимости** перечислены ресурсы, необходимые для правильной работы соответствующего ресурса. Если для создания / обновления выбран ресурс с зависимостями, система предложит оператору автоматическое создание необходимых ресурсов. Если ресурсы из списка зависимостей не созданы, основной ресурс в большинстве случаев будет работать некорректно.

Пример: если для создания выбрана инструментальная панель 912 с множеством виджетов 943, система выдаст оператору подсказку создать эти виджеты. Если эти виджеты не были созданы, инструментальная панель откроется пустой.

6.5 Мониторинг состояния сервера

Являясь сервером приложений, AtomMind Server дает системным администраторам детальную информацию для отслеживания состояния сервера и приложений, работающих на нем.

Большая часть такой необработанной информации доступна через действие **Показать информацию о сервере** Корневого контекста 1559 и включает:

- Версию сервера, дату установки, время запуска, продолжительность непрерывной работы и запуска
- Использование памяти сервера, занятое место на диске и загрузку процессора
- Общую и детальную статистику обработки событий
- Окружение сервера Java
- Состояние Отказоустойчивого кластера AtomMind Server и Отказоустойчивого кластера БД
- Подробную статистику контекстов (количество сущностей контекста 41, число их использований, поконтекстную статистику обработки событий и пр.)
- Общую статистику устройств (количество операций чтения/записи переменных устройства, вызовов функций и полученных событий устройства)
- Информацию о лицензии

- Информацию об активных плагинах
- Информацию об установленных модулях 1321
- Подробную информацию об активных пользовательских соединениях
- Общую и детальную информацию о потоках и пулах потоков сервера

Подробную информацию о конкретных метриках сервера можно найти в описаниях соответствующих переменных Корневого контекста 1559.

Определенный набор диагностических инструментальных панелей доступен в категории ресурсов 208 **Диагностика системы**. Эти панели проводят подробный графический анализ изменений показателей производительности сервера во времени.

6.6 Использование памяти

С точки зрения операционной системы, лежащей в основе, AtomMind Server, как любое другое программное обеспечение на базе Java, **потребляет столько памяти, сколько есть в его распоряжении**. Никак не больше и редко меньше. Максимально разрешенное количество 163 определяется настройкой `-Xmx`, заданной в файле свойств запуска 162.

В большинстве случаев, инсталлятор AtomMind устанавливает это значение на 50% оперативной памяти, доступной на машине, где запущен AtomMind. Таким образом, если системный администратор проверит использование памяти сервера при помощи Windows Task Manager или Linux команды `top`, он, скорее всего, заметит, что объем занимаемой AtomMind Serverом памяти довольно высок.

Однако **это не имеет ничего общего с реальным использованием памяти!** Виртуальная машина Java (JVM), запускающая AtomMind Server, имеет собственную систему управления динамической памятью. Система довольно сложная, но если коротко, у нее есть периодический процесс под названием Сбор Мусора. Этот процесс очищает все временные файлы в памяти, позволяя видеть, сколько на самом деле памяти используется сервером в долгосрочной перспективе.

Единственный способ узнать объем реально потребляемой памяти сервера - это проделать последовательность двух операций:

- Выполнить действие Запустить очистку памяти 1559 из **Корневого** контекста, чтобы запустить полный цикл "сбора мусора" JVM
- Проверить значение поля Использование памяти, % от максимально допустимого 1567 в переменной **Статус сервера Корневого** контекста, например, выполнив действие Просмотреть информацию о сервере 1559 из **Корневого** контекста

Проверка параметра **Использование памяти, % от максимально допустимого** без предварительного запуска очистки памяти ("сбора мусора") покажет случайные значения в диапазоне от фактического использования памяти до 100%. Этот результат не пригоден для какой-либо интерпретации.

Если объем фактически используемой памяти сервера (сразу после цикла очистки) превышает 90%, Виртуальная машина Java сервера испытывает недостаток памяти, тратя больше времени на процесс сбора мусора и увеличивая нагрузку на ЦП.

Если JVM не может очистить достаточно памяти для продолжения нормальной работы сервера, она будет потреблять вплоть до 100% ЦП для процесса сбора мусора, даже до того, как зарегистрирует первую ошибку о нехватке памяти!

Таким образом, если фактическое использование памяти превышает 90% разрешенного максимума, значение `-Xmx` должно быть увеличено для эффективной работы сервера. Если новое значение `-Xmx` выше 70-80% размера доступной оперативной памяти, оперативную память сервера нужно расширить для обеспечения достаточного количества памяти для AtomMind Server, операционной системы и стороннего ПО.

И, наоборот, если фактическое потребление памяти довольно низкое, а сервер уже запущен в рабочем режиме, значение параметра `-Xmx` можно снизить для освобождения большего количества памяти (например, для базы данных сервера или других приложений). Например, если использование памяти равно 30%, значение `-Xmx` можно разделить на два, а самое низкое использование памяти поднимется до 60%`-Xmx`, что является достаточно безопасным.

поконтекстное использование памяти

Возможно также анализировать примерный размер динамической памяти JVM, удерживаемой каждым отдельным контекстом.

Поконтекстное использование памяти отображается в колонке **Удерживаемая память** таблицы статистики. Получить доступ к ней можно через действие Показать статистику контекстов 1564 корневого контекста.

6.7 Кастомизация платформы и приложений

AtomMind дает гибкие возможности для ребрендинга и реализации проектов, что позволяет нашим партнерам разрабатывать собственные продукты и сервисы. Эти возможности включают:

- **Замена фирменного оформления**. Все логотипы, авторские права, названия продукции, названия компании, ссылки на веб-сайты и другие отличительные черты компании можно изменить соответственно Вашему брэнду.
- **Реализация приложений**. Кастомизированная дистрибуция AtomMind может включать любое количество ресурсов (таких как тревоги 779, отчеты 928, фильтры событий 762 или виджеты 943), которые пакетируются в приложения 1318. Например, если вы создаете систему мониторинга водосборников, ваше приложение может включать тревогу **Обнаружен перелив** и отчет **Объем воды в резервуаре (почасовой)**. Приложения можно распространять через магазин 1321 ТВЭЛ или самого партнера.
- **Интернационализация**. Все текстовые ресурсы могут быть переведены на любой язык. Ваши приложения на базе платформы могут также включать пакеты ресурсов на многих языках.
- **Локализация**. Включает в себя правила форматирования даты/времени и других значений, специфичных для страны. Например, персонализированный пакет AtomMind может предопределить структуру даты/времени 2146.
- **Полная настройка пользовательского интерфейса**. Разработка UI по вашим макетам с точностью до пикселя. Нестандартные графические компоненты, темы и т.д.
- **Пользовательские плагины**. Пользовательские плагины AtomMind Server позволяют подключать новые умные устройства к системе, а также вводить новые контексты 41 сервера или добавлять новые действия 87 в существующие контексты.

Свяжитесь с ТВЭЛ для получения информации о доступных возможностях кастомизации.

6.8 Поиск и устранение неисправностей

Этот раздел содержит разъяснения и решения, как устранить возможные несложные проблемы различных компонентов AtomMind.

6.8.1 Проблемы установки AtomMind Server

В этом разделе описаны различные проблемы и ошибки в процессе установки AtomMind Server.

Ошибка установщика на RHEL 7.4

Если у вас ОС Red Hat Enterprise Linux 7.4, может возникнуть ошибка установщика программы. Обновление RHEL 7.4 включает пакет `stix-fonts`. Если такое обновление установлено, шрифт по умолчанию меняется с `utopia` на `stix`. Шрифты Java по умолчанию тоже относятся к `stix`, включая семейство шрифтов `sans-serif`. Библиотека Linux fontconfig разработана так, что помещает шрифты внутри системы и выбирает их в зависимости от требований приложений. Таким образом, Linux не распознает шрифт `sans-serif` и генерирует Исключение.

Для устранения ошибки установщика необходимо обновить JDK до версии 1.8.192 или выше. Если у вас нет возможности обновить ваш JDK, можно также установить шрифт `dejavu-serif`, чтобы решить проблему. Установить этот шрифт можно с использованием следующего скрипта:

```
yum install dejavu-serif-fonts
```

6.8.2 Проблемы запуска AtomMind Server

AtomMind Server может отображать сообщение об ошибке или же запускаться успешно, но не принимать входящие соединения от устройств или клиентов. Это означает, что возможно есть серьезные ошибки во время запуска сервера. Проверьте файл журнала AtomMind Servera, чтобы найти причину ошибки. Этот файл по умолчанию называется `server.log` и находится в директории инсталляции AtomMind Servera. См раздел Настройка журналирования 166.

Иногда критические ошибки, отображаемые в извещении об ошибке, вызваны небольшой проблемой, которую можно диагностировать при помощи файла журнала.

Лицензия повреждена

Сообщение об ошибке появляющется на AtomMind Server, запущенном на Linux или Mac OS без корневых полномочий. Для исправления данной ситуации перезапустите сервер с корневыми полномочиями.

Для Mac OS (с учетом того, что приложение установлено в папке /Application/AtomMind/):

```
sudo /Applications/AtomMind/am_server.app/Contents/MacOS/JavaApplicationStub
```

Для Linux:

```
su
service am_server_service start
```

Появление сообщения: "Не удалось запустить JVM. Главный метод мог сгенерировать исключение".

Это сообщение означает, что критическая ошибка произошла во время запуска JVM. Чтобы увидеть само сообщение об ошибке, которое позволит диагностировать проблему, используйте загрузчик 158 am_server_console , чтобы запустить AtomMind Server.

Для Windows окно консоли может исчезнуть раньше того, как Вы успеете прочитать сообщение. В этом случае следует запустить интерпретатор командной строки, чтобы активировать am_server_console.

Для этого следует перейти в *Start Menu* > *Run*, набрать cmd и кликнуть OK. Откроется окно интерпреторатора командной строки. Теперь наберите *AtomMind* /am_server_console , чтобы запустить модуль запуска.

Появление сообщения: "Невозможно зарезервировать достаточное пространство для кучи объектов"

Когда Вы попытаетесь запустить AtomMind Server, Вы можете увидеть следующее сообщение:

```
Error occurred during initialization of VM

Could not reserve enough space for object heap

The JVM could not be started. The main method may have thrown an exception.
```

Это означает, что Виртуальная машина Java не может назначить объем памяти, заданный параметров загрузчика - Xmx (максимальная память) 158 в одном непрерывном блоке. Существует несколько возможных решений:

- Уменьшить значение параметра -Xmx , если Ваш AtomMind Server не слишком загружен.
- Попытаться высвободить часть памяти путем отмены загрузки приложений, перезагружающих машину.
- Выделить дополнительную память компьютеру, на котором запущен AtomMind Server.
- На ПК с Windows некоторые приложения могут регистрировать собственные динамические библиотеки (DLLs), располагающиеся в памяти по определенному абсолютному адресу. Это делит память на несколько блоков и не позволяет JVM AtomMind Server назначать один крупный блок памяти. Чтобы решить эту проблему, следует использовать редактор реестра для удаления всех введенных данных с HKEY_LOCAL_MACHINE\Software\Microsoft\WindowsNT\CurrentVersion\Windows\AppInit_DLLs. Сначала следует сделать резервную копию!

Появление сообщения об ошибке запуска сервиса базы данных

Эта ошибка означает, что сервер не смог установить соединение с базой данных 692. Обычно основная причина ошибки содержится в сообщении об ошибке. Если основную причину нельзя определить после прочтения сообщения об ошибке запуска сервера, попробуйте выполнить следующее:

- Отключить пул соединений 186 с БД
- Запусить сервер и дождаться появления сообщения об ошибке
- Найти окончание файла журнала 166 сервера (server.log по умолчанию), чтоб определить основную причину проблем с БД.
- Если Вы используете сервер баз данных mysql, поставляющийся совместно с AtomMind, скорее всего не запустился сервис MySQL. Проверьте файлы AtomMind\mysql\data*.err на наличие ошибок. Поскольку сервер баз данных mysql, поставляющийся совместно с AtomMind, настроен на высокую производительность, он может быть неспособен назначать достаточный объем буферной памяти на некоторых ПК. В этом случае следует уменьшить значения параметров innodb_buffer_pool_size и innodb_buffer_pool_size в конфигурационном файле MySQL (AtomMind\mysql\my.ini).

Сервер был запущен, но клиенты не могут подключиться

Если AtomMind Server был запущен, но к нему невозможно подключиться при помощи AtomMind Client и Web UI, возможно, сервер все еще находится в процессе загрузки. Проверьте файл журнала сервера (`server.log`), чтобы узнать, не выполняет ли сервер какую-либо длительную операцию, такую как, например, обслуживание БД. Пожалуйста, свяжитесь с группой тех поддержки ТВЭЛ, если на загрузку сервера уходит больше 3-5 минут.

Неправильная версия Java (JRE/JDK) используется AtomMind Serverом для Linux

Откройте файл запуска AtomMind Server (это shell-скрипт) для редактирования и отмените преобразование строки `INSTALL4J_JAVA_HOME_OVERRIDE` в комментарии. Укажите правильный путь JRE/JDK в этой строке:

```
#!/bin/sh
# Uncomment the following line to override the JVM search sequence
INSTALL4J_JAVA_HOME_OVERRIDE=/usr/java/path
# Uncomment the following line to add additional VM parameters
```

6.8.3 Ошибки во время работы AtomMind Server

В этом разделе говорится о различных ошибках во время работы, которые можно выявить при помощи журнала 166 AtomMind Server или событий 73 системы.

Сбой сервера из-за невыполнения проверки данных

Наши новые партнеры и заказчики часто жалуются, что AtomMind Server может дать сбой в случае неправильной конфигурации, и удивляются, почему конфигурация не проверена на "защиту от неумелого обращения". Тем не менее, отсутствие валидации данных на уровне платформы явно определено стратегией развития платформы.

AtomMind - инструмент low-code разработки, поэтому платформа должна быть максимально гибкой, чтобы охватить все возможные сценарии применения. Однако, конфигурация платформы, прекрасно подходящая для одного случая, приведет к сбою системы в другом практическом случае.

Например, если мы конфигурируем драйвер устройства базы данных 647, чтобы синхронизировать 514 его переменную, следом за чем идет сложный SQL запрос с периодом в 10 миллисекунд, AtomMind Server скорее всего даст сбой при попытке выполнить сотню "тяжелых" запросов в секунду. В то же время, мы не можем ограничить минимальный период синхронизации оной секундой (как иногда просят пользователи) потому что, например, для драйвера Modbus 583 довольно типично читать реестры ПЛК каждые несколько миллисекунд. Более того, даже драйвер одной и той же БД может читать переменные и отправлять простые запросы раз в несколько сотен миллисекунд!

В другом примере, неопытный пользователь может установить размер пула потоков модели 816 на несколько тысяч потоков при настройке периферийного сервера с ограниченными вычислительными ресурсами. Это безусловно приведет к сбою сервера при обслуживании любых запросов. В то же время мы не можем ограничить размер пула, скажем, сотней потоков, так как крупные 128-ядерные серверы, работающие в сердце ситуационного центра, могут запускать модели с десятком тысяч потоков.

Идею отсутствия защиты от сбоев можно лучше понять, сравнив ее с разработкой нового ПО с использованием языка объектно-ориентированного программирования. Например, вы можете написать неэффективный и с ошибками код на Java, который даст сбой, "съест" много памяти и места на диске и даже затронет другие приложения на том же хосте. Однако, сама по себе платформа для разработки на Java не ограждает вас от написания подобного кода: это работа архитектора ПО - обеспечить, чтобы все было сделано правильно.

Точно так же архитекторы приложений на AtomMind должны обеспечить соответствие конфигурации AtomMind Server конкретным условиям и ресурсам выполнения. Кроме того, обычно они же реализуют валидацию данных изнутри сервисов и продуктов на базе платформы, тем самым защищая конечных заказчиков и операторов системы от отказа системы в случае простой ошибки ввода данных.

Сервер потребляет слишком много оперативной памяти

Изучите раздел Использование памяти 211, чтобы понять как настраивается использование памяти сервером AtomMind и как его правильно интерпретировать.

Регистрируется ошибка: "Не способен создать новый поток"

Это сообщение означает, что ОС не позволяет виртуальной машине Java создавать новый поток. Существует несколько решений для данной проблемы:

- Если ваш сервер использует модели 810 или виджеты 943, запущенные на сервере виртуальной машины, уменьшить количество параллельных привязок, используемых ими.
- Уменьшить максимальную память, доступную JVM 158 (Heap Size JVM). Это увеличит объем памяти, доступной для индивидуальных потоков.
- Уменьшить размер стека потока 158.
- Переключиться на 64-битную ОС и виртуальную машину Java.

Ошибка нехватки памяти: Java Heap Space

Эта ошибка означает, что Вашей виртуальной машине Java не хватает памяти. В этом случае следует увеличить максимальную память, доступную JVM 158.

Ошибка нехватки памяти: PermGen Space

Эта ошибка означает, что Вашей виртуальной машине Java не хватает памяти для хранения загруженных классов. В этом случае следует увеличить значение параметра `-XX:MaxPermSize` в параметрах загрузчика сервера 158.

AtomMind Server, запущенный на Linux, регистрирует ошибку "java.net.SocketException: слишком много открытых файлов"

Это происходит, когда AtomMind Server открывает большое количество сокетов, общаясь при этом с подключенными к сети устройствами (например, через протокол SNMP).

Для решения этой проблемы добавьте следующую строку к началу скриптов запуска AtomMind Server (`am_server`, `am_server_console` и `am_server_service`):

```
ulimit -n 100000
```

Эта команда увеличивает лимит открытых файлов, разрешенных для AtomMind Server, до 100000.

Скрипты запуска сервера будут заново созданы во время обновления сервера, поэтому эту инструкцию нужно добавить заново при обновлении сервера.

Консольный вывод AtomMind Server содержит мусор, когда используется неанглийская версия AtomMind

Это происходит, когда AtomMind Server запущен в консольном режиме 160, поскольку командный процессор Windows не использует кодирование UTF-8 по умолчанию. Для активации UTF-8 как кодирования по умолчанию:

- Откройте редактор регистратора
- Пройдите по `HKEY_LOCAL_MACHINE\Software\Microsoft\Command Processor\AutoRun`
- Создайте параметр `AutoRun` типа REG_SZ, если он не существует
- Поменяйте значение параметра `AutoRun` на `chcp 65001`
- Перезапустите AtomMind Server

я не указал ни IP адрес, ни имя хоста в настройках некоторых подключений, а подключение до сих пор успешно установлено

AtomMind Server наследует поведение по умолчанию виртуальной машины Java и разрешает незаполненные IP адреса и имена хостов в возвращаемый адрес `localhost` (`127.0.0.1`). Таким образом, если локальная машина может принять подключение на указанный TCP/UDP порт, подключение может быть успешным даже при незаполненном адресе.

6.8.3.1 AtomMind Server не отвечает

Иногда бывают такие ситуации, когда доступ к серверу не возможен, но он тем не менее запущен. Вы можете видеть, что процесс запущен, но когда пытаетесь получить доступ к Web UI 220 или подключиться к нему, используя AtomMind Client 359, запрос превышает лимит времени.

Такое поведение может быть обусловлено несколькими причинами:

1. AtomMind Server не имеет достаточной памяти
2. AtomMind Server использует своп-память
3. AtomMind Server вызывает большую нагрузку на процессор
4. AtomMind Server запущен с ошибками
5. Доступ к AtomMind Server блокируется брандмауэром

Если не удается установить соединение с AtomMind Server, следует:

- Убедиться, что AtomMind Server запущен (например, найти в системном трее иконку 173 AtomMind Server). См. раздел Проблемы запуска AtomMind Server 212, чтобы найти и устранить неисправность при запуске сервера.
- Проверить, что firewall, запущенный на ПК AtomMind Server, не блокирует входящие подключению. Firewall должен разрешать входящие подключения на TCP порт 6460(AtomMind Client 359), TCP порт 8443 (безопасные соединений Web UI) 220 и TCP порт 8080 (Web UI 220 незащищенное соединения). Попробуйте полностью отключить firewall.
- Убедиться, что Вы задали правильный адрес и порт AtomMind Server в настройках подключения к серверу (если используется AtomMind Client) или адресная строка браузера (для Web UI).
- Если IP-адрес сервера 179 задан в глобальных настройках AtomMind Server, убедитесь, что Вы пытаетесь подключиться к серверу, используя этот адрес.
- Возможно, что AtomMind Server принимает входящие соединения не на стандартных портах. Проверьте в глобальных настройках сервера текущий номер порта клиента 181 и номера порта Web UI (защищенной 355 и незащищенный 355).

У AtomMind Server недостаточно памяти

Недостаток памяти - самая распространенная причина низкой производительности сервера и его зависания. Сервер никогда не будет использовать больше памяти, чем определено параметром `-Xmx`, расположенным в файлах 162 `server_*.vmoptions`. Чем ближе реальное использование памяти к этому лимиту, тем больше процессорного времени тратится на *Сбор мусора Java* (очистку памяти). Если сервер попытается занять больше памяти, чем разрешено параметром `-Xmx`, он потратит все процессорное время, пытаясь освободить память, и по сути, прекратит отвечать.

Постарайтесь увеличить значение параметра `-Xmx` до 50% оперативной памяти серверной машины, если база данных 692 сервера находится на той же машине, и до 75% оперативной памяти при использовании автономного сервера базы данных.

Более подробно см. главу Использование памяти 211.

AtomMind Server использует своп-память

AtomMind Server основан на платформе Java, чье управление динамической памятью очень чувствительно к свопингу. Проще говоря, если содержимое памяти, выделенное приложением Java, передается операционной системой в пространство свопинга, производительность приложения значительно снижается, что приводит к отсутствию всякого ответа.

Максимальный размер памяти, используемый AtomMind Server, определяется параметром `-Xmx`, расположенным в файлах `server_*.vmoptions`. Если это значение выставлено слишком высоко, сервер выделит много памяти, а лежащая в основе операционная система, скорее всего, перенесет определенные части в низкопроизводительное пространство свопинга.

Используйте диагностические инструменты использования свопа (такие как `smem` для Linux), чтобы убедиться, что AtomMind Server не использует пространство свопинга. Если использует, снизьте значение параметра `-Xmx` (убедившись при этом, что у сервера достаточно памяти для этого) или увеличьте оперативную память машины сервера.

AtomMind Server вызывает большую нагрузку на процессор

Может случится так, что AtomMind Server потребляет почти 100% процессорного времени. Такая ситуация часто вызвана неправильными настройками системных объектов, скорее всего, задан короткий период опроса или других действий с частой периодичностью выполнения.

Чтобы найти основную причину большой нагрузки на процессор, следует последовательно деактивировать типы системных ресурсов. Если нагрузка на процессор снижается до нормального значения после деактивации ресурсов определенного типа, включайте их обратно один за другим, чтобы определить ресурс, чьи настройки вызывают избыточную нагрузку на процессор.

1.**Временно отключите все устройства**. Настройки синхронизации устройства могут вызывать частый опрос, который расходует много процессорного времени.

2.**Отключите все тревоги**. У какого-то триггера тревог может быть слишком короткий период проверки, или же может вызываться операция, требующая для выполнения больших ресурсов (например, запрос).

3.**Отключите все датчики**. Один из них может слишком часто выполнять операцию подсчета, или же может вызываться операция, требующая для выполнения больших ресурсов.

4.**Отключите все запланированные задачи**. У задачи могут быть триггеры, которые вызывают ее слишком частое выполнение, или же сама задача может потреблять много процессорного времени, даже если выполняется лишь единожды.

5.**Отключите все инструментальные панели**. Даже будучи запущенными на компьютере оператора, инструментальные панели могут ссылаться на серверные операции, приводящие к высокой загрузке процессора.

Может оказаться сложным контролировать сервер при высокой нагрузке на процессор. Чтобы этого избежать, попробуйте запустить AtomMind Server в безопасном режиме 172. Затем удалите или подстройте ресурсы, снижающие производительность процессора, и перезапустите сервер в нормальном режиме.

Сервер запущен с ошибками

Проверьте файл журнала 166, чтобы найти причину сбоя в работе сервера. Если файл журнала не помогает разрешить проблему:

- Выключите сервер (см. выключение сервера 161)
- Перезапустите сервер в режиме консоли (используйте исполняемый файл **am_server_console**)
- Проверьте консольный вывод на наличие появившихся ошибок

Доступ блокируется Firewall

Может случится так, что запуск AtomMind Server осуществлен успешно, но вы не можете подключиться к нему, используя AtomMind Client или браузер. В файле журнала не видно неуспешных попыток подключения. В этом случае попробуйте отключить firewall, запущенный на ПК AtomMind Server. Он может мешать серверу принимать подключения клиента.

Если AtomMind Server запущен как сервис 159, возможно, придется вручную добавлять следующие порты в список исключений для firewall для того, чтобы разрешить AtomMind Server принимать подключения на данных портах:

- Порт **6460**, TCP (подключения AtomMind Client 359)
- Порт **6450**, TCP (подключения устройства)
- Порт **6440**, TCP (подключения Net Admin 1448)
- Порт **8443** (безопасные подключения к Web UI 220 и Web-сервису 1417)
- Порт **8080** (небезопасные подключения к Web UI 220 и Web-сервису 1417)
- Порт **161**, UDP/TCP (SNMP-ловушки 643)

6.8.4 Ошибки во время работы AtomMind Client

Этот раздел описывает различные ошибки при выполнении, которые могут быть диагностированы при использовании функции регистрации 422 AtomMind Client.

Регистрируются ошибки: "Невозможно создать новый поток"

Это сообщение означает, что операционная система не позволяет виртуальной машине Java создать новый поток. См. возможное решение здесь 214.

Клиент потребляет слишком много оперативной памяти

AtomMind Client использует столько оперативной памяти, сколько допустимо для него. Более подробно об этом см. Сервер потребляет слишком много оперативной памяти 214.

неправильная отрисовка

При возникновении проблем с отрисовкой внутри десктопной версии вашего AtomMind Client, следуйте следующей инструкции:

- Установите общую переменную окружения `J2D_D3D` на `false` внутри вашей ОС. Эта настройка используется для отключения использования Direct3D системой Java 2D в Java 1.4.1_02 и более поздних версиях.
- Если у вас ОС Windows, откройте "Control Panel" -> "System" -> "Advanced System Settings" -> "Environment Variables..."; затем кликните по кнопке "New" в разделе "System Variables" и введите `J2D_D3D` и `false` в поля "Variable name:" и "Variable value:". Сохраните изменения и перезагрузите ПК.
- В качестве альтернативы, вы можете добавить параметр `-Dsun.java2d.d3d=false` в файл `client.vmoptions`.

Зависание клиентского приложения

Чещу всего причиной снижения производительности AtomMind Client является недостаточный объем памяти при работе со сложными панелями инструментов 912/виджетами 943 или с большими таблицами данных 49. В этом случае попробуйте увеличить размер памяти Java VM (параметр `-Xmx`). Иногда необходимо увеличить этот параметр в несколько раз по сравнению со значением по умолчанию. Более подробно об этом см. раздел Настройка параметров Java VM 363.

6.8.5 Оптимизация производительности

AtomMind - это сложная система, производительность которой зависит от многих факторов. В этом разделе приведены советы по оптимизации для различных компонентов системы.

Оптимизация архитектуры вашего приложения

AtomMind - очень гибкая платформа малокодовой разработки. В большинстве случаев функционал одного приложения можно реализовать множеством способов, используя разные модули AtomMind. Существует бесконечное количество сценариев конфигурирования для этих модулей.

Все модули, методы и шаблоны, используемые для создания готового бизнес-приложения или сервиса, составляют архитектуру этого приложения/сервиса. Если архитектура спроектирована не лучшим образом, это отрицательно скажется на эффективности готового приложения с точки зрения использования ресурсов.

Таким образом, даже если в вашем приложении нет очевидных программных ошибок, в случае слишком большого потребления ресурсов следует внимательно пересмотреть архитектуру приложения. Обычно обнаружение и устранение всего нескольких узких мест способно повысить производительность в 10, а то и 100 раз.

Апгрейд оборудования AtomMind Server

AtomMind Server - это центральный компонент системы, и его производительность во многом влияет на остальные компоненты системы. Есть три ключевых фактора, которые влияют на производительность сервера:

- Количество процессоров/ядер и их частота
- Объем установленной оперативной памяти
- Производительность ввода-вывода подсистемы хранения данных

Рекомендуемые параметры оборудования можно найти в разделе Системные требования 146. Однако для некоторых драйверов устройств требуется большой объем памяти и/или вычислительная мощность.

В целом, AtomMind Server следует перенести на сервер с большей производительностью, если долгосрочная средняя нагрузка CPU превышает 50%. Такая высокая нагрузка может вызвать проблемы или даже утрату данных во время максимальных нагрузок.

Увеличение памяти может потребоваться в следующих случаях:

- Если обнаружено интенсивное использование своп или участились прерывания Page Fault
- Если в процессе работы AtomMind Server увеличивается время отклика
- Если в файле журнала сервера появляется сообщение `java.lang.OutOfMemoryError: Java heap space`

Рекомендуемая последовательность для решения проблем с памятью:

1. Увеличить лимиты памяти 219 для Виртуальной машины Java AtomMind Server
2. Физически добавить оперативную память в ПК, на котором запущен AtomMind Server
3. Переключиться на выделенный сервер БД

Увеличение лимита памяти JVM AtomMind Server

Вопрос расхода ресурсов памяти следует тщательно проанализировать, если датчик использования памяти AtomMind Server 2185 сообщает об объемах использования выше 90%. Важно знать, что Виртуальная машина Java AtomMind Servera, аналогично любой другой JVM, никогда не будет использовать всю RAM, доступную на ПК, на которой она запущена. Использование памяти JVM контролируют два параметра:

- Начальный объем памяти, назначенный JVM
- Максимальный объем памяти, который может использовать JVM

Эти параметры описаны здесь 158.

Если датчик использования памяти сообщает о высоком проценте использования памяти, необходимо увеличить параметр (`-Xmx`) максимального объема памяти в параметрах загрузчика AtomMind Server 158. Рекомендуется установить значение до половины от доступной оперативной памяти.

Установление максимального объема памяти может вызвать проблему с запуском AtomMind Server , поскольку JVM необходима эта память, чтобы быть доступной в одном сплошном блоке (дополнительную информацию см. здесь 213). Если Вы столкнулись с такой проблемой, рекомендуется установка дополнительной памяти.

Переключение на выделенный сервер БД

При крупных инсталляциях AtomMind с тысячами устройств, AtomMind Server обычно работает при большой нагрузке, получая миллионы событий и выполняя сотни одновременных синхронизаций 514. Все данные сохраняются в базе данных 692. В некоторых случаях даже производительность многопроцессорной системы оказывается недостаточной для выполнения обработки всех данных.

При таких крупных инсталляциях, и AtomMind Server, и сервер БД, запущенные на одной и той же машине, могут вызвать серьезное снижение производительности. Мы рекомендуем разместить Ваш сервер БД на выделенной машине, которая подключена к машине AtomMind Server через высокоскоростной сетевой интерфейс.

Рассмотрите возможность перехода на выделенный сервер БД, если:

- У Вас более 10 000 устройств, управляемых одной инсталляцией AtomMind Server
- Ваша БД содержит более 100 миллионов событий
- Существуют устройства, чья история событий содержит более 1 миллиона событий (и, таким образом, есть таблицы баз данных с более чем 1 миллионом записей)
- Объем памяти, необходимый для хорошей производительности для Вашего сервера БД, составляет больше 50% от физически установленной на ПК, на которой запущены AtomMind Server и сервер БД.
- Общая производительность Вашей системы AtomMind низкая из-за высокой загрузки CPU или интенсивного использования файла подкачки.

7 Web UI

AtomMind Web UI - это веб интерфейс на базе браузера, позволяющий пользователям получить доступ к платформе AtomMind и работать с решениями, продуктами и сервисами на ее основе. Операторы, администраторы, разработчики и инженеры используют Web UI, чтобы:

- Разворачивать, настраивать и администрировать AtomMind Server, все его модули и плагины 207, а также все продукты и сервисы на его основе
- Подключать, настраивать, мониторить устройства и источники данных 497, а также управлять ими
- Хранить, анализировать и визуализировать любые данные в рамках единой модели данных 41
- Разрабатывать решения, продукты и сервисы с использованием инструментов серверной и клиентской разработки AtomMind
- Использовать разработанные решения, продукты и сервисы на практике

Альтернатива Web UI - более старое приложение AtomMind Client 359 для десктопа. Оно еще может применяться для некоторых проектов по автоматизации, требующих пользовательских интерфейсов типа терминала, SCADA и подобных.

Работа Web UI опирается на Встроенный веб сервер.

Встроенный веб сервер

Так как Web UI - это полнофункциональное интернет приложение на основе React (приложение, к которому можно получить доступ через стандартный веб браузер), для работы ему необходим веб сервер. Этот веб сервер 354 встроен в AtomMind Server и плотно интегрирован с его функциями. Веб сервер запускается автоматически при запуске AtomMind Server и выключается при его остановке.

Чтобы узнать подробнее о конфигурировании встроенного веб сервера, см. раздел Настройка веб сервера 354.

7.1 Требования

Получить доступ к Web UI можно через веб браузер на настольном ПК, мобильном устройстве или любой другой платформе.

В настоящее время поддерживаются последние версии следующих веб-браузеров:

- Google Chrome
- Mozilla Firefox

Использование других браузеров для работы с Web UI может привести к непредсказуемому поведению.

7.2 Получение доступа к Web UI

Перед получением доступа к Web UI, убедитесь, что AtomMind Server успешно запущен и работает в нормальном режиме. Процедура запуска описана в разделе Запуск и остановка 158.

Доступ к Web UI через иконку в трее

Данный метод пригоден только для открытия Web UI с того же оборудования, которое запускает AtomMind Server:

- Поместите иконку AtomMind Server в системный трей 173.
- Нажмите правой клавишей мыши по иконке.
- Выберите пункт **Open AtomMind Server Web UI.**

Прямой доступ к Web UI

Данный метод позволяет получить доступ к Web UI с любого устройства:

- Откройте веб-браузер, такой как Google Chrome или Mozilla Firefox.
- В адресной строке введите `https://xxx.xxx.xxx.xxx:8443/web`, где xxx.xxx.xxx.xxx - IP адрес или имя хоста AtomMind Server (если вы работаете локально на сервере, то `localhost`).

- В появившемся диалоговом окне входа в систему, введите имя пользователя и пароль вашей учетной записи 478 .

Если активна настройка глобальной конфигурации Включить незащищенный доступ к веб-приложениям 354, возможно также получить доступ к Web UI через небезопасное подключение с использованием URL:

```
http://xxx.xxx.xxx.xxx:8080/web
```

Если AtomMind Server использует нестандартные номера портов (защищенный 355 и/или незащищенный 355) для получения доступа к веб-подключениям, подключение к Web UI через указанные выше URL будет невозможным. В этом случае:

- Проверьте текущие номера портов в файле конфигурации AtomMind Server 178, либо через подключение к AtomMind Server при помощи AtomMind Client 359 и просмотр глобальной конфигурации.
- Замените номер порта в одной из указанных выше URL и повторите попытку получения доступа к серверу.

В разделе URL и маршрутизация 225 см. более подробно о перенаправлении операторов на определенную инструментальную панель или другой ресурс web UI.

Проблемы подключения

Если вы убедились в том, что Web UI активен, и используется правильная URL, но так и не можете получить доступ к Web UI, попробуйте выполнить следующее:

- Убедитесь, что включен встроенный веб сервер (опция Включить встроенный веб-сервер 354 установлена на **true**).
- Проверьте файл журнала AtomMind Server 167 на наличие ошибок при запуске Web UI.
- Если вы не нашли решение по устранению проблемы, свяжитесь по службой поддержки ТВЭЛ.

7.3 Вход/выход из системы

Как только вы получили доступ 220 к Web UI в окне браузера, вам необходимо войти в систему, используя имя и пароль вашей учетной записи AtomMind Server 478.

Введите свои имя и пароль в диалоговом окне авторизации и нажмите **Вход**, чтобы пройти аутентификацию и начать работу с Web UI.

Если вы получили доступ к AtomMind впервые, пожалуйста, ознакомьтесь с документацией об идентификационных данных по умолчанию учетных записей администратора 479 и оператора 480.

Ссылка **Помощь** открывает самую последнюю онлайн версию документации AtomMind.

Ссылка **Зарегистрироваться** позволяет создать новую учетную запись AtomMind Server 478, если включена опция глобальной конфигурации Включить саморегистрацию пользователя 180.

Саморегистрация

При нажатии ссылки **Зарегистрироваться** в диалоговом окне авторизации, вы перейдете на страницу саморегистрации.

Введите желаемое имя пользователя, пароль и другую информацию и нажмите **Зарегистрироваться**, чтобы создать новую учетную запись. Далее перед вами снова появится окно входа в систему.

Выход из системы

Для выхода из Web UI, нажмите кнопку **Выход** на инструментальной панели главного окна. Перед вами появится окно входа в систему.

7.3.1 Упрощенный режим

AtomMind - платформа, обеспечивающая работу приложений, поэтому ее пользователи делятся на "продвинутых" (администраторы платформы, системные инженеры, разработчики решений/сервисов/продуктов и др.) и "обычных" пользователей, работающих с решениями и сервисами на базе платформы (операторы, инженерно-технический персонал, аналитики, специалисты по обработке данных, администраторы уровня приложения и др.)

Если продвинутым пользователям необходим доступ к конфигурации самой платформы и административным действиям, то доступ обычных пользователей должен быть ограничен уровнем интерфейсов определенного приложения.

Упрощенный режим применяется, чтобы скрыть административные интерфейсы платформы от обычных пользователей. Упрощенный режим активируется настройкой Упрощенный режим 222 в учетной записи пользователя, и после входа пользователя с систему административные панели инструментов не отображаются. В упрощенной режиме отображаются только инструментальные панели и компоненты UI, добавленные в список автозапуска 941.

Если никакие действия автозапуска не настроены для пользователя, то после входа в упрощенном режиме пользователь увидит пустую страницу.

7.4 Обзор Web UI

На верхнем уровне, компоновка Web UI состоит из тулбара 222 и *инструментальных панелей* с различными *компонентами* и *редакторами*.

По умолчанию имеется только одна инструментальная панель - **Default Dashboard**. Компоновка по умолчанию основной инструментальной панели включает:

- Системное дерево 275
- Приветствие с описанием возможных дальнейших действий

Если никакие действия автозапуска 941 не были настроены для пользователя, то после входа с использованием упрощенного режима 222 пользователь увидит пустую страницу. Это не указывает на ошибку. Пожалуйста, используйте список автозапуска для создания интерфейса конкретного приложения для системных операторов, использующих при входе упрощенный режим.

7.4.1 Тулбар

Тулбар в Web UI включает несколько элементов:

- Логотип AtomMind .
- Кнопка **Home**. Переключение на Инструментальную панель по умолчанию.
- Кнопка **Logout**. Завершение текущей сессии работы с Web UI и выход текущего пользователя из системы. Перенаправление на страницу входа в систему 221.

7.4.2 Всплывающие окна тревог

Всплывающие тревоги - это небольшие окна, которые появляются каждый раз при возникновении тревог 779, если была активирована опция **Показать всплывающее окно владельцу**. Окна тревог всплывают поверх других компонентов Web UI.

Окно тревоги показывает ключевую информацию о конкретной тревоге: уровень критичности, имя, сообщение, триггер, причина и др.

Всплывающее окно позволяет управлять тревогой:

- Закрывать или закреплять всплывающее окно
- Просматривать подробную информацию, связанную с тревогой
- Открывать и менять конфигурацию тревоги

7.5 Инструментальные панели

Высокоуровневыми элементами Web UI являются инструментальные панели. Обычно инструментальные панели создаются и редактируются в конструкторе UI 348. Как только инструментальная панель открывается в действующем приложении, она становится динамической веб страницей, которая отображает и обновляет данные сервера, реагирует на ввод оператора и реализует маршрутизацию 225 на другие инструментальные панели (веб страницы).

Поскольку каждая инструментальная панель - это контекст сервера (такой же, как тревога 779 или модель 810), администрирование и общая настройка панелей описаны в специальном разделе Инструментальные панели 912 документации AtomMind Server.

7.5.1 Модель данных

Каждая инструментальная панель представлена контекстом 41 на серверной стороне, который содержит конфигурацию компоновки 240 инструментальной панели, компонентов 231, привязок 226 и так далее.

В то же время, с началом жизненного цикла 223 инструментальной панели (например, с запуском инструментальной панели) создается отдельное контекстное дерево 41, которое представляет модель данных запущенной инструментальной панели. Каждый контекст в таком дереве соответствует одному компоненту UI 231 запущенной инструментальной панели. Контексты всех компонентов добавляются в корневой каталог дерева контекстов, поэтому иерархия контекстов компонентов не соответствует иерархии контейнеров 240 и компонентов, встроенных в инструментальную панель.

Переменные 61, функции 70 и события 73 контекста компонентов инструментальной панели соответствуют свойствам, операциям и событиям этого UI компонента. Таким образом, обращаясь к элементам контекста компонента из привязок 226 инструментальной панели, мы, по сути, работаем с компонентами UI - конфигурируем их, реагируем на их события, и пр.

7.5.2 Экземпляры

Экземпляр веб-панели инструментов - это копия панели инструментов, которая запущена или кэширована 224 в памяти AtomMind Server.

Каждый экземпляр содержит контекстное дерево 41 UI компонентов 231 панели инструментов, включая фактические значения свойств компонентов, загруженные из шаблона панели, либо полученные в результате выполнения привязок 226 инструментальной панели.

ID экземпляра

Каждый экземпляр инструментальной панели определяется несколькими параметрами, которые составляют его уникальный ID:

- Пусть контекста инструментальной панели 1490
- Токен (уникальный ID) веб-сессии пользователя 478 Web UI
- Уникальный ID вкладки браузера с запущенной инструментальной панелью

ID экземпляров инструментальной панели используются для хранения и поиска экземпляров в кэше 224.

7.5.3 Жизненный цикл

Как только пользователь открывает инструментальную панель во время сессии работы с Web UI, внутри AtomMind Server создается экземпляр 223 этой инструментальной панели, и начинается ее жизненный цикл.

Структура жизненного цикла инструментальной панели сильно зависит от того, включено ли кэширование 224 для конкретного экземпляра.

Жизненный цикл некэшируемой инструментальной панели

Жизненный цикл некэшируемой инструментальной панели прост и прозрачен:

1. Инструментальная панель запускается
2. Выполняются привязки 226 при запуске

3. Инструментальная панель отображается в окне браузера
4. Событие отображение 244 возникает в корневом контексте инструментальной панели
5. Инструментальная панель запущена
6. Другая инструментальная панель появляется на переднем плане
7. Инструментальная панель скрывается
8. Обработчик привязок инструментальной панели останавливается, панель закрывается

Основная фаза работы инструментальной панели заканчивается, когда не передний план выходит другая панель. Это может произойти в следующих случаях:

- Для панелей верхнего уровня - при нажатии на ссылку или другом действии, которое перенаправляет пользователя на другую веб-страницу с изменением URL браузера
- Для вложенных панелей 255 - при скрытии контейнера, в который вложена панель (например, вкладка панели со вкладками 246)

Однако, если вкладка браузера переключена пользователем, основная фаза работы инструментальной панели продолжается.

Жизненный цикл кэшируемой инструментальной панели

Жизненный цикл кэшируемой инструментальной панели предполагает, что работу панели можно приостановить и возобновить множество раз:

1. Инструментальная панель запускается
2. Выполняются привязки 226 при запуске
3. Инструментальная панель отображается в окне браузера
4. Событие отображение 244 возникает в корневом контексте инструментальной панели
5. Инструментальная панель запущена
6. Другая инструментальная панель появляется на переднем плане
7. Инструментальная панель скрывается
8. Обработчик привязок инструментальной панели останавливается, панель ставится на паузу
9. Если кэш 224 инструментальной панели переполняется, панель закрывается
10. Если инструментальная панель снова выходит на передний план, жизненный цикл повторяется с шага 3 (инструментальная панель отображается в окне браузера)

Кэшируемая инструментальная панель используется повторно при выходе на передний план в одном из следующих случаев:

- Для панелей верхнего уровня - если пользователь перенаправляется на URL инструментальной панели 225.
- Для вложенных панелей 255 - при отображении контейнера, в который вложена панель (например, вкладка панели со вкладками 246)

7.5.4 Кэширование

Механизм кэширования экземпляра инструментальной панели обеспечивает два важных производительных 352 преимущества:

- Как только скрытая и кэшированная панель вновь отображается, выполняемые при ее запуске привязки 226 не вычисляются повторно
- Кэшированные инструментальные панели содержат ввод информации пользователем (например, значения в текстовых полях), который еще не был преобразован в "основную" модель данных 41 сервера

Можно активировать кэширование для всех панелей (через настройки плагина Web UI 353) или только для некоторых панелей (через настройки инструментальной панели 914).

Настройка кэша позволяет ограничить размер кэша определенным количеством ID экземпляров 223 инструментальной панели или максимальным объемом оперативной памяти, занимаемой всеми панелями. При превышении размера кэша самые старые экземпляры инструментальной панели будут закрыты, и при их

следующем отображении пользователю произойдет полный рестарт (включая выполнение привязок 226 при запуске и пр.)

Более подробно о том, как кэширование влияет на поведение инструментальных панелей, см. в разделе Жизненный цикл 223.

7.6 URL и маршрутизация

Существует несколько способов доступа к Web UI, которые позволяют получить доступ к его административной части, определенному приложению (доступ через автозапуск 941) или конкретной инструментальной панели 223.

В таблице описываются URL, позволяющие получить доступ к определенным частям Web UI:

URL	Описание
/	Домашняя страница встроенного веб сервера 354 AtomMind Server. Переадресация на /web/admin.
/web	Домашняя страница Web UI 220. Переадресация на /web/admin.
/web/login	Страница прохождения аутентификации и авторизации 477 пользователя. На эту страницу происходит переадресация пользователя с любых других страниц Web UI, если пользователь еще не прошел аутентификацию. Когда аутентификация пройдена, пользователь вновь оказывается на перенаправившей его странице.
/web/admin	Поведение этой страницы зависит от того, активирован ли упрощенный режим 222 у авторизованного пользователя: • Если упрощенный режим отключен, отображается *панель инструментов* Web UI *по умолчанию*, которая дает пользователю доступ к системному дереву 275 и позволяет конфигурировать систему • Если упрощенный режим включен, эта страница делает переадресацию на /web/dashboards чтобы инициировать последовательность действий автозапуска и отобразить специальный интерфейс приложения
/web/dashboards	Страница запускает последовательность действий при автозапуске 941 для авторизованного пользователя. Автозапуск используется в основном для определения инструментальных панелей, которые должны открываться по умолчанию.
/web/dashboards/*dashboard_context_path*	Страница открывает определенную инструментальную панель, указанную частью dashboard_context_path в URL. Например, /web/dashboards/users.admin.dashboards.overview URL открывает инструментальную панель overview, принадлежащую администратору по умолчанию 479.
/web/cc	Адрес центра управления Web 354.

7.7 Гибкий дизайн

Инструментальные панели, созданные для Web UI, могут динамически адаптироваться под различные расширения экрана и размеры окна браузера. Такой подход широко известен как *гибкий веб дизайн*.

Процесс создания гибких интерфейсов выглядит следующим образом:

- Тулбар 349 конструктора Web UI содержит поле со списком **Выбор расширения**, которое выбирает режим редактирования для различной ширины окна браузера
- После того, как выбрана ширина окна, любые и изменения в сетке 241 будут соответствовать выбранной ширине
- Все изменения сетки отражаются в соответствующих строках таблицы Свойства расширения 232, которая принадлежит контейнеру редактируемой сетки (например, если выбрана ширина Tablet 800px, все изменения сетки будут сохраняться в строках с параметром **Разрешение**, равным 800).

Когда инструментальная панель отрисовывается во время выполнения, Web UI перестраивает ее компоновку каждый раз при изменении ширины окна браузера.

Для каждого компонента в контейнере с сеткой выбирается строка таблицы **Свойства расширения** с наиболее релевантной шириной (указанной параметром **Разрешение**). Ограничения сетки (X, Y, ширина, высота), указанные в этой строке, используются для отрисовки компонента в сетке контейнера.

7.8 Привязки

Если компоненты 231 UI составляют "тело" инструментальной панели, то ее "механизм" представлен привязками 735. Привязки определяют отношения между компонентами виджетов и данными из контекстов 41 сервера, такими как их переменные 61 и функции 70. Каждая привязка *вычисляется* в разные моменты работы инструментальной панели. Результаты вычислений используются для изменения UI компонентов или модели данных сервера.

Привязки веб инструментальной панели расширяют привязки сервера 736. В то время как привязки сервера определяют отношения объектов на стороне сервера, привязки инструментальной панели могут также определять отношения между объектами сервера и визуальными компонентами, используя при этом тот же самый синтаксис.

Просматривать, создавать и редактировать привязки инструментальной панели можно с помощью свойства **Привязки** корневой панели 241. Эта таблица содержит все привязки, доступные для инструментальной панели.

7.8.1 Цель привязки

Цель привязки - это объект, на который воздействует привязка. Например, целью может быть свойство компонента инструментальной панели, либо какие-то данные внутри контекста AtomMind Server.

По сути, цель привязки - это особый вид ссылки 117.

Поддерживаются несколько типов целей привязок:

- Ссылки без схемы 117 (а значит, без префикса), которые указывают на данные контекста
- Ссылки, использующие схему **form** (и префикс `form/`) указывающие на свойства компонентов инструментальной панели

Цели привязок инструментальной панели поддерживают все варианты синтаксисы целей привязок сервера 737.

Дополнительно поддерживаются следующие варианты синтаксиса:

1. Свойство компонента

`form/component:property`

Данная цель привязки указывает на определенное **свойство** компонента инструментальной панели.

Цель инструментальной панели, которая ссылается на ячейку свойства, представленного таблицей внутри определенной **строки** и **поля**, имеет следующий формат:

`form/component:property$field[row]`

Пример: `form/textField1`

Привязка с этой целью запишет результат в свойство по умолчанию компонента `textField1`. Если `textField1` - это текстовое поле 315, его свойство **текст** будет изменено.

Пример: `form/image1:imageTable$imageData[0]`

Эта цель привязки меняет бинарное изображение в первой сроке таблицы изображений компонента Image.

Пример: `form/gauge:datasets[1]`

Эта цель привязки меняет значение второго набора данных (с индексом=1) компонента Gauge.

2. Действие контекста

`context:action!`

Эта цель начинает интерактивное выполнение действия **action** из контекста **context**. Результат вычисления привязки передается действию как входное значение. Тип значения должен быть Data Table 49.

Пример: `cardholders:import!`

Начинает действие импорта держателей карт (модуль Cardholders входит в решения Учет рабочего времени и Контроль доступа), то есть это действие с названием **import** в контексте с путем **cardholders**.

ЗАПУСК ОТНОСИТЕЛЬНЫХ ОТЧЕТОВ И ИНСТРУМЕНТАЛЬНЫХ ПАНЕЛЕЙ

Отчеты 928, инструментальные панели 912 и некоторые другие объекты системы могут быть *абсолютными* и *относительными*. Если абсолютный объект запускается как таковой, относительные объекты запускаются **для определенного контекста** 41, который служит источником данных.

Таким образом, запуск относительного отчета или инструментальной панели напрямик через цель привязки не сообщит системе, для какого контекста *должен быть запущен объект* при выполнении привязки. Вместо этого в цели привязки следует указать действие запуска объекта, **расположенное в целевом контексте**.

Пример: Допустим, есть относительная инструментальная панель **Traffic Chart**, действующая для всех сетевых устройств 497. Можно открыть график трафика **определенного устройства**, используя активатор привязок. Для этого:

- Найдите исходное устройство в редакторе целей привязок
- Выберите действие **Traffic Chart**

7.8.2 Выражение привязки

Выражения привязок инструментальной панели очень похожи на выражения привязок сервера 738. Однако выражения привязок в виджетах могут включать несколько основных типов ссылок:

- **Стандартные ссылки** (без схемы 117), которые указывают на переменные 61 или функции 70 контекста 955, их поля или свойства
- **Ссылки на компоненты** (использующие схему `form/`), которые указывают на свойства компонентов 231 инструментальной панели
- **Специальные ссылки Web UI** (использующие схему `web/`), которые запускают специфичные для Web UI операции

Ссылки на компоненты

Ссылка на компонент указывает на свойство компонента инструментальной панели (например, текст метки 280). Она имеет следующий формат:

```
form/component:property
```

`form` - имя схемы 117, используемой для идентификации ссылок на компонент. Схема говорит обработчику языка выражений использовать распознаватель 118, которые понимает ссылки на компоненты и знает, что с ними делать. Ссылки на компоненты необходимо всегда начинать с `form/`.

`component` - имя компонента 955 инструментальной панели, с которым вы хотите работать. Имя компонента отображается в дереве компонентов 350 и в заголовке окна свойств компонентов 350, если компонент выбран в конструкторе.

`property` - имя определенного свойства внутри компонента. Имена свойств можно найти в описании свойств каждого компонента инструментальной панели. Указывать свойство в ссылке на компонент необязательно. Если свойство не указано, ссылка ведет к свойству по умолчанию компонента.

Формат свойства, т.е. типа значения, возвращаемого ссылкой на свойство компонента, можно найти в описании этого свойства в справочнике компонентов 231.

ПРИМЕРЫ ССЫЛОК НА СВОЙСТВО КОМПОНЕНТА

```
{form/userNameField:}
```

```
{form/userNameField:text}
```

Обе ссылки разрешаются в текст, который содержится в `userNameField` (при условии, что это текстовое поле 315). Первая ссылка указывает на свойство по умолчанию компонента, то есть `text`, а вторая прямо называет его (`:text`).

Специальные ссылки Web UI

Специальная ссылка инициирует выполнение операции, специфичной для Web UI. Ссылка имеет следующий формат:

```
web/operation(["parameter"])
```

Поддерживаются следующие операции:

- `web/logout()` - прекращает текущую сессию Web UI и осуществляет выход пользователя
- `web/redirect("context" [, parameter1 [, parameter2...]])` - переадресует текущего пользователя на другую веб инструментальную панель[918] согласно указанному пути **контекста**
- `web/executeAction("context:action!")` - выполняет **действие** из **контекста** с указанным путем

Пример выражения привязки инструментальной панели

```
{form/numberField1:value} * 100
```

Это выражение разрешится в число, равное свойству **value** компонента Числовое поле, названного **numberField1**, умноженному на 100.

7.8.3 Активатор привязки

Активатор привязки - это специальная ссылка[117], которая указывает на:

- Свойство[231] или событие[239] компонента инструментальной панели
- Свойство (переменную) или событие контекста сервера (см. активатор привязки сервера[739])

Изменение компонента инструментальной панели или переменной контекста сервера, либо возникновение события компонента панели или контекста сервера запускает выполнение привязки.

Активаторы привязок инструментальной панели поддерживают все варианты синтаксиса активаторов привязок сервера[739].

Дополнительно поддерживаются следующие варианты синтаксиса:

1. Свойство компонента инструментальной панели

```
form/component:property
```

При изменении свойства `property` компонента `component` запустится активатор.

Пример: `form/setpoint:text`

Этот активатор запустится при редактировании свойства `text` в текстовом поле[315] с именем `setpoint`.

2. Событие компонента инструментальной панели

```
form/component:event@
```

Если у компонента `component` есть событие `event`, активатор запустится при возникновении события.

Пример: `form/button1:click@`

Этот активатор запустится при нажатии на кнопку[331] `button1`.

7.8.4 Условие привязки

Условия привязок инструментальной панели очень похожи на условия привязок сервера[740]. Однако, условия привязок в инструментальных панелях могут включать два типа ссылок:

- **Стандартные ссылки**, указывающие на переменные контекста[223] или функции[70], их поля и свойства, и
- **Ссылки на компоненты**, указывающие на свойства компонентов[231] инструментальной панели.

Более подробно о ссылках на компоненты инструментальной панели см. в разделе выражение привязки инструментальной панели[227].

Пример условия привязки инструментальной панели

`cell({env/value}, "altDown") && (cell({env/value}, "keyCode") == 'A')` - это выражение запустит привязку, чей активатор является событием типа "печать клавиши" только при нажатии комбинации клавиш Alt-A.

7.8.5 Выполнение привязок

Обычно привязки выполняются одна за другой. Например, если ваша инструментальная панель отображает 20 различных метрик устройства, и каждое чтение метрики с удаленного сервера занимает 1 секунду, загрузка инструментальной панели займет 20 секунд.

Чтобы увеличить скорость загрузки инструментальной панели, включите флаг **Параллельная работа привязок при запуске** в корневой панели 241. В этом случае, каждая привязка будет обрабатываться в отдельном потоке. В нашем примере это приведет к одновременному выполнению всех запросов метрик, и инструментальная панель загрузится за 1 секунду (в 20 быстрее).

Более подробно см. общий раздел Выполнение привязок 742.

7.8.6 Примеры привязок

В этом разделе даны некоторые реальные примеры привязок инструментальных панелей.

Пример 1: Привязка цвета панели к статусу контроллера

Эта привязка сделает панель 246 статуса красной, если контроллер пришлет тревогу о падении напряжения. В остальных случаях панель будет зеленой.

Обратите внимание, что выражение и ссылка на цель используют относительные пути контекста ("."), чтобы инструментальная панель была совместима со множеством одинаковых контроллеров. Инструментальная панель должна быть относительной 913 и действующей для всех устройств типа "контроллер напряжения".

Цель `form/voltageAlertPanel:style`

Выражение `{.:voltageAlert$voltageAlert} ? "background-color: red" : "background-color: green"`

Активатор `.:voltageAlert$voltageAlert`

Условие

Опции `On Startup, On Event`

Приведенную выше привязку можно изменить так, чтобы панель становилась красной, если показание температуры превышает 50 градусов:

```
({.:voltage$voltage} > 50) ? "background-color: red" : "background-color: green"
```

Пример 2: Привязка компонента с числовым полем к уставке контроллера

Следующая пара привязок будет менять уставку контроллера (например, контроллера температуры) при изменении значения компонента с числовым полем.

Первая привязка используется для чтения текущего значения уставки с контроллера и инициализации числового поля:

Цель `form/numberField0:value`

Выражение `{.:currentTemperature$temperatureCelsius}`

Активатор

Условие

Опции `On Startup`

Вторая привязка записывает новую уставку на контроллер после перемещения регулятора:

Цель `.:currentTemperature$temperatureCelsius`

Выражение `{form/numberField0:value}`

Активатор	
Условие	`form/numberField0:value`
Опции	`On Event`

Пример 3: Привязка редактора таблицы к результатам запроса

Привязка выполняет запрос[829] сервера, текст которого содержится в текстовом поле `queryText`, и помещает результаты запроса в компонент Редактор таблицы данных[282] с именем `results`.

Запрос выполняется путем вызова `executeQuery` корневого контекста сервера[1559]. Операция выполняется при нажатии на кнопку[331] `execute`.

Обратите внимание, что первый параметр функции `executeQuery` помещен в одинарные кавычки, чтобы система обрабатывала его как выражение.

Цель	`form/results:dataTable`
Выражение	`{:executeQuery('{form/queryText:text}')}`
Активатор	`form/executeButton:mouseClicked@`
Условие	
Опции	`On Event`

Можно использовать альтернативный синтаксис для выражения этой привязки:

```
callFunction("", "executeQuery", {form/queryText:text})
```

Это выражение работает точно так же, как и приведенное выше, но использует функцию `callFunction()` языка выражений для вызова функции контекста сервера, прямо указывая путь контекста, имя функции и ее параметры.

Пример 4: Вызов функции устройства

Этот пример объясняет, как вызвать функцию[70] устройства или сервера из инструментальной панели. Предположим, у нас есть устройство, поддерживающее операцию `multiply` с двумя числовыми аргументами. Мы хотим вызвать эту операцию нажатием кнопки `calculate`, учитывая числа, введенные в два текстовых поля (`argument1` и `argument2`).

МЕТОД ОДИН: ВЫЗОВ ФУНКЦИИ ИЗ ЦЕЛИ ПРИВЯЗКИ ИНСТРУМЕНТАЛЬНОЙ ПАНЕЛИ

В этом случае цель нашей привязки будет указывать на функцию, и ее выражение вернет таблицу предопределенных параметров:

Цель	`.:multiply()`
Выражение	`table(functionInputFormat(), {form/argument1:text}, {form/argument2:text})`
Активатор	`form/calculateButton:mouseClicked@`
Условие	
Опции	`On Event`

Обратите внимание, что мы используем относительный путь контекста (".") в цели, чтобы вычислительная инструментальная панель была совместима со множеством вычислительных агентов. Чтобы использовать ее только с одним агентом, укажите абсолютный путь, например, `users.my_user.devices.my_agent.`

В этом примере результат вычисления, возвращаемый операцией `multiply`, будет отброшен. Но что если мы захотим включить его в другой компонент инструментальной панели (например, метку[280])? Есть решение:

МЕТОД ДВА: ВЫЗОВ ФУНКЦИИ ИЗ ВЫРАЖЕНИЯ ПРИВЯЗКИ ИНСТРУМЕНТАЛЬНОЙ ПАНЕЛИ

В этом случае выражение нашей привязки вызовет функцию и вернет ее выход (Таблицу данных 49). Сам результат вычисления (числовое поле `result`) извлекается из этой таблицы при помощи функции `cell()`. Этот результат записывается в метку `result`:

Цель	`form/result:text`
Выражение	`cell(callFunction(dc(), "multiply", {form/argument1:text}, {form/argument2:text}), "result")`
Активатор	`Button:mouseClicked@`
Условие	
Опции	`On Event`

Обратите внимание, что используется относительный путь контекста устройства (возвращаемый функцией `dc()`). Чтобы использовать абсолютный путь контекста, замените `dc()` на полный путь устройства, например, `users.my_user.devices.my_agent`.

Существует другой (более компактный) стиль записи приведенного выше выражения привязки с тем же результатом.

Этот стиль использует ссылку на функцию 120 вместо функции `callFunction()` языка выражений:

```
{.:multiply('{form/argument1:text}', '{form/argument2:text}')$result}
```

7.9 Компоненты

Компоненты UI - это базовые структурные элементы веб инструментальной панели. Их можно разделить на две большие группы:

- Регулярные компоненты
- Контейнеры 240

По сути, каждая инструментальная панель - это иерархия вложенных контейнеров с различными компонентами и контейнеров, позиционированных 240 внутри них.

Модель данных компонентов

В основе каждого контейнера лежит виртуальный контекст 41, который обеспечивает единую точку доступа к его свойствам 231, функциям 238 и событиям 239. Такой подход позволяет привязкам 226 инструментальной панели передавать данные между компонентами UI и единой моделью данных 41 AtomMind Server.

Структура дерева контекстов 41 компонента не совпадает со структурой самих компонентов и их контейнеров в инструментальной панели. Каждый контекст компонента добавляется в корень контекстного дерева компонентов, таким образом, путь контекста компонента равен имени компонента.

Более подробно см. в разделе Модель данных инструментальной панели 223.

7.9.1 Свойства компонентов

Каждый UI компонент имеет несколько *свойств*. На них можно ссылаться из привязок 226, которые "маршрутизируют" данные к компонентам и от компонентов. Некоторые свойства являются общими 231 для всех компонентов инструментальной панели, другие - специфичны для определенных компонентов и поэтому описаны в разделах, посвященных этим компонентам.

Каждое свойство компонента представлено переменной в контексте компонента. Более подробно см. в разделе Модель данных инструментальной панели 223.

7.9.1.1 Общие свойства компонентов

В разделе перечислены и описаны свойства, присущие большинству компонентов инструментальной панели.

Типы свойств

Описание каждого свойства компонента виджета включает *тип* свойства. Тип свойства - это тип значения, возвращаемого ссылкой на свойство компонента из привязки инструментальной панели (String, Boolean, Integer и т.д.).

Индексируемые свойства

Индексируемое свойство - это свойство, имеющее несколько элементов, т.е. представленное массивом или списком. При ссылке на элементы индексируемых свойств из привязок 226 необходимо добавлять в цель привязки **индекс** элемента (нумерация с нуля).

Общие свойства компонентов

Здесь описываются свойства, поддерживаемые большинством компонентов инструментальной панели. Описание каждого отдельного компонента включает список поддерживаемых общих свойств.

ИМЯ

Данное свойство определяет уникальное имя компонента.

Имя переменной: name

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*.

Формат 50 записи:

Имя поля	Тип поля	Примечания
name	String	Уникальное имя компонента.

ВИДИМЫЙ

Флаг определяет видимость компонента.

Имя переменной: visible

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*.

Формат 50 записи:

Имя поля	Тип поля	Примечания
visible	Boolean	Включенный флаг показывает, что компонент видим.

СВОЙСТВА РАСШИРЕНИЯ

Данное свойство определяет поведение компонента для различных расширений экрана пользователя.

Имя переменной: gridResponsive

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*.

Формат 50 записи:

Имя поля	Тип поля	Примечания
responsive	Integer	Точка останова, при которой компонент адаптируется для текущего разрешения экрана пользователя с помощью перечисленных ниже настроек. Указывается в пикселях.

x	Integer	Горизонтальная позиция компонента на сетке 225 инструментальной панели.
y	Integer	Вертикальная позиция компонента на сетке 225 инструментальной панели.
h	Integer	Количество столбцов, занимаемых компонентом на сетке 225 инструментальной панели.
w	Integer	Количество строк, занимаемых компонентом на сетке 225 инструментальной панели.
style	String	CSS стиль для применения к компоненту.

ШИРИНА СЕТКИ

Данное свойство определяет количество столбцов, занимаемых компонентом на сетке 225 инструментальной панели.

Имя переменной: gridWidth

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*.

Формат 50 записи:

Имя поля	Тип поля	Примечания
gridWidth	Integer	Количество столбцов, занимаемых компонентом.

ВЫСОТА СЕТКИ

Данное свойство определяет количество строк, занимаемых компонентом на сетке 225 инструментальной панели.

Имя переменной: gridHeight

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*.

Имя поля	Тип поля	Примечания
gridHeight	Integer	Количество строк, занимаемых компонентом

ВЫХОД ЗА ПРЕДЕЛЫ КОНТЕНТА

Определяет поведение контекста при его выходе за пределы контейнера.

Имя переменной: contentOverflow

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*.

Формат 50 записи:

Имя поля	Тип поля	Примечания
contentOverflow	String	Данное свойство имеет несколько допустимых значений:

		• **Auto -** прокрутка включается автоматически, как только контент превысит размеры контейнера • **Hidden -** при выходе на пределы контейнера, контент скрывается • **Scroll -** прокрутка включена всегда, даже если контент не выходит за пределы контейнера • **Unset -** отменяет наследование данного свойства из любых источников (например, свойства Стиль инструментальной панели) • **Visible** - контент всегда будет видимым, даже при превышении пределов контейнера

СТИЛЬ КОНТЕЙНЕРА

CSS стиль для применения к контейнеру компонента.

Имя переменной: containerStyle

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*.

Формат 50 записи:

Имя поля	Тип поля	Примечания
containerStyle	String	CSS стиль для применения к контейнеру компонента.

ПОЛЬЗОВАТЕЛЬСКИЕ КЛАССЫ

Позволяет указать пользовательские CSS классы, которые можно использовать в свойстве Стиль 234.

Имя переменной: customClasses

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*.

Формат 50 записи:

Имя поля	Тип поля	Примечания
customClasses	String	Пользовательские CSS классы, которые можно использовать в свойстве Стиль 234.

СТИЛЬ

CSS стиль для применения к компоненту.

Имя переменной: style

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*.

Формат 50 записи:

Имя поля	Тип поля	Примечания
style	String	CSS стиль для применения к компоненту.

7.9.1.2 Общие свойства контейнеров

В разделе перечислены и описаны свойства, присущие большинству контейнеров инструментальных панелей.

ИЗМЕНЯЕМЫЙ

Если включено, позволяет менять размер добавленных в данный контейнер компонентов, изменяя количество столбцов и строк, которые они занимают в сетке.

Имя переменной: resizable

Записи: 1

Формат записи::

Имя поля	Тип поля	Примечания
resizable	Boolean	Изменяемый

ПОДВИЖНЫЙ

Если включено, позволяет перемещать добавленные в данный контейнер компоненты, меняя из положение на сетке.

Имя переменной: movable

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
movable	Boolean	Подвижный

ВЫСОТА СТРОКИ

Высота одной строки в сетке, в пикселях.

Имя переменной: rowHeight

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
rowHeight	Integer	Высота строки

ПРОМЕЖУТОК

Определяет ширину линий сетки ячеек, в пикселях.

Имя переменной: gap

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
gap	String	Промежуток

ПОКАЗЫВАТЬ ФОН СЕТКИ

Настраивает отображение ячеек сетки. Это свойство имеет несколько допустимых значений:

- Отсутствует
- Строки
- Столбцы
- Строки и столбцы

Имя переменной: showGridBackground

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
showGridBackground	String	Показывать фон сетки

ТИП СЕТКИ

Тип сетки определяет поведение инструментальной панели, когда содержимое не помещается на экране. Это свойство имеет два допустимых значения:

- **Плавающая -** содержимое инструментальной панели будет подгоняться под размер окна браузера
- **С прокруткой -** при превышении содержимым размеров окна, будет появляться вертикальная полоса прокрутки

Имя переменной: gridType

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
gridType	String	Тип сетки

КОМПОНОВКА

Определяет тип компоновки 240 контейнера. Это свойство имеет два допустимых значения:

- **Сетка -** более подробно см. в разделе сетка 241
- **Абсолютная -** более подробно см. в разделе абсолютное позиционирование 241

Имя переменной: layout

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
layout	String	Компоновка

ШАГ СЕТКИ

Размер ячейки сетки, в пикселях.

Имя переменной: gridStep

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
gridStep	String	Шаг сетки

ВЫРАВНИВАТЬ ПО СЕТКЕ

Если включено, при добавлении и перемещении компонентов они будут выравниваться по сетке абсолютного позиционирования.

Имя переменной: snapToGrid

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
snapToGrid	Boolean	Выравнивать по сетке

ИНФОРМАЦИЯ О СТОЛБЦАХ СЕТКИ

Эта переменная описывает количество столбцов сетки и их размер.

Имя переменной: gridCols

Записи: 0..не ограничено

Формат записи:

Имя поля	Тип поля	Примечания
template	String	**Template**. Числовое значение размера столбца сетки.
unit	String	**Unit**. Единица измерения, *fr* (фреймы) или *px* (пиксели).

ИНФОРМАЦИЯ О СТРОКАХ СЕТКИ

Эта переменная описывает количество строк сетки и их размер.

Имя переменной: gridRows

Записи: 0..unlimited

Формат записи:

Имя поля	Тип поля	Примечания
template	String	**Template**. Числовое значение размера строки сетки.
unit	String	**Unit**. Единица измерения, *fr* (фреймы) или *px* (пиксели).

ОТСТУП СВЕРХУ

Отступ контейнера сверху, в пикселях (например, *10px*).

Имя переменной: indentTop

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
indentTop	String	Отступ сверху

ОТСТУП СНИЗУ

Отступ контейнера снизу, в пикселях (например, *10px*).

Имя переменной: indentBottom

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
indentBottom	String	Отступ снизу

ОТСТУП СЛЕВА

Отступ контейнера слева, в пикселях (например, *10px*).

Имя переменной: indentLeft

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
indentLeft	String	Отступ слева

ОТСТУП СПРАВА

Отступ контейнера справа, в пикселях (например, *10px*).

Имя переменной: indentRight

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
indentRight	Отступ слева	Отступ справа

7.9.2 Функции компонентов

Функции компонентов виджета зависят от функций 70 инструментальной панели 231 или контейнера 240. Функции компонентов можно использовать для вызова функциональности компонента без взаимодействия с пользователем.

Как и события компонента 239, функции компонента вызываются из привязок инструментальной панели 226.

Более подробно см. в разделе Модель данных инструментальной панели 223.

Вызов функций компонентов инструментальной панели

Вы можете вызвать функции компонентов виджета из привязок виджета, используя схему 117 `form/` .

Для этого укажите функцию в качестве цели привязки 226 и создайте таблицу с параметрами функции в выражении привязки 227.

Цель	`form:componentName/componentFunction()`
Выражение	Выражение, которое вычисляется в таблицу с параметрами функции.
Активатор	Любой действительный активатор привязки 228.

Другой способ вызвать функции компонента - через выражения привязки 227. Этот метод можно применить, если вы хотите использовать выходные данные функции компонента. Как и другие функции, функции компонента имеют результатом таблицы данных 49. Если цель привязки не разрешает тип значения Таблица данных, вам придется расширить выражение, чтобы оно конвертировало результат функции в требуемый формат.

Цель	Любая действительная цель привязки 226.
Выражение	`{form/componentName:componentFunction(functionParameters)}`
Активатор	Любой действительный активатор привязки 228.

7.9.3 События компонентов

Компоненты инструментальных панелей могут генерировать события при взаимодействием с пользователем. Например, у кнопки есть событие **действие,** которое возникает, если пользователь нажал на эту кнопку.

Событие может быть определено как активатор 228 какой-либо привязки. "Активатор" так и называется, поскольку запускает активность привязки - запись данных с виджета в контекст на сервере, чтение данных из контекста и использование их в качестве контента инструментальной панели, либо выполнение еще какой-либо обработки данных.

Допустим, у нас есть текстовое поле **Name**, связанное с полем в какой-либо переменной контекста сервера (например, имя пользователя). Его привязка также определяет событие **активатор** -- событие **действие** кнопки **Save**, расположенной тут же рядом. Когда пользователь нажимает на эту кнопку, происходит следующая последовательность событий:

1) Все привязки виджета сканируются, чтобы найти их события- активаторы.

2) Система обнаруживает, что у привязки для текстового поля **Name** это событие **действие** определено как активатор привязки.

3) Привязка выполняется одновременно с другими привязками, имеющими такое же событие-активатор (например, **Last Name**, при наличии). Проще говоря, данные из текстовых полей теперь записаны в базу данных.

Более подробно см. в разделе модель данных инструментальной панели 223.

7.9.3.1 Общие события компонентов

В разделе описываются события, поддерживаемые большинством компонентов. Описание каждого отдельного компонента включает список поддерживаемых общих событий.

НАЖАТИЕ МЫШИ

Данное событие возникает, когда пользователь кликает мышью по компоненту.

Имя события:	mouseClicked
Записи:	1

Формат 50 записи:

Имя поля	Тип поля	Примечания

id	String	ID события.
value	Data Table	Содержит информацию, относящуюся к событию. Эта вложенная таблица данных имеет одно поле: • **ID** - имя соответствующего компонента, например, *image*.
level	Integer	Уровень 75 события по умолчанию.

ИЗМЕНЕНИЕ ВХОДНОГО ЗНАЧЕНИЯ

Данное событие возникает, когда пользователь вводит данные в поле области ввода.

Имя события: inputChange

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	ID события.
value	Data Table	Содержит информацию, относящуюся к событию. Эта вложенная таблица данных имеет два поля: • **ID** - имя соответствующего компонента, например, *textField0*. • **Value** - текущее значение, введенное в поле ввода.
level	Integer	Уровень 75 события по умолчанию.

7.9.4 Контейнеры

Основное различие между *контейнерами* и регулярными компонентами заключается в том, что контейнеры могут содержать внутри себя другие компоненты и контейнеры.

В каждом контейнере имеется одно или более *подокон,* в которых можно разместить дочерние компоненты. Например, контейнер Панель 246 имеет единственное, всегда видимое подокно без каких-либо декораций. Контейнер Панель со вкладками 246 может иметь любое количество подокон (вкладок), содержащих дочерние компоненты. Каждое подокно имеет отдельную компоновку и может содержать несколько дочерних компонентов или контейнеров.

Базовый компонент веб инструментальной панели - ее корневая панель 241. Этот компонент служит контейнером для всех остальных компонентов инструментальной панели. Когда инструментальная панель запущена, ее корневая панель занимает все доступное пространство внутри инструменатльной панели.

7.9.4.1 Компоновка контейнеров

Каждое подокно в контейнере имеет свою компоновку дочерних компонентов. Существуют два типа компоновки:

- Сетка 241
- Абсолютное 241 позиционирование

Изменить компоновку можно редактируя свойство **Layout** подокна.

Если вы хотите создать инструментальную панель изменяемого размера, используйте компоновку Сетка.

Ограничения компонентов

Ограничения компонента инструментальной панели определяют его положение и размер внутри родительского контейнера. Ограничения также определяют, как компоненты меняют размер и положение при изменении размера окна инструментальной панели.

Каждая компоновка предполагает собственные ограничения для каждого компонента инструментальной панели, за исключением корневого подокна.

7.9.4.1.1 Сетка

Данная компоновка представляет собой сетку из рядов и столбцов, позволяя некоторым компонентам занимать несколько рядов или столбцов. Ряды могут быть разной высоты, так же и столбцы могут быть разной ширины. При редактировании инструментальной панели в Конструкторе UI 348 эту сетку можно отображать или скрывать, но при запущенном виджете она никогда не видна. Отображаются только сами компоненты.

Компоновка Сетка поддерживается и отрисовывается с помощью технологии *CSS Grid Layout*.

7.9.4.1.2 Абсолютное позиционирование

Данная компоновка позволяет определить точное положение и размер для каждого компонента, то есть их X и Y координаты, ширину и высоту на уровне пикселей. Это может быть очень удобно для создания человеко-машинных интерфейсов (HMI), планов зданий/этажей и прочих подобных интерфейсов. Компоненты в абсолютной компоновке могут легко перекрывать друг друга.

7.9.4.2 Корневая панель

Корневая панель - это основной контейнер инструментальной панели для всех других компонентов и контейнеров.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

НАСЛЕДУЕМЫЕ ПЕРЕМЕННЫЕ (СВОЙСТВА)

Компорент корневая панель является контейнером 240, поэтому наследует общие свойства контейнеров 235.

ШИРИНА МАКЕТА

Ширина макета для абсолютного позиционирования, в пикселях.

Имя переменной: layoutWidth

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
layoutWidth	Integer	Ширина макета

ВЫСОТА МАКЕТА

Высота макета для абсолютного позиционирования, в пикселях

Имя переменной: layoutHeight

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания

layoutHeight	String	Высота макета

ПОЛЬЗОВАТЕЛЬСКИЕ СВОЙСТВА

Эта переменная позволяет добавлять пользовательские свойства (переменные) в корневую панель.

Имя переменной: customProperties

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
name	String	**Name**. Имя пользовательского свойства.
format	Data Table	**Format**. Формат пользовательского свойства. Более подробно см. формат таблицы 2053.
description	String	**Description**. Описание пользовательского свойства.
help	String	**Help**. Справочное сообщение пользовательского свойства.

ПРИВЯЗКИ

Эта переменная содержит все доступные для инструментальной панели привязки. Более подробно см. в разделе Привязки 226.

Имя переменной: bindings

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
target	String	**Target.** Цель привязки.
expression	String	**Expression**. Выражение привязки.
activator	String	**Activator**. Активатор привязки.
condition	String	**Condition.** Выражение условия привязки.
onstartup	Boolean	**On Startup**. Если включено, привязка будет выполняться после полной загрузки инструментальной панели. Более подробно см. Жизненный цикл инструментальной панели 223.
onevent	Boolean	**On Event**. Если включено, привязка будет выполняться при возникновении события, указанного в поле *Activator* переменной *Привязки*.
periodically	Boolean	**Periodically**. Если включено, привязка будет выполняться постоянно по истечении периода времени, указанного в поле *Period* переменной *Привязки*.
period	Long	**Period**. Период времени для периодического выполнения привязки.

ПАРАЛЛЕЛЬНАЯ РАБОТА ПРИВЯЗОК ПРИ ЗАПУСКЕ

Если включено, каждая привязка будет обрабатываться в отдельном потоке. Более подробно см. Производительность привязки 229.

Имя переменной: startupBindingsConcurrency

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
startupBindingsConcurrency	Boolean	Параллельная работа привязок при запуске

КОЛИЧЕСТВО ПАРАЛЛЕЛЬНО РАБОТАЮЩИХ ПРИВЯЗОК

Максимальное количество заданий на обработку привязок при нормальных условиях. Если необходимо начать выполнение большего количества заданий на обработку, задачи сверх указанного количества помещаются в очередь на обработку по мере освобождения потоков.

Имя переменной: normalConcurrentBindings

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
normalConcurrentBindings	Integer	Количество параллельно работающих привязок

МАКСИМУМ ПАРАЛЛЕЛЬНО РАБОТАЮЩИХ ПРИВЯЗОК

Абсолютный максимум числа потоков в пуле обработки привязок. Если данное максимальное число достигнуто, а еще имеются задачи на немедленное выполнение, такие задачи будут либо не выполнены, либо заблокированы до высвобождения ресурсов какого-либо потока.

Имя переменной: maximumConcurrentBindings

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
maximumBindingQueueLength	Integer	Максимум параллельно работающих привязок

МАКСИМАЛЬНАЯ ДЛИНА ОЧЕРЕДИ НЕОБРАБОТАННЫХ ПРИВЯЗОК

Максимальное число задач на обработку привязок, которые могут быть поставлены в очередь. При переполнении очереди будут создаваться дополнительные потоки обработки привязок в пуле до тех пор, пока не будет достигнут Максимум параллельно работающих привязок.

Имя переменной: maximumBindingQueueLength

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания

maximumBinding QueueLength	Integer	Максимальная длина очереди необработанных привязок

Общие события [? 239]

Общие события: Нажатие мыши

ПОКАЗ

Событие возникает сразу после отображения веб инструментальной панели при запуске или после ее извлечения из кэша 224. Более подробно см. Жизненный цикл инструментальной панели 223.

Имя события: shown

Записи: 0

НЕДЕЙСТВИТЕЛЬНО

Событие используется только во время редактирования инструментальной панели. Не стоит использовать его для управления нормальными операциями инструментальной панели.

Имя события: invalidated

Записи: 0

ПЕРЕАДРЕСОВАНО

Событие возникает при выполнении операции переадресация 228 из привязки.

Имя события: redirected

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
dashboard	String	Путь контекста инструментальной панели 1490, к которому ведет операция переадресации.
mode	String	Режим работы инструментальной панели: редактирование, просмотр или запущена.
historyUrlChanged	Boolean	Определяет, был ли изменен URL в истории браузера во время переадресации.
parameters	Data Table	Эта таблица содержит все параметры для вызова операции переадресации, по одному параметру в каждом ряду.

ДЕЙСТВИЕ ЗАПУЩЕНО

Событие возникает при запуске действия 87 из инструментальной панели.

Имя события: actionLaunched

Записи: 1

Формат 50 записи:

Имя	Тип поля	Примечания

поля		
reference	String	Ссылка 117, используемая для запуска действия.
parameters	Data Table	Параметры действия.
actionId	String	ID действия.

ПРИВЯЗКА ОБРАБОТАНА

Событие возникает после успешной обработки привязки инструментальной панели. В настоящее время, данное событие возникает только в режиме предпросмотра инструментальной панели.

Имя события: bindingProcessed

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
timestamp	Date	Дата/время обработки.
bindingNumber	Integer	Номер привязки в списке привязок, по сути, ID привязки.
target	String	Цель привязки.
expression	String	Выражение привязки.
method	Integer	Метод вычисления привязки: при запуске, при событии или периодически.
cause	String	Причина привязки.
evaluationOptions	String	Опции вычисления привязки, включающие активатор и условие привязки.
result	String	Результат вычисления привязки.

ОШИБКА ПРИВЯЗКИ

Событие возникает, если привязка инструментальной панели обработана с ошибкой. В настоящее время, данное событие возникает только в режиме предпросмотра инструментальной панели.

Имя события: bindingError

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
timestamp	Date	Дата/время обработки.
bindingNumber	Integer	Номер привязки в списке привязок, по сути, ID привязки.
target	String	Цель привязки.

expression	String	Выражение привязки.
method	Integer	Метод вычисления привязки: при запуске, при событии или периодически.
cause	String	Причина привязки.
error	String	Текст ошибки привязки.
stacktrace	String	Трассировка стека ошибки.

7.9.4.3 Панель

Панель - это простой контейнер с отдельной компоновкой, который может содержать другие компоненты или контейнеры.

Общие переменные (свойства) [?231]

Общие переменные: Имя232, Видимый232, Свойства расширения232, Выход за пределы контента233, Стиль контейнера234, Пользовательские классы234, Стиль234

НАСЛЕДУЕМЫЕ ПЕРЕМЕННЫЕ (СВОЙСТВА)

Компонент Панель является контейнером240, поэтому наследует общие свойства контейнеров235.

Общие события [?239]

Общие события: Нажатие мыши239

7.9.4.4 Панель с вкладками

Панель с вкладками позволяет пользователю переключаться между компонентами в группе, кликая на вкладку с определенным заголовком.

Общие переменные (свойства) [?231]

Общие переменные: Имя232, Видимый232, Свойства расширения232, Выход за пределы контента233, Стиль контейнера234, Пользовательские классы234, Стиль234

ПОЗИЦИЯ ВКЛАДОК

Определяет позицию вкладок на панели с вкладками. Это свойство имеет несколько допустимых значений:

- Сверху
- Снизу
- Слева
- Справа

Имя переменной: tabPosition

Записи: 1

Формат50 записи:

Имя поля	Тип поля	Примечания

tabPosition	String	Позиция вкладок

ТИП ВКЛАДОК

Определяет визуальное представление вкладок. Это свойство имеет два возможных значения:

- Строка
- Карточка

Имя переменной: tabType

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
tabType	String	Тип вкладок

АКТИВНАЯ ВКЛАДКА

Имя вкладки, которая будет открываться по умолчанию при загрузке инструментальной панели.

Имя переменной: activeTab

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
activeTab	String	Активная вкладка

КНОПКИ

Эта переменная позволяет добавлять настраиваемые кнопки на панель с вкладками.

Имя переменной: externalButtons

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	**ID**. Уникальный ID кнопки.
description	String	**Description**. Текст, отображаемый на кнопке
image	Data Block	**Image**. Изображение кнопки.
Style	String	**Style**. CSS стиль для применения к этой кнопке

СКРЫТЬ ПОЛОСУ ТАБУЛЯЦИИ

Если включено, скрывает полосу переключения между вкладками.

Имя переменной: hideBar

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
hideBar	Boolean	Скрыть полосу табуляции

СТИЛЬ ДОПОЛНИТЕЛЬНОГО БЛОКА

CSS стиль для применения к дополнительному блоку на панели с вкладками.

Имя переменной: defaultContext

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
context	String	Стиль дополнительного блока.

МАРШРУТИЗИРУЕМЫЙ

Если включено, разрешает маршрутизацию между вкладками. Более подробно см. URL и маршрутизация 225.

Имя переменной: routable

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
routable	String	Маршрутизируемый

ОБщие события [? 239]

НАЖАТИЕ НА КНОПКУ

Это событие возникает, когда пользователь кликает мышью по кнопке, определенной в таблице **Кнопки.**

Имя события: buttonClicked

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	**ID**. уникальный ID нажатой кнопки.

НАЖАТИЕ НА ПАНЕЛЬ ВКЛАДОК

Это событие возникает при выборе новой вкладки.

Имя события: tabPanelClicked

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	**ID**. уникальный ID выбранной вкладки.

7.9.4.4.1 Вкладка

Каждая вкладка включает собственную компоновку, которая может содержать другие компоненты или контейнеры.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

НАСЛЕДУЕМЫЕ ПЕРЕМЕННЫЕ (СВОЙСТВА)

Каждая вкладка является контейнером 240, поэтому наследует общие свойства контейнеров 235.

ЗАГОЛОВОК

Заголовок текущей вкладки.

Имя переменной: title

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
title	String	Заголовок.

ПИКТОГРАММА

Изображение, отображаемое рядом с заголовком вкладки.

Имя переменной: tabImage

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
tabImage	Data Block	Изображение.

ВКЛАДКА ОТКЛЮЧЕНА

Если владка отключена, она будет недоступной для выбора.

Имя переменной: tabDisabled

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
tabDisabled	Boolean	Вкладка отключена

СКРЫТЬ ВКЛАДКУ

Если включено, вкладка будет скрыта в списке вкладок.

Имя переменной: tabHidden

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
tabHidden	Boolean	Вкладка скрыта

БЕЙДЖ

Текст для отображения в бейдже вкладки.

Имя переменной: tabBadgeVariable

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
tabBadgeVariable	String	Бейдж.

ИНДЕКС

Определяет положение вкладки в списке вкладок.

Имя переменной: index

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
index	Integer	Индекс

Общие события [? 239]

Общие события: Нажатие мыши 239

7.9.4.5 Панель с шагами

Степпер - это контейнер, который позволяет отображать прогресс через последовательность логических нумерованных шагов. Он обычно используется для создания мастеров настройки и других пошаговых UI процедур.

Общие переменные (свойства) [?231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

КОЛИЧЕСТВО

Количество шагов, добавленных в настоящее время в степпер.

Имя переменной: stepsCount

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
stepsCount	Integer	Количество

ТЕКУЩИЙ ШАГ

Порядковый номер текущего шага.

Имя переменной: currentStep

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
currentStep	Integer	Текущий шаг.

НАПРАВЛЕНИЕ

Определяет направление степпера. Это свойство имеет два возможных значения:

- Горизонтальное
- Вертикальное

Имя переменной: stepsDirection

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
stepsDirection	String	Направление

ИМЯ КНОПКИ "ГОТОВО"

Текст кнопки "Готово".

Имя переменной: doneButtonName

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
doneButtonName	String	Имя кнопки "Готово"

ИМЯ КНОПКИ СЛЕДУЮЩЕГО ШАГА

Текст кнопки "Далее".

Имя переменной: nextButtonName

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
nextButtonName	String	Имя кнопки "Далее".

ИМЯ КНОПКИ ПРЕДЫДУЩЕГО ШАГА

Текст кнопки "Назад".

Имя переменной: previousButtonName

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
previousButtonName	String	Имя кнопки "Назад".

КОНЕЧНОЕ СООБЩЕНИЕ

Текст всплывающего сообщения после нажатия на кнопку "Готово".

Имя переменной: doneMessage

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
doneMessage	String	Конечное сообщение

ПРЕДСТАВЛЕНИЕ ШАГОВ

Определяет визуальное представление степпера. Это свойство имеет два возможных значения:

- С иконками
- С точками

Имя переменной: stepsView

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
stepsView	String	Представление шагов

КЛИК ПО ШАГАМ ВКЛЮЧЕН

Если включено, активирует навигацию через клики по шагам.

Имя переменной: stepsClickable

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
stepsClickable	Boolean	Клик по шагам включен

Общие события [? 239]

НАЖАТИЕ НА КНОПКУ "ГОТОВО"

Это событие возникает, когда пользователь кликает мышью по кнопке "Готово".

Имя события: doneButtonClick

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	ID события.
value	String	Содержит информацию о событии. Вложенная таблица данных с одним полем: • **ID** - имя соответствующего компонента, например, *steps0*.
level	Integer	Уровень 75 события по умолчанию.

7.9.4.5.1 Шаг

Каждый шаг содержит отдельную компоновку, которая может включать другие компоненты или контейнеры.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

НАСЛЕДУЕМЫЕ ПЕРЕМЕННЫЕ (СВОЙСТВА)

Каждый шаг является контейнером 240, поэтому наследует общие свойства контейнеров 235.

ЗАГОЛОВОК

Заголовок шага.

Имя переменной: stepTitle

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
stepTitle	String	Заголовок

ОПИСАНИЕ

Описание шага.

Имя переменной: stepDescription

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
stepDescription	Data Block	Описание

СТАТУС

Определяет статус текущего шага. Это свойство имеет несколько возможных значений:

- По умолчанию
- Ожидание
- В процессе
- Завершение
- Ошибка

Имя переменной: stepStatus

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
stepStatus	String	Статус

ИНДЕКС

Определяет индекс шага.

Имя переменной: index

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания

index	Integer	Индекс

ПИКТОГРАММА

Изображение, отображаемое рядом с заголовком шага.

Имя переменной: stepIcon

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
stepIcon	Data Block	Пиктограмма

Общие события [? 239]

Общие события: Нажатие мыши 239

7.9.4.6 Вложенная панель

Вложенная инструментальная панель - это контейнер, используемый для отображения одной инструментальной панели внутри другой.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

ССЫЛКА

Путь к контексту инструментальной панели, соответствующей панели для загрузки и отображения внутри данного компонента.

Имя переменной: reference

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
reference	String	Ссылка

КОНТЕКСТ ПО УМОЛЧАНИЮ

Путь контекста, который станет контекстом по умолчанию 946 для вложенной панели.

Имя переменной: defaultContext

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
context	String	Контекст по умолчанию

Общие события [? 239]

Общие события: Изменение входного значения 239

7.9.5 Редактор свойств

Редактор свойств используется для изменения свойств различных контекстов 41. Например, настройки самого AtomMind Server и устройство 497 редактируются в Редакторе свойств.

Технически, Редактор свойств позволяет изменять значения одной и более переменных 61 (свойств) контекста. При запуске для определенного контекста, он загружает значение каждой переменной контекста, затем пользователь может изменить данные значения и записать их снова в контекст.

Каждое свойство (переменная) редактируется в отдельном компоненте Редактор таблицы данных 282, потому что свойство представляет собой Таблицу данных 49.

Редактор свойств состоит из панели инструментов и подокна Свойства.

панель инструментов

	Перезагрузить. Перезагружает значения всех свойств из контекста-источника. Значения, которые были недавно изменены в редакторе, теряются.
	Сохранить. Сохраняет значения измененных свойств в контекст-источник. Сохраненные свойства отмечаются как неизмененные.
	Импорт свойств. Импортирует 382 значения свойств из файла.
	Экспорт свойств. Экспортирует 382 значения свойств в файл.

Если Редактор свойств открывается на отдельной странице, он имеет кнопки **ОК** и **Отмена.** Кнопка **ОК** сохраняет значения всех измененных свойств и закрывает диалоговое окно. Кнопка **Отмена** прекращает операцию без сохранения.

Большинство элементов в редакторе свойств, так же как и актуальные свойства, имеют всплывающие подсказки. Они появляются при наведении и удержании в течение некоторого времени курсора мыши на элементе.

Контекстное меню

Контекстное меню появляется при нажатии правой кнопкой мыши на одном из свойств в Редакторе. Оно содержит список действий, относящихся к переменной 102, которые "знают", что делать с выбранной переменной. Количество и тип доступных действий, относящихся к переменной, зависят от типа переменной, для которой отображается контекстное меню.

Режимы редактора свойств

Редактор свойств может работать в двух режимах:

- Нормальный
- Только чтение

Режим "Только чтение" не позволяет изменять или сохранять свойства.

Существуют также два режима представления Редактора свойств:

- Простой
- Расширенный

Режимы представления свойства

ПРОСТОЙ РЕЖИМ

В *простом режиме*, каждый Редактор таблицы данных, представляющий значение одного свойства, занимает отдельную вкладку. На рисунке ниже, редактор таблицы данных, содержащий значение одного свойства, отмечен красным прямоугольником.

Имя свойства и подробное описание отображаются во всплывающей подсказке для вкладки.

РАСШИРЕННЫЙ РЕЖИМ

В *расширенном режиме* свойства группируются согласно их группе переменных 61. Свойства каждой группы отображаются в отдельной вкладке. Имена групп отображаются в виде заголовков вкладок.

Свойства каждой группы представлены в виде таблицы с двумя или тремя столбцами. Первый столбец является дополнительным и может содержать иконку статуса свойства. Редактор свойств постоянно контролирует статус каждой переменной и обновляет иконку во время каждого изменения. При наведении курсора мыши на иконку появляется всплывающая подсказка с информацией о статусе.

Столбец Свойство содержит описания свойства. Всплывающие подсказки для ячеек таблицы представляют информацию об имени каждого свойства и его подробное описание.

Столбец **Значение** содержит изменения согласно определению переменной 61 изменяемого контекста. Если Таблица данных 49, представляющая значение свойства, всегда состоит из одного поля с одной записью, Редактор таблицы данных 282 отображается в третьем столбце. В других случаях, третий столбец будет содержать кнопку **[...]**, которая открывает Редактор таблицы данных на отдельной странице. Встроенный в таблицу Редактор таблицы данных отмечен красным.

Импорт и экспорт свойств

Свойства могут быть экспортированы и импортированы из внешних файлов. По умолчанию, файлы свойств имеют разрешение `.prs`. Свойства импортируются по имени: если Редактор свойств содержит свойство с таким же именем, что и свойство, сохраненное в файле, их значения объединяются во время выполнения операции Интеллектуальное копирование таблицы данных.

Процедура импорта свойств не может перезаписать значения в редакторе на полученные при чтении файла, т.к они могут иметь различный формат 49. Однако, в любом случае делается все возможное, чтобы импортировался как можно больший объем данных.

7.9.6 Селектор объектов

Компонент *Селектор объектов* используется для выбора контекстов 41, переменных 61, функций 70, событий 73 AtomMind Server и даже определенных **полей** внутри них.

Чтобы увидеть разницу между деревом контекстов в Селекторе объектов и деревом контекстов сервера нажмите здесь 45.

Содержание может фильтроваться таким образом, что для выборки отображаются только определенные типы объектов.

Применение Селектора объектов

1. Селектор объектов используется UI процедурой Выбрать объекты 94 для выбора различных контекстов и их элементов.
2. Селектор объектов используется для выбора контекстов и масок контекстов при редактироании полей путей/масок контекстов в Редакторе таблицы данных 282.

Иконки в Селекторе объектов

Если определены пользовательские иконки для различных контекстов и их частей, они отображаются в дереве Селектора объектов. Во всех других случаях используются следующие иконки:

	"Закрытый" контекст (для свернутых узлов).
	"Открытый" контекст (для раскрытых узлов).
	Переменная (Свойство).
	Функция (имейте в виду, что функции отличаются от действий 87).
	Событие.

	Поле входа (для ввода значений переменных, событий и функций). Отображается описание и тип поля.
	Поле выхода (используется функциями). Отображается описание и тип поля.

7.9.7 Навигация

Данный раздел описывает компоненты **Навигации** инструментальной панели.

7.9.7.1 Меню Навигации

Данный компонент отображает список вариантов выбора для временных поверхностей.

Общие переменные (свойства) [?231]

Общие переменные: Имя232, Видимый232, Свойства расширения232, Ширина сетки233, Высота сетки233, Выход за пределы контента233, Стиль контейнера234, Пользовательские классы234, Стиль234

ВКЛЮЧИТЬ СВЕРНУТОГО МЕНЮ

Имя переменной: inlineCollapsed

Записи: 1

Формат50 записи:

Имя поля	Тип поля	Примечания
inlineCollapsed	Boolean	Если включено, свернутое меню в режиме *Встроенного* меню будут открываться горизонтально, а не вертикально.

ШИРИНА СВЕРНУТОГО МЕНЮ

Имя переменной: toggledMenuWidth

Записи: 1

Формат50 записи:

Имя поля	Тип поля	Примечания
toggledMenuWidth	Integer	Определяет максимальную ширину элементов свернутого меню в режиме *Встроенного* меню.

ЭЛЕМЕНТЫ МЕНЮ

Данное свойство определяет иерархию и визуальное представление элементов меню.

Имя переменной: menuItems

Записи: 0..не ограничено

Формат50 записи:

Имя поля	Тип поля	Примечания
image	Data Block	**Image**. Изображение иконки элемента меню.

key	String	**Key**. Уникальный ключ элемента меню.
value	String	**Value**. Отображаемое имя элемента меню.
disabled	String	**Disabled**. Отключает определенный элемент меню. Отключенные элементы выделяются серым цветом и недоступны для выбора.
parent	String	**Parent Key**. Ключ родительского элемента меню.

ВЫБРАННЫЕ ЭЛЕМЕНТЫ

Данное свойство определяет ключи элементов меню, которые будут отображаться в выбранном состоянии.

Имя переменной: selectedKeys

Записи: 0..не ограничено

Формат записи:

Имя поля	Тип поля	Примечания
key	String	Ключ

РАЗВЕРНУТЫЕ ЭЛЕМЕНТЫ

Данное свойство определяет ключи элементов меню, которые будут отображаться в развернутом состоянии.

Имя переменной: openKeys

Записи: 0.не ограничено

Формат записи:

Имя поля	Тип поля	Примечания
key	String	Key

РЕЖИМ МЕНЮ

Определяет режим отображения меню:

- Вертикальный
- Горизонтальный
- Встроенный

Имя переменной: mode

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
mode	String	Режим меню

МНОЖЕСТВЕННЫЙ ВЫБОР

Если включено, разрешает множественный выбор элементов меню.

Имя переменной: multiple

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
multiple	Boolean	Множественный выбор

ТЕМА МЕНЮ

Определяет преднастроенный визуальный стиль меню. Доступны следующие стили:

- Светлый
- Темный

Имя переменной: theme

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
theme	String	Тема меню

ЗАДЕРЖКА ОТКРЫТИЯ ПОДМЕНЮ ПРИ НАВЕДЕНИИ

Определяет задержку перед открытием свернутого меню при наведении курсора мыши на соответствующий элемент меню.

Имя переменной: subMenuOpenDelay

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
subMenuOpenDelay	Long	Задержка открытия подменю при наведении

ЗАДЕРЖКА ЗАКРЫТИЯ ПОДМЕНЮ

Определяет задержку перед закрытием свернутого меню после отведения курсова мыши с элемента меню.

Имя переменной: subMenuCloseDelay

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
subMenuCloseDelay	Long	Задержка закрытия подменю

ВОЗМОЖНОСТЬ ВЫБОРЫ

Если включено, разрешает выбор элементов с использование флажков.

Имя переменной: selectable

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
selectable	Boolean	Возможность выбора

ТРИГГЕР ПОДМЕНЮ

Определяет действие по умолчанию, которое открывает свернутый элемент меню. Данное свойство имеет два возможных значения:

- Нажатие
- Наведение

Имя переменной: triggerSubMenuAction

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
triggerSubMenuAction	String	Триггер подменю

МАКСИМАЛЬНЫЙ ЗНАЧЕНИЕ СЧЕТЧИКА

Определяет максимальное значение счетчика, отображаемое на значке элемента меню. Если значение значка больше указанного в данном свойстве, счетчик будет отображаться как значение поля Максимальный значение счётчика со знаком плюс, например, "10+".

Имя переменной: overflowCount

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
overflowCount	Integer	Максимальный значение счетчика

СМЕЩЕНИЕ ПО ОСИ X

Определяет горизонтальное смещение значка элемента меню, в пикселях.

Имя переменной: offsetX

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
offsetX	Integer	Смещение по оси x

СМЕЩЕНИЕ ПО ОСИ Y

Определяет вертикальное смещение значка элемента меню, в пикселях.

Имя переменной: syncTime

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
syncTime	Integer	Смещение по оси y

ОТОБРАЖЕНИЕ НУЛЯ

Если включено, нулевые значения также будут отображаться на значках.

Имя переменной: showZero

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
showZero	Boolean	Отображение нуля

ПОЛЬЗОВАТЕЛЬСКИЕ ЗНАЧКИ

Данное свойство позволяет определить отображение пользовательских значков на соответствующих элементах меню.

Имя переменной: menuCustomBadges

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
key	String	**Key**. Уникальный ID значка.
color	String	**Color**. Цвет значка.
count	Integer	**Count**. Числовое значение для отображения на значке.
dot	Boolean	**Dot**. Если включено, заменяет значок на небольшую статичную точку.
status	String	**Status**. Если выбрано, заменяет значок на небольшую статичную точку, которая может отображаться с набором статусов. Данное свойство имеет несколько возможных значений: • **Not Set** • **Default** • **Success** • **Processing** • **Error** • **Warning** Обратите внимание, что для использования данного свойства поле *dot* должно быть выключено.
statusText	String	**Status Text**. Определяет текст, отображаемые справа от точки.

tooltipText	String	**Count Tooltip Text**. Определяет текст всплывающей подсказки для значка.

Общие события [?239]

Общие события: Нажатие мыши 239

ВНЕСЕНИЕ ИЗМЕНЕНИЙ

Данное событие возникает, когда пользователь меняет выборку элементов меню в текущем сворачиваемом меню.

Имя события: mouseOpenChange

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	ID события.
value	Data Table	Содержит информацию о событии. Вложенная таблица данных с одним полем: • **ID** - имя соответствующего компонента, например, *image*.
level	Integer	Уровень 75 события по умолчанию.

ВЫБРАН

Данное событие возникает, когда пользователь выбирает определенный элемент меню.

Имя события: mouseSelect

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	ID события.
value	Data Table	Содержит информацию о событии. Вложенная таблица данных с одним полем: • **ID** - имя соответствующего компонента, например, *image*.
level	Integer	Уровень 75 события по умолчанию.

ОТМЕНА ВЫБОРА

Данное событие возникает, когда пользователь отменяет выбор определенного элемента меню.

Имя события: mouseDeSelect

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	ID события.
value	Data Table	Содержит информацию о событии. Вложенная таблица данных с одним полем: • **ID** - имя соответствующего компонента, например, *image*.
level	Integer	Уровень 75 события по умолчанию.

7.9.7.2 Навигационные цепочки

Навигационная цепочка отображает текущее расположение внутри иерархии. Это позволяет вернуться к состояниям выше в иерархии.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

ДАННЫЕ

Данное свойство определяет структуру навигационной цепочки и ее визуальное представление.

Имя переменной: data

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	**ID**. Уникальный ID навигационной цепочки.
name	String	**Name**. Отображает имя навигационной цепочки.
index	String	**Index**. Индекс навигационной цепочки.

ТЕКУЩИЙ ИНДЕКС

Определяет, какая навигационная цепочка отображается как активная. Данное свойство должно совпадать с полем **Index** свойства **Data**.

Имя переменной: currentBreadcrumb

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
currentBreadcrumb	String	Текущий индекс

РАЗДЕЛИТЕЛЬ

Изображение, используемое в качестве разделителя между навигационными цепочками.

Имя переменной: separator

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
separator	Data Block	Разделитель

Общие события [? 239]

НАЖАТИЕ НА НАВИГАЦИОННУЮ ЦЕПОЧКУ

Данное событие возникает, когда пользователь нажимает на определенную часть навигационной цепочки.

Имя события: breadcrumbClicked

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	ID события.
value	Data Table	Содержит информацию о событии. Вложенная таблица данных с одним полем: • **ID** - ID нажатой навигационной цепочки.
level	Integer	Уровень 75 события по умолчанию.

7.9.7.3 Карта

Данный компонент отображает географическую карту, дорожную карту, карту местности или спутниковую карту. Компонент позволяет создавать множество слоев, которые показывают и динамически обновляют геозоны, устройства или пользовательские объекты. Карта может использовать множество тайловых источников геоданных и изображений (например, Google Maps, Bing Maps, OpenStreetMap или Yandex Maps).

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

МАСШТАБ

Определяет масштаб карты по умолчанию

Имя переменной: zoom

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
zoom	Integer	Масштаб

СЛОИ

Данное свойство позволяет определить слои карты и управлять их видимостью.

Имя переменной: layers

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	**ID**. Уникальный ID слоя.
name	String	**Name**. Имя слоя.
description	String	**Description**. Описание слоя.
show	Boolean	**Show**. Определяет видимость слоя.

МАРКЕРЫ

Данное свойство позволяет определить отображение маркеров на выбранном слое карты и управлять их визуальным представлением.

Имя переменной: markers

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	**ID**. Уникальный ID маркера.
layer	String	**Layer**. ID слоя карты для отображения маркера.
latitude	Double	**Latitude**. Широта маркера.
longitude	Double	**Longitude**. Долгота маркера.
riseOnHover	Boolean	**Rise On Hover**.
riseOffset	Integer	**Z-index Rise Offset**.
draggable	Boolean	**Draggable**.
popupOptions	Data Table	**Popup Options**.
tooltipOptions	Data Table	**Tooltip Options**.
style	String	**Style**. Стиль CSS, который будет применен к маркеру.

ПУТИ

Данное свойство позволяет определить отображение путей на выбранном слое карты и управлять их визуальным представлением.

Имя переменной: tracks

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	**ID**. Уникальный ID пути.
polyline	Data Table	**Polyline**. Определяет точки, формирующие ломаную линию. Вложенная таблица данных с несколькими полями: • **Latitude -** широта точки • **Longitude -** долгота точки • **Description -** описание точки
description	String	**Description**. Описание пути.
layerId	String	**Layer**. ID слоя карты для отображения пути.
polylineStyleId	String	**Polyline Style**. Стиль CSS, который будет применен к ломаной линии.
circleMarkerStyleId	String	**Point Style**. Стиль CSS, который будет применен к круговому маркеру.

ОБЛАСТИ

Данное свойство позволяет определить отображение областей на выбранном слое карты и управлять их визуальным представлением.

Имя переменной: areas

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	**ID**. Уникальный ID области.
polyline	Data Table	**Polyline**. Определяет точки, формирующие область. Вложенная таблица данных с несколькими полями: • **Latitude -** широта точки • **Longitude -** долгота точки • **Description -** описание точки
description	String	**Description**. Описание области.
layerId	String	**Layer**. ID слоя карты для отображения области.
areaStyleId	String	**Style**. Стиль CSS, который будет применен к области.

КРУГЛЫЕ ОБЛАСТИ

Данное свойство позволяет определить отображение круглых областей на выбранном слое карты и управлять их визуальным представлением.

Имя переменной: circleAreas

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания

id	String	**ID**. Уникальный ID круглой области.
description	String	**Description**. Описание круглой области.
latitude	Double	**Latitude**. Широта круглой области.
longitude	Double	**Longitude**. Долгота круглой области.
radius	Integer	**Radius**. Радиус круглой области, в пикселях.
layerId	String	**Layer**. ID слоя карты для отображения круглой области.
circleAreaStyleId	String	**Style**. Стиль CSS, который будет применен к круглой области.

УПРАВЛЕНИЕ ПРИБЛИЖЕНИЕМ

Если включено, отображает кнопки управления приближением.

Имя переменной: zoomControl

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
zoomControl	Boolean	Управление приближением

ПОКАЗАТЬ СТРОКУ ПОИСКА

Если включено, отображает строку поиска.

Имя переменной: enableSearch

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
enableSearch	Boolean	Показать строку поиска

КЛЮЧ ДОСТУПА ДЛЯ ТАЙЛОВ

Пользовательский ключ доступа, который будет добавляться к каждому запросу тайла.

Имя переменной: accessToken

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
accessToken	String	Ключ доступа для тайлов

ШИРОТА

Определяет широту фиксированного центра карты.

Имя переменной: latitude

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
latitude	Double	Широта

ДОЛГОТА

Определяет долготу фиксированного центра карты.

Имя переменной: longitude

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
longitude	Double	Долгота

ВЫБРАННЫЙ МАРКЕР

Указывает маркер, выбираемый по умолчанию. Значение данного поля должно соответствовать полю **ID** свойства **Markers**.

Имя переменной: selectedMarker

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
selectedMarker	String	Выбранный маркер

ССЫЛКА НА КАРТУ

URL источника тайлов карты.

Имя переменной: mapLink

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
mapLink	String	Ссылка на карту

ПРОВАЙДЕР ПОИСКА ПО КАРТЕ

Определяет провайдера поиска по карте для строки поиска. Данной свойство имеет два возможных значения:

- **Nominatim**
- **Latitude Longitude**

Имя переменной: mapProvider

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания

mapProvider	String	Провайдер поиска по карте

КНОПКА ЦЕНТРИРОВАНИЯ НА МАРКЕРЕ

Если включено, на маркере, указанном в свойстве **Selected Marker,** отображается карта с указанием центра кнопки.

Имя переменной: centerMap

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
centerMap	Boolean	Кнопка центрирования на маркере

ПОКАЗЫВАТЬ ПОЛЬЗОВАТЕЛЬСКУЮ КНОПКУ

Если включено, отображает пользовательскую кнопку в настраиваемым поведением.

Имя переменной: customButton

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
customButton	Boolean	Показывать пользовательскую кнопку

ПРОКСИ

Если включено, все ответы на запросы тайлов будут идти через AtomMind как прокси-сервер.

Имя переменной: proxy

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
proxy	Boolean	Прокси

СТИЛИ МАРКЕРОВ

Определяет стили и опции визуализации маркеров.

Имя переменной: markersStyles

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	**ID**. Уникальный ID маркера. Значение данного поля должно соответствовать полю **ID** свойства **Markers**.
name	String	**Name**. Имя предустановленного стиля.
description	String	**Description**. Описание предустановленного стиля.

selectedMarkerImage	Data Block	**Selected Marker Image**. Изображение маркера в выбранном состоянии.
defaultMarkerImage	Data Block	**Default Marker Image**. Изображение маркера в состоянии по умолчанию.

СТИЛИ ПУТЕЙ

Определяет стили и опции визуализации путей.

Имя переменной: polylinesStyles

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	**ID**. Уникальный ID пути. Значение данного поля должно соответствовать полю **ID** свойства **Tracks**.
name	String	**Name**. Имя предустановленного стиля.
description	String	**Description**. Описание предустановленного стиля.
color	Color	**Color**. Цвет ломаной линии.
width	Integer	**Width**. Ширина ломаной линии.
opacity	Double	**Opacity**. Прозрачность ломаной линии.
stroke	String	**Stroke**. Штрих ломаной линии.
dashArray	String	**Dash Length Array**. Рисунок тире и пробелов для рисования пути.
dashOffest	String	**Dash Offset**. Смещение массива штрихов.
additionalStyles	String	**Additional Styles**. Дополнительный стиль CSS, который будет применен к ломаной линии.

СТИЛИ КРУГОВОГО МАРКЕРА

Определяет стили и опции визуализации круговых маркеров.

Имя переменной: circleMarkersStyles

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	**ID**. Уникальный ID кругового маркера. Значение данного поля должно соответствовать полю **ID** свойства **Circle Markers**.
name	String	**Name**. Имя предустановленного стиля.
description	String	**Description**. Описание предустановленного стиля.
color	String	**Color**. Цвет маркера.

width	Integer	**Width**. Ширина маркера.
opacity	Double	**Opacity**. Прозрачность ломаной линии.
fill	Boolean	**Fill**. Если включено, круговой маркер будет заполнен цветом, указанным в полях **Fill Color** и **Fill Opacity**.
fillColor	String	**Fill Color**. Цвет заливки кругового маркера.
fillOpacity	Double	**Fill Opacity**. Прозрачность цвета заливки кругового маркера.
stroke	String	**Stroke**. Штрих маркера.
dashArray	String	**Dash Length Array**. Рисунок тире и пробелов для рисования контура кругового маркера.
dashOffest	String	**Dash Offset**. Смещение массива штрихов.
radius	Integer	**Radius**. Радиус кругового маркера.
additionalStyles	String	**Additional Styles**. Дополнительный стиль CSS, который будет применен к круговому маркеру.

СТИЛИ ОБЛАСТЕЙ

Определяет стили и опции визуализации областей.

Имя переменной: areasStyles

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	**ID**. Уникальный ID области. Значение данного поля должно соответствовать полю *id* свойства *areas*.
name	String	**Name**. Имя предустановленного стиля.
description	String	**Description**. Описание предустановленного стиля.
color	String	**Color**. Цвет области.
width	Integer	**Width**. Ширина ломаной линии.
opacity	Double	**Opacity**. Прозрачность ломаной линии.
fill	Boolean	**Fill**. Если включено, область будет заполнена цветом, указанным в полях **Fill Color** и **Fill Opacity**.
fillColor	String	**Fill Color**. Цвет заливки области.
fillOpacity	Double	**Fill Opacity**. Прозрачность цвета заливки области.
stroke	String	**Stroke**. Штрих границы области.
dashArray	String	**Dash Length Array**. Рисунок тире и пробелов для рисования контура области.
dashOffest	String	**Dash Offset**. Смещение массива штрихов.

additionalStyles	String	**Additional Styles**. Дополнительный стиль CSS, который будет применен к области.

СТИЛИ КРУГЛЫХ ОБЛАСТЕЙ

Определяет стили и опции визуализации круглый областей.

Имя переменной: circleAreasStyles

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	**ID**. Уникальный ID круглой области. Значение данного поля должно соответствовать полю *Id* свойства *Markers*.
name	String	**Name**. Имя предустановленного стиля.
description	String	**Description**. Описание предустановленного стиля.
color	String	**Color**. Цвет круглой области.
weight	Integer	**Width**. Ширина круглой области.
opacity	Double	**Opacity**. Прозрачность круглой области.
fill	Boolean	**Fill**. Если включено, круглая область будет заполнена цветом, указанным в полях **Fill Color** и **Fill Opacity**.
fillColor	String	**Fill Color**. Цвет заливки круглой области.
fillOpacity	Double	**Fill Opacity**. Прозрачность цвета заливки круглой области.
stroke	String	**Stroke**. Штрих границы круглой области.
dashArray	String	**Dash Length Array**. Рисунок тире и пробелов для рисования контура круглой области.
dashOffest	String	**Dash Offset**. Смещение массива штрихов.
additionalStyles	String	**Additional Styles**. Дополнительный стиль CSS, который будет применен к круглой области.

Общие события [? 239]

НАЖАТИЕ НА МАРКЕР

Событие возникает, когда пользователь нажимает на маркер.

Имя события: markerClick

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	ID события.

value	Data Table	Содержит вложенную таблицу с информацией о нажатом маркере. В таблице несколько полей: • **Latitude** • **Longitude** • **Name**

ПОИСК ПО КАРТЕ

Событие возникает, когда пользователь совершает поисковый запрос с использованием интегрированного поискового компонента.

Имя события: mapSearch

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	ID события.
value	Data Table	Содержит вложенную таблицу с информацией о результате поиска. В таблице несколько полей: • **Latitude** • **Longitude** • **Address**
level	Integer	Уровень 75 события по умолчанию.

НАЖАТИЕ НА ПОЛЬЗОВАТЕЛЬСКУЮ КНОПКУ

Событие возникает, когда пользователь нажимает на пользовательскую кнопку на компоненте карты.

Имя события: customButtonClick

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	ID события.
value	Data Table	Содержит вложенную таблицу с информацией о результате нажатия на кнопку. В таблице одно поле: • **Custom Button -** содержит имя события, например, *customButtonClicked.*
level	Integer	Уровень 75 события по умолчанию.

7.9.8 Отображение данных

Данный раздел описывает компоненты **Отображения данных**.

7.9.8.1 Системное дерево

Системное дерево используется для просмотра и администрирования различных серверных ресурсов и устройств.

Каждый узел дерева представляет контекст 41 сервера (и может включать подконтексты).

Чтобы увидеть разницу между деревом контекстов в Системном дереве и настоящим серверным деревом контекстов, перейдите сюда 45.

Панель инструментов

Компонент Системной дерево имеет отдельную панель инструментов, которая позволяет осуществлять следующие операции:

- **Поиск** контекстов по их описаниям
- Включать **Множественный выбор** для осуществления пакетных операций со множеством контекстов сразу (см. раздел Работа с множеством узлов 275 ниже).

Контекстное меню

При нажатии правой кнопкой мыши по узлу Системного дерева, открывается контекстное меню. Опции контекстного меню специфичны для узлов. Это действия 87, определенные в контексте AtomMind Server, соответствующем текущему узлу. Действие обычно взаимодействует с пользователем, выполняя различные UI Процедуры 88.

Пункт меню, соответствующий действию по умолчанию 88 (т.е. запуск действия происходит по двойному нажатию на узел), выделен **жирным** шрифтом.

Работа с множеством узлов

Для того, чтобы выбрать узлы из диапазона, нажмите на узел и, удерживая нажатой клавишу **Shift,** нажмите на другой узел. Будут выбраны оба узла, а также все узлы между ними. Чтобы выбрать несколько узлов, находящихся в разных диапазонах (т.е. если между двумя узлами есть невыбираемые узлы), удерживайте нажатой клавишу **Ctrl** и нажмите на узлы, которые хотите выбрать. При нажатии правой кнопкой мыши на выбранный узел, появится контекстное меню с операциями для всех выбранных узлов.

Более подробно о том, какие действия появляются в контекстном меню при выборе нескольких узлов, а также о том, как они работают, см. в главе Группировка действий 101.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

КОРНЕВОЙ ЭЛЕМЕНТ

Определяет корневой контекст контекстного дерева, отображаемого компонентом.

Имя переменной: root

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
root	String	Корневой элемент

ВЫБРАННЫЕ УЗЛЫ

Данное свойство определяет, какие узлы дерева будут выбираться по умолчанию.

Имя переменной: selectedNodes

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
key	String	**Key**. Уникальный ключ выбираемого узла дерева.
title	String	**Title**. Отображаемое имя узла.

РАЗВЁРНУТЫЕ УЗЛЫ

Данное свойство определяет, какие узлы дерева будут разворачиваться по умолчанию. Развернутый узел должен иметь хотя бы один дочерний узел.

Имя переменной: expandedNodes

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
key	String	**Key**. Уникальный ключ развернутого узла дерева.
title	String	**Title**. Отображаемое имя узла.

Общие события [? 239]

Общие события: отсутствуют

7.9.8.2 Пользовательское дерево

Данный компонент используется для построения структуры пользовательского дерева.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

МЕТКА

Текст заголовка дерева.

Имя переменной: treeLabel

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
treeLabel	String	Метка

ДАННЫЕ В ДЕРЕВЕ

Данное свойство определяет иерархию и визуальное представление узлов дерева.

Имя переменной: treeData

Записи: 0..не ограничено

Формат записи:

Имя поля	Тип поля	Примечания
key	String	**Key**. Уникальный ключ узла дерева.
title	String	**Title**. Отображаемое имя узла.
children	Data Table	**Children**. Определяет дочерние узлы для текущего узла. Вложенная таблица данных с двумя полями: • **Key -** уникальный ключ дочернего узла • **Title -** отображаемое имя дочернего узла.
nodeStyleId	String	**Node Style**. Уникальный ID стиля CSS для применения к узлу. Данное поле должно соответствовать полю *id* свойства *nodeStyle*.

ВЫБРАННЫЕ УЗЛЫ

Данное свойство определяет, какие узлы дерева будут выбираться по умолчанию.

Имя переменной: selectedNodes

Записи: 0..не ограничено

Формат записи:

Имя поля	Тип поля	Примечания
key	String	**Key**. Уникальный ключ выбираемого узла дерева.

РАЗВЁРНУТЫЕ УЗЛЫ

Данное свойство определяет, какие узлы дерева будут разворачиваться по умолчанию. Развернутый узел должен иметь хотя бы один дочерний узел.

Имя переменной: expandedNodes

Записи: 0..не ограничено

Формат записи:

Имя поля	Тип поля	Примечания
key	String	**Key**. Уникальный ключ развернутого узла дерева.

ОТМЕЧЕННЫЕ УЗЛЫ

Данное свойство определяет, какие узлы дерева будут отмечаться по умолчанию. Для использования этого свойства необходимо активировать свойство **Добавить флажок выбора**.

Обратите внимание, что отмеченные узла отличаются от выбранных узлов. Каждый узел может быть отмечен и/или выбран по отдельности.

Имя переменной: checkedNodes

Записи: 0..не ограничено

Формат записи:

Имя поля	Тип поля	Примечания
key	String	**Key**. Уникальный ключ отмечаемого узла.

ФИЛЬТРУЮЩИЙСЯ

Если включено, данное свойство позволяет фильтровать данные с использованием интегрированного поля поиска, расположенного в самом верху компонента дерева. Узлы дерева, соответствующие поисковому запросу, будут подсвечиваться.

Имя переменной: filterable

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
filterable	Boolean	Фильтрующийся

ЗАПОЛНИТЕЛЬ

Текст, отображаемый в поле поиска, если никакой другой текст не введен.

Имя переменной: inputPlaceholder

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
inputPlaceholder	String	Заполнитель

МНОЖЕСТВЕННЫЙ ВЫБОР

Если включено, разрешает выбирать несколько узлов.

Имя переменной: multiple

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
multiple	Boolean	Множественный выбор

СТИЛЬ ДЕРЕВА

Данное свойство определяет визуальное представление различных компонентов дерева.

Имя переменной: treeStyle

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания

folderImage	Data Block	**Folder Image**. Изображение для значка папки (узел с дочерними узлами).
fileImage	Data Block	**File Image**. Изображение для значка файла (узел без дочерних узлов).
switcherOpenImage	Data Block	**Open Switcher Image**. Изображение переключателя (открыт).
switcherCloseImage	Data Block	**Close Switcher Image**. Изображение переключателя (закрыт).

СТИЛЬ УЗЛА

Данное свойство определяет визуальное представление различных узлов дерева.

Имя переменной: nodeStyle

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	**ID**. Уникальный ID преднастроенного стиля узла.
name	String	**Name**. Имя преднастроенного стиля узла.
folderImage	Data Block	**Folder Image**. Изображение для значка папки (узел с дочерними узлами).
fileImage	Data Block	**File Image**. Изображение для значка файла (узел без дочерних узлов).
switcherOpenImage	Data Block	**Open Switcher Image**. Изображение переключателя (открыт).
swicherCloseImage	Data Block	**Close Switcher Image**. Изображение переключателя (закрыт).

ДОБАВИТЬ ФЛАЖОК ВЫБОРА

Если включено, добавляет флажки выбора к узлам дерева.

Имя переменной: checkable

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
checkable	Boolean	Добавить флажок выбора

ИЗОБРАЖЕНИЕ ДЛЯ КНОПКИ ФИЛЬТРАЦИИ

Изображение для кнопки фильтрации.

Имя переменной: filterButtonImage

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
filterButtonImage	Data Block	Изображение для кнопки фильтрации

СТИЛЬ ЗАГОЛОВКА

CSS стиль для применения к тексту заголовка.

Имя переменной: styleTreeHeaderLabel

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
styleTreeHeaderLabel	String	Стиль заголовка

СТИЛЬ КОНТЕЙНЕРА ЗАГОЛОВКА

CSS стиль для применения к контейнеру заголовка.

Имя переменной: styleTreeHeaderLabelContainer

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
styleTreeHeaderLabelContainer	String	Стиль контейнера заголовка

Общие события [? 239]

Общие события: отсутствуют

7.9.8.3 Метка

Метка - это область отображения для короткой текстовой строки, которая не реагирует на события ввода и поэтому не может получить фокус ввода с клавиатуры.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

ТЕКСТ

Текст сообщения метки.

Имя переменной: text

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
text	String	Текст

Общие события [? 239]

Общие события: Нажатие мыши 239

7.9.8.4 Изображение

Данный компонент отображает изображение.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

ИЗОБРАЖЕНИЕ

Изображение, отображаемое данным компонентом.

Имя переменной: image

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
image	Data Block	Изображение

АЛЬТЕРНАТИВНЫЙ ТЕКСТ

Описание, которое будет отображаться, если изображение не загрузилось корректно.

Имя переменной: alt

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
alt	String	Альтернативный текст

ВСПЛЫВАЮЩАЯ ПОДСКАЗКА

Текст подсказки, вплывающей при наведении курсора мыши.

Имя переменной: imageTitle

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
imageTitle	String	Всплывающая подсказка

Общие события [? 239]

Общие события: Нажатие мыши 239

7.9.8.5 Индикатор выполнения

Компонент, который, по умолчанию, отображает целое значение в пределах ограниченного интервала. Индикатор выполнения обычно демонстрирует прогресс какой-либо работы, показывая процент ее выполнения.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

ЗНАЧЕНИЕ

Значение, отображаемое на индикаторе выполнения (в процентах).

Имя переменной: value

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
value	Integer	Значение

Общие события [? 239]

Общие события: отсутствуют

7.9.8.6 Таблица данных

Редактор таблицы данных (РТД) является компонентом, необходимым для изменения или просмотра одной Таблицы данных 49. Он используется:

- Редактором свойств 256 для редактирования переменных 61 контекста 41
- Журналом событий 303 для просмотра Таблиц данных, относящихся к событиям 73
- UI процедурой Редактировать данные 90 для изменения параметров входа функции 70 и просмотра ее выхода, и т.д.
- А также многими другими средствами системы.

Большинство элементов Редактора таблицы данных, такие как заголовки столбцов и ячейки данных, обычно содержат всплывающие подсказки, которые появляются при наведении курсора мыши на элемент.

Всплывающая подсказка для заголовка столбца содержит поле **type** и строку **help**, заданные в формате 49 таблицы, если такой определен.

Всплывающая подсказка для ячейки содержит строковое представление значения ячейки, а также тип поля, используемого для данной ячейки. Это может пригодиться, например, если:

- значение ячейки слишком длинное и не помещается в ней,
- редактор ячейки имеет допустимые значения 49, и вы хотите увидеть само значение, а не его описание в раскрывающемся списке.

Отображения временных меток

Редактор таблицы данных использует настройки **Формат даты**, **Формат времени** и **Временная зона,** заданные в Свойствах учетной записи пользователя 481, чтобы правильно преобразовывать и отображать временные метки.

Все значения даты/времени отображаются во временной зоне активного пользователя. Если временная зона не указана в настройках пользователя, будет использоваться временная зона AtomMind Server.

Источник данных

Редактор таблицы данных может использовать два типа источников данных:

ТАБЛИЦА ДАННЫХ

Это опция по умолчанию, которая чаще всего используется и отображает отдельную таблицу данных. В этом режиме прокрутка недоступна. Фильтрация и сортировка применимы к записям только в просматриваемой/редактируемой таблице.

СПИСОК ЭКЗЕМПЛЯРОВ КЛАССА

В этом режиме редактируемая таблица представляет собой часть более длинного списка объектов (таких как экземпляры класса 885), доступного на стороне AtomMind Server.

Если список объектов используется в качестве источника, кнопки прокрутки и обновления доступны на панели инструментов. Если применяются правила фильтрации и сортировки, то фильтруются/сортируются серверные объекты, а не записи в отображаемой таблице.

Режимы

Редактор таблицы данных может работать с различных режимах, описанных в этом разделе.

РЕЖИМЫ РЕДАКТИРОВАНИЯ

Доступны два режима редактирования:

- **Обычный** режим. В этом режиме разрешено изменение значений.
- Режим **Только чтение**. В этом режиме редактирование данных не разрешено.

РЕЖИМЫ ПРЕДСТАВЛЕНИЯ ДАННЫХ

Существуют два режима представления данных:

- **Множество записей**. В этом режиме заголовок таблицы показывает описание для множества столбцов, а каждый ряд представляет собой одну запись таблицы данных.
- **Отдельная запись** (режим также называется **Вертикальным**). РТД использует этот режим, когда редактируемая таблица содержит всего одну запись, и добавление дополнительных записей невозможно. Этот режим имеет структуру из двух столбцов: левый столбец показывает имена полей, а правый содержит значения.

панель инструментов

Панель инструментов Редактора таблицы данных обеспечивает доступ к операциям, которые могут выполняться с таблицей данных 49.

	**Показать/Скрыть дополнительные свойства**	Эта триггерная кнопка используется, чтобы включить показ или скрытие полей или столбцов, помеченных как дополнительные.
	**Обновить**	Кнопка доступна только в режиме **Список серверных объектов**. Кнопка "Обновить" перезагружает весь список и обновляет текущую страницу.
	**Добавить ряд**	Добавляет новый ряд перед выбранным на данный момент. Если ряд не выбран, новый добавляется в конец таблицы. Данная кнопка отключена или скрыта, если добавление рядов к таблице невозможно. Если компонент работает в режиме **Список серверных объектов**, эта кнопка позволяет вставить новый объект в список на стороне сервера, а не добавлять запись прямо с отображаемую таблицу. Сначала отобразится еще один Редактор таблица данных, чтобы дать заполнить поля вновь создаваемого объекта. Новый объект станет видимым, только если он отвечает текущим правилам фильтрации и

		подходит к текущей странице. В большинстве случаев, чтобы увидеть новый объект, потребуется нажать на кнопку "Обновить".
	**Удалить выбранные ряды**	Удаляет выбранный ряд (ряды). Данная кнопка отключена или скрыта, если удаление рядов из таблицы невозможно. Если компонент работает в режиме **Список серверных объектов**, эта кнопка удаляет объект(ы) из списка на стороне сервера помимо удаления соответствующих записей из отображаемой таблицы.
	**В начало**	Кнопка доступна только режиме **Список серверных объектов**. Делает прокрутку списка до первой страницы.
	**Предыдущая страница**	Кнопка доступна только режиме **Список серверных объектов**. Делает прокрутку списка до предыдущей страницы.
	**Следующая страница**	Кнопка доступна только режиме **Список серверных объектов**. Делает прокрутку списка до следующей страницы.
	**В конец**	Кнопка доступна только режиме **Список серверных объектов**. Делает прокрутку списка до последней страницы.
	Row Count	Кнопка доступна только режиме **Список серверных объектов**. Определяет количество объектов для отображения на одной странице. Перечисленные выше кнопки прокрутки могут использоваться для навигации по длинному списку.
	**Поднять ряд**	Поднимает текущий ряд вверх. Данная кнопка скрыта, если порядок рядов не может быть изменен. Данная операция меняет актуальные данные в таблице (в противоположность операции упорядочивания, которая просто меняет представление данных).
	**Опустить ряд**	Опускает текущий ряд. Данная кнопка скрыта, если порядок рядов не может быть изменен. Данная операция меняет актуальные данные в таблице (в противоположность операции упорядочивания, которая просто меняет представление данных).
	**Отменить изменения**	Отменяет изменения в таблице. Данная кнопка включается только после того, как в таблицу были внесены изменения.
	**Включить/Отключить горизонтальную прокрутку**	Если горизонтальная прокрутка отключена, ширина всех столбцов подстраивается автоматически под ширину окна Редактора таблицы данных. Если прокрутка включена, в Редактор таблицы данных имеется горизонтальная полоса прокрутки, и ширина все столбцов подстраивается под их содержимое. По умолчанию данная опция отключена.
	**Импорт таблицы данных**	Импортирует 300 данные из файла.
	**Экспорт таблицы данных**	Экспортирует 300 данные в файл.
	**Создать отчет**	Формирует отчет 300 на основе редактируемой или просматриваемой таблицы.
	**Помощь**	Открывает раздел документации, относящийся к редактируемым/просматриваемым данным.

Обработка привязок данных

Редактор таблицы данных оперирует привязками 741, содержащимися в определении формата 49 редактируемой Таблицы данных. Во время загрузки редактора, он читает список привязок для таблицы и обрабатывает их в фоновом режиме. Привязки оцениваются и используются для изменения ячеек таблицы. Выражения привязок, используемые редактором, могут содержать относительные ссылки на ячейки изменяемой таблицы вместе с ссылками на данные из различных контекстов 41. Чтобы узнать, как создать такие ссылки, прочтите информацию о привязках данных здесь 49.

Общие переменные (Свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

ОБЩЕ СВОЙСТВА

Компонент Таблица данных служит основой для некоторых других компонентов.

Переменные (свойства), общие для наследуемых компонентов, описаны в разделе Общие свойства 285.

ТАБЛИЦА ДАННЫХ

Имя переменной: dataTable

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
dataTable	Data Table	Таблица данных, отображаемая в компоненте.

ОБщие события [? 239]

ОБЩИЕ СОБЫТИЯ

Компонент Таблица данных служит основой для некоторых других компонентов.

События, общие для наследуемых компонентов, описаны в разделе Общие события 294.

7.9.8.6.1 Общие свойства

В этом разделе описываются свойства компонента Таблица данных, общие для всех унаследованных компонентов.

ПОКАЗАТЬ СТРОКУ ПОИСКА

Если включено, позволяет осуществлять поиск по журналу событий.

Имя переменной: enableSearch

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
enableSearch	Boolean	Показать строку поиска

ЗАПОЛНИТЕЛЬ В ПОИСКЕ

Сообщение, которое отображается в строке поиска, если не введен другой текст.

Имя переменной: searchPlaceholder

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
searchPlaceholder	String	Заполнитель в поиске

ТОЛЬКО ЧТЕНИЕ

Если включено, запрещает любое редактирование пользователем в журнале событий.

Имя переменной: readOnly

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
readOnly	Boolean	Только чтение

СОБЫТИЕ ВМЕСТО ДОБАВЛЕНИЯ ЗАПИСИ

Если включено, разрешает добавлять записи к журнал событий, используя соответствующее событие.

Имя переменной: addRecordUsingEvent

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
addRecordUsingEvent	Boolean	Событие вместо добавления записи

СОБЫТИЕ ВМЕСТО УДАЛЕНИЯ ЗАПИСИ

Если включено, разрешает удалять записи из журнала событий, используя соответствующее событие.

Имя переменной: deleteRecordUsingEvent

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
deleteRecordUsingEvent	String	Событие вместо удаления записи

СОБЫТИЕ ВМЕСТО ИЗМЕНЕНИЯ ЯЧЕЙКИ

Если включено, разрешает обновлять ячейки журнала событий, используя соответствующее событие.

Имя переменной: updateCellUsingEvent

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
updateCellUsingEvent	Boolean	Событие вместо изменения ячейки

СОРТИРОВКА НЕСКОЛЬКИХ СТОЛБЦОВ

Если включено, разрешает использовать сортировку по более чем одному столбцу журнала событий.

Имя переменной: multipleSorting

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
multipleSorting	Boolean	Сортировка нескольких столбцов

ПОКАЗАТЬ ПАНЕЛЬ ИНСТРУМЕНТОВ

Этот флаг определяет отображение панели инструментов журнала событий.

Имя переменной: showToolbar

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
showToolbar	Boolean	Показать панель инструментов

СТРОК НА СТРАНИЦЕ

Определяет количество строк журнала событий по умолчанию, отображаемых на одной странице.

Имя переменной: rowsPerPage

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
rowsPerPage	Integer	Строк на странице

ЗНАЧЕНИЯ ФИЛЬТРА СТРАНИЦ

Данное свойство позволяет указать пользовательские номера страниц, которые будут доступны в пагинаторе.

Имя переменной: customPaginatorValues

Записи: 0..не ограничено

Формат записи:

Имя поля	Тип поля	Примечания
value	Integer	Значения фильтра страниц

ПОКАЗАТЬ СЕЛЕКТОР «СТРОК НА СТРАНИЦЕ»

Этот флаг определяет, будет ли отображаться селектор "Строк на странице".

Имя переменной: showRowsPerPageSelector

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания

showRowsPerPageSelector	Boolean	Показать селектор «Строк на странице»

ТИП СЕЛЕКТОРА «СТРОК НА СТРАНИЦЕ»

Определяет тип селектора "Строк на странице". Данное свойство имеет два возможных значения:

- **Combo Box -** селектор будет отображаться в виде поля с выпадающим списком возможных значений
- **Button Group -** селектор будет отображаться в виде набора кнопок

Имя переменной: rowsPerPageSelectorType

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
rowsPerPageSelectorType	String	Тип селектора «Строк на странице»

ФИКСИРОВАННЫЙ ФОРМАТ

Если включено, применяет к Журналу событий пользовательский формат, указанный в свойстве **Формат**.

Имя переменной: fixedFormat

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
fixedFormat	Boolean	Фиксированный формат

ФОРМАТ

Пользовательский формат журнала событий. Полное и подробное описание формата таблицы данных см. в соответствующем разделе 50.

Имя переменной: format

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
format	Data Table	Формат

ВЫБРАННЫЕ СТРОКИ

Данное свойство позволяет определить, какие строки будут выбраны по умолчанию.

Имя переменной: selectedRows

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания

key	String	Уникальный ключ выбранной строки таблицы данных.

НАСТРОЙКИ СТОЛБЦОВ

Данное свойство включает различные настройки визуального представления и поведения столбцов Журнала событий.

Имя переменной: columnsSettings

Записи: 0..не ограничено

Формат записи:

Имя поля	Тип поля	Примечания
name	String	**Name**. Имя столбца.
fixed	String	**Fixed Column**.
headerTextBehavior	String	**Header Text Behavior.**
wordBreak	String	**Word Break**. Определяет поведение переноса строки, если текст выходит за пределы ячейки столбца.
textAlignHorizontal	String	**Body Text Align Horizontal**. Определяет горизонтальное положение текста в ячейке столбца.
textAlignHeaderHorizontal	String	**Header Text Align Horizontal**. Определяет горизонтальное положение текста в заголовке столбца.
textAlignVertical	String	**Body Text Align Vertical**. Определяет вертикальное положение текста в ячейке столбца.
textAlignHeaderVertical	String	**Header Text Align Vertical**. Определяет вертикальное положение текста в заголовке столбца.
sorter	Boolean	**Sorter**. Активирует сортировку по столбцу.
search	Boolean	**Search**. Активирует поиск по данным столбца.
filterTable	Data Table	**Filter Table**. Позволяет создавать фильтры для определенного столбца. Вложенная таблица данных с двумя полями: • **Value -** значение ячейки для фильтрации. Обратите внимание, что если столбец имеет тип полей *Boolean*, разреены только значения *true* или *false* • **Text -** описание фильтра
width	Integer	**Width**. Ширина столбца, в пикселях.

ВЫСОТА ДО ПОЯВЛЕНИЯ СКРОЛЛА

Максимальная высота в пикселях, после которой включается вертикальный скролл в ячейке журнала событий.

Имя переменной: heightScrollEnabled

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
heightScrollEnabled	Integer	Максимальная высота без вертикальногро скролла

ВКЛЮЧИТЬ ВЫБОР СТРОКИ

Если включено, позволяет выбирать строки с помощью флажков.

Имя переменной: checkRows

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
checkRows	Boolean	Включить выбор строки

ПОЛЕ ДЛЯ ОПРЕДЕЛЕНИЯ БЛОКИРОВКИ СТРОКИ

Имя переменной: columnToDefineRowLock

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
columnToDefineRowLock	String	

ПОЛЕ ДЛЯ ОПРЕДЕЛЕНИЕ КОЛОНКИ ДЛЯ БЛОКИРОВКИ ДОПОЛНИТЕЛЬНЫХ КНОПОК

Имя переменной: columnToDefineKebabLock

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
columnToDefineKebabLock	String	

НАЗВАНИЕ КОЛОНКИ ВЛОЖЕННОЙ ТАБЛИЦЫ

Имя переменной: columnToDefineNameForNestedTable

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
columnToDefineNameForNestedTable	String	

РАЗРЕШИТЬ ИЗМЕНЕНИЕ РАЗМЕРА СТОЛБЦА

Если включено, позволяет менять размер столбца.

Имя переменной: resizableColumn

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
resizableColumn	Boolean	Разрешить изменение размера столбца

ПОДТВЕРЖДЕНИЕ УДАЛЕНИЯ СТРОК

Если включено, действие панели инструментов Удалить ряд потребует подтверждения.

Имя переменной: confirmActionDelete

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
confirmActionDelete	Boolean	

ПОДТВЕРЖДЕНИЕ ДОБАВЛЕНИЯ СТРОК

Если включено, действие панели инструментов Добавить ряд потребует подтверждения.

Имя переменной: confirmActionAdd

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
confirmActionAdd	Boolean	

ФИКСИРОВАННОЕ ПОЛОЖЕНИЕ ЛЕВОГО СТОЛБЦА

Имя переменной: fixedLeftColumn

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
fixedLeftColumn	Boolean	

ИКОНКА ФИЛЬТРАЦИИ

Изображение для иконки фильтрации.

Имя переменной: filterIcon

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
filterIcon	Data Block	Иконка фильтрации

ИКОНКА ПОИСКА

Изображение для иконки поиска.

Имя переменной: searchIcon

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
searchIcon	Data Block	Иконка поиска

ВКЛЮЧИТЬ

Если включено, добавляет к журналу событий дополнительный столбец с настраиваемыми кнопками.

Имя переменной: additionalActionEnable

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
additionalActionEnable	Boolean	

ПОКАЗАТЬ В РЯД

Имя переменной: additionalActionShowInRow

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
additionalActionShowInRow	Boolean	

ИКОНКА

Имя переменной: additionalActionIcon

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
additionalActionIcon	Data Block	Этот флаг показывает, будет ли компонент "слушать" события Нажатие мыши.

ШИРИНА ДОПОЛНИТЕЛЬНОЙ ИКОНКИ

Имя переменной: additionalActionWidth

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
additionalActionWidth	Integer	Ширина в пикселях, которая будет добавлена к контейнеру иконки.

НАСТРОЙКИ

Данное свойство позволяет задать настройки элемента Дополнительное действие.

Имя переменной: additionalActionSettings

Записи: 0..не ограничено

Формат записи:

Имя поля	Тип поля	Примечания
uid	String	Уникальный ID кнопки дополнительного действия.
desc	String	Описание кнопки.
icon	Data Block	Изображение для кнопки в обычном состоянии.
iconHover	Data Block	Изображение для кнопки в состоянии Наведение курсора мыши.
iconActive	Data Block	Изображение для кнопки в состоянии Активна.
buttonAction	String	Определяет, какое действие будет выполняться для соответствующего ряда путем нажатия на кнопку. Свойство имеет несколько допустимых значений: • **Edit** • **Delete** • **Cancel** • **Apply**
confirm	Boolean	Если включено, действие по кнопке потребует подтверждения.
confirmTitle	String	Текст сообщения подтверждения.

СТИЛЬ

Имя переменной: additionalActionStyle

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
additionalActionStyle	String	Стиль CSS, который будет применен к элементу *Дополнительное действие*.

СТИЛЬ ПОДСКАЗКИ

Имя переменной: tooltipStyle

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
tooltipStyle	String	Стиль CSS, который будет применен к подсказке.

7.9.8.6.2 Общие события

В этом разделе описываются события компонента Таблица данных, общие для всех унаследованных компонентов.

ВЫБОР СТРОКИ

Событие возникает, когда пользователь выбирает определенную строку при помощи флажка. Для этого события требуется активация свойства **Включить выбор ряда**.

Имя события: rowSelection

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	ID события.
value	String	ID выбранного ряда.
level	Integer	Уровень 75 события по умолчанию.

НАЖАТИЕ НА ЯЧЕЙКУ

Событие возникает, когда пользователь нажимает мышью на определенную ячейку.

Имя события: cellClicked

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	ID события.
value	Data Table	Данное свойство содержит вложенную таблицу данных с несколькими полями: • **Record Index -** индекс записи • **Field Index -** индекс поля, содержащего определенную ячейку • **Record -** индекс ячейки в определенном поле
level	Integer	Уровень 75 события по умолчанию.

ЯЧЕЙКА ОБНОВЛЕНА

Событие возникает, когда пользователь обновляет значение определенной ячейки в таблице данных.

Имя события: cellUpdated

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	ID события.
value	String	Данное свойство содержит вложенную таблицу данных с несколькими полями: • **Record Index -** индекс записи • **Field -** индекс поля, содержащего определенную ячейку • **Value -** новое значение, введенное в ячейку.
level	Integer	Уровень 75 события по умолчанию.

ЗАПИСЬ ДОБАВЛЕНА

Событие возникает, когда пользователь добавляет ряд в журнал событий.

Имя события: rowAdded

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	ID события.
level	Integer	Уровень 75 события по умолчанию.

ЗАПИСЬ УДАЛЕНА

Событие возникает, когда пользователь удаляет запись из журнала событий.

Имя события: rowRemoved

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	ID события.
value	Integer	Индекс удаленной записи.
level	Integer	Уровень 75 события по умолчанию.

ИЗМЕНЕНИЕ ПРАВИЛ СОРТИРОВКИ ИЛИ ФИЛЬТРАЦИИ

Событие возникает, когда пользователь применяет сортировку или фильтрацию к событиям в журнале.

Имя события: sortingOrFilteringUpdated

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	ID события.
value	String	Данное свойство содержит вложенную таблицу данных с двумя полями: • **Sorting Data -** содержит вложенную таблицу данных с именами полей и применяемыми к ним правилами сортировки • **Filtering Data -** содержит вложенную таблицу данных с именами полей и применяемыми к ним правилами фильтрации
level	Integer	Уровень 75 события по умолчанию.

7.9.8.6.3 Изменение данных

Если Редактор таблиц работает в режиме редактирования, значения в таблице могут быть изменены.

Прорисовщик и редактор, используемые для отображения и изменения значения в каждой ячейке, зависят от нескольких факторов.

- Тип поля
- Редактор/Прорисовщик, заданный в определении поля 49 (т.е. является частью Формата таблицы)
- Доступность Значений выборки 49, определенных для данного поля в Формате таблицы.
- Является ли значение поля NULL.

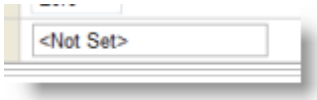
Специальные Прорисовщики/Редакторы

Существует несколько особых случаев, когда используются нестандартные редакторы:

ПРОРИСОВЩИК ПУСТОГО ЗНАЧЕНИЯ

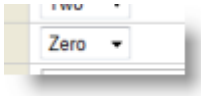
Если поле определено как *Пустое* и его значение - *NULL,* оно отображается как `<Not Set>` в редакторе таблиц. Нажмите на ячейку, чтобы начать изменение и установите для нее *непустое значение по умолчанию*.

Вы можете вернуть значение NULL при помощи операции **Удалить значение** в контекстном меню ячейки.



РЕДАКТОР ЗНАЧЕНИЙ ВЫБОРКИ

Если формат поля содержит значения выборки 49, изменение выполняется при помощи раскрывающегося списка:



Если в формате поля установлен флажок Расширенные 49 значения выборки 49, можно ввести любое пользовательское значение в текстовое поле рядом с комбинированным списком, если в нем выбрана опция **Другое**:

Стандартные Прорисовщики/Редакторы

ПОЛЯ ТИПА STRING/INTEGER/LONG/FLOAT

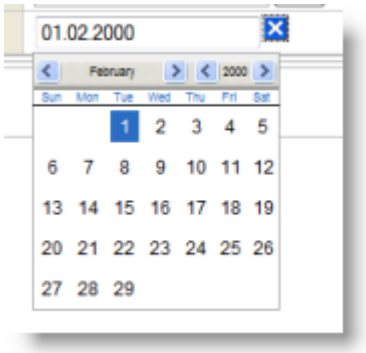
Значения *Integer, Long, String* и *Float* изменяются в обычном текстовом поле:

ПОЛЯ ТИПА BOOLEAN

Boolean значения изменяются при помощи контрольных кнопок и представлены обозначениями **Да** (для TRUE) или **Нет** (для FALSE) в режиме "Только чтение".

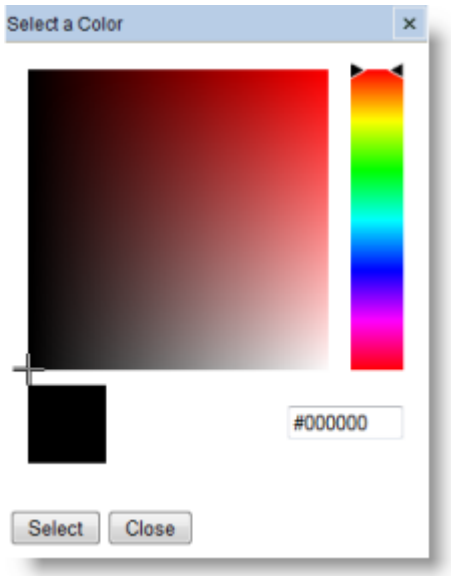
ПОЛЯ ДАТЫ

Значения *даты* и *даты/времени* изменяются при помощи окна выбора дат:



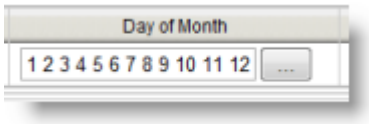
ПОЛЯ ЦВЕТА

Цвета выбираются при помощи Пикера цвета:



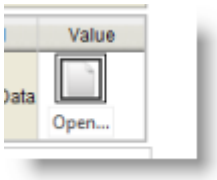
ПОЛЯ ТАБЛИЦЫ ДАННЫХ

Поля таблицы данных (т.е. поля, значения которых представляют собой таблицы данных) изменяются встроенным Редактором таблиц, который открывается в отдельной странице. Страница открывается при нажатии кнопки **[...]** в ячейке, содержащей внедренную Таблицу данных:



ПОЛЯ БЛОКОВ ДАННЫХ

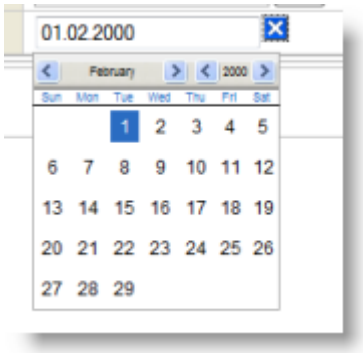
Редактор блоков данных по умолчанию позволяет выбрать и сохранить файл любого типа из/в систему локальных файлов AtomMind Clienta:



Дополнительные Прорисовщики/Редакторы

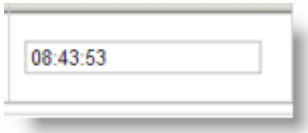
РЕДАКТОР ДАТЫ

Редактор даты позволяет задать дату при помощи выбора даты, однако, установка времени невозможна:



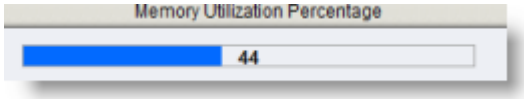
РЕДАКТОР ВРЕМЕНИ

Редактор времени позволяет задать время в форме строки, однако, дату выбрать невозможно:



ПРОРИСОВЩИК СТОЛБЦОВ

Прорисовщик столбцов отображает значения типа *Integer*, *Long*, *Float* и *String* в виде процентов от максимума, определяемого Опциями редактора:



Строковые значения конвертируются в числа максимально продуктивным способом.

РЕДАКТОР ПЕРИОДА

Редактор периода позволяет назначить временной период в виде определенного числа временных единиц:

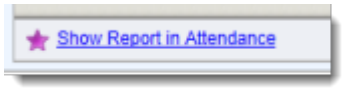
РЕДАКТОР ПАРОЛЯ

Редактор пароля очень похож на обычную текстовую область, однако, вводимые символы не отображаются:

Копировать значения из ячеек, которые используют данный редактор невозможно.

ССЫЛКА

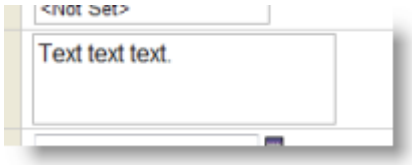
Ссылка отображает активные ссылки синим подчеркнутым шрифтом:



Нажатие по ссылке запускает действие сервера.

ТЕКСТОВАЯ ОБЛАСТЬ

Длинные *строковые* значения могут просматриваться и редактироваться в редакторе Текстовой области:



РЕДАКТОРЫ КОНТЕКСТОВ И КОНТЕКСТНЫХ МАСОК

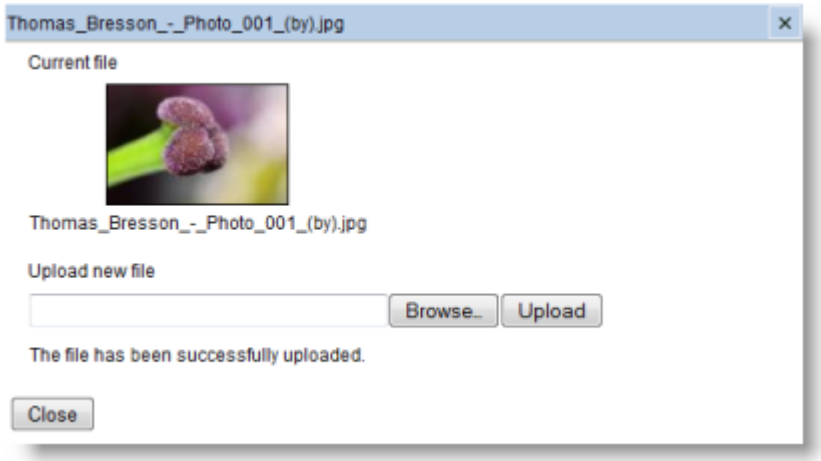
Строковые значения, относящиеся к путям контекста[41] и контекстным маскам[44], определяются при помощи компонента **Селектор маски контекста**. Он выглядит как обычное текстовое поле с кнопкой справа:

Нажатие кнопки **[...]** открывает компонент Селектор объектов[257], чтобы по указанию и щелчку можно было построить маску.

РЕДАКТОРЫ ЗВУКА/ИЗОБРАЖЕНИЯ/ФАЙЛА

Поля *блоков данных* могут содержать изображения, звуки или типичные файлы

Файлы, звуки и изображения внедрены в ячейки при помощи Селектора файлов:



Изображения в таблице данных показываются в виде эскиза (см. предыдущий скриншот). Нажатие на эскиз открывает изображение в полном размере.

Звуки проигрываются при нажатии на кнопку **Воспроизвести**.

7.9.8.6.4 Контекстное меню

Контекстное меню в Редакторе таблицы данных включает в себя несколько операций:

Удалить значение	Устанавливает значение ячейки на NULL (<Не задано>). Данная операция применима в том случае, если текущее поле является пустым, т.е. поддерживает значения NULL.
Отменить изменения значений в ячейках	Возвращает значение ячейки, которое было во время открытия Редактора таблицы данных. Если в это время ячейки не существовало (т.е. ячейка принадлежит только что созданному ряду), данная операция установит значение по умолчанию (см. далее).
Установить ячейку по умолчанию	Устанавливает значение ячейки на значение по умолчанию для данного столбца (обычно заданное на сервере в качестве части описания формата таблицы).
Внести значение ячейки в столбец	Копирует значение из выбранной ячейки во все другие ячейки текущего столбца.
Сброс изменений в таблице	Отменяет все изменения в таблице, внесенные после ее открытия.
Просмотр формата таблицы	Показывает формат 49 текущей таблицы.

Когда редактор таблицы данных используется для изменения значения переменной 61 контекста (в противоположность сырым данным от другого источника), контекстное меню в редакторе содержит дополнительные действия, относящиеся к данной переменной 102.

7.9.8.6.5 Упорядочивание и фильтрация

Упорядочивание и фильтрация редактора таблицы данных в AtomMind Web Desktop похожи на те же функции редактора таблицы данных AtomMind Client 392.

7.9.8.6.6 Экспорт/импорт данных

Экспорт и импорт данных редактором таблицы данных в AtomMind Web Desktop происходят по тому же принципу, что и те же функции в редакторе таблицы данных AtomMind Client 394.

7.9.8.6.7 Создание отчета

Создание отчета редактором таблицы данных в AtomMind Web Desktop схоже с теми же функциями Редактора таблицы данных 397 в AtomMind Client.

7.9.8.7 Классовая таблица данных

Данный компонент используется для редактирования и просмотра списка экземпляров класса 885.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

НАСЛЕДУЕМЫЕ ПЕРЕМЕННЫЕ (СВОЙСТВА)

Компонент Классовая таблица данных имеет в основе компонент Таблица данных 282, поэтому наследует общие с ним свойства 285.

СПИСОК ЭКЗЕМПЛЯРОВ КЛАССА

Данное свойство определяет параметры списка экземпляров класса, которыми управляет данный компонент.

Имя переменной: classInstanceListParameters

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
storageContext	String	**Storage Context**. Путь к контексту хранилища экземпляров класса.
storageView	String	**View**. Просмотр 888 по умолчанию выбранного экземпляра класса.
storageQuery	String	**Custom Query**. Пользовательский запрос 829 на подготовку исходных данных.
storageTable	String	**Table**. Пользовательский формат таблицы 2123 для выбранного экземпляра класса.
storageFilter	Data Table	**Filter**. Определяет правила фильтрации для выбранного экземпляра класса. Вложенная таблица дынных с несколькими полями: • **Logical Operation** • **Type** • **Column** • **Operation** • **Value** • **Nested Conditions**
storageSorting	Data Table	**Sorting**. Определяет правила сортировки для выбранного экземпляра класса. Вложенная таблица дынных с двумя поля: • **Column or Expression** • **Sort Order**
relationField	String	**Relation**. Определяет поле, выбранное для связи с другими экземплярами класса.

Общие события [? 239]

НАСЛЕДУЕМЫЕ СОБЫТИЯ

Компонент Классовая таблица данных имеет в основе компонент Таблица данных 282, поэтому наследует общие с ним события 294.

7.9.8.8 Таймер

Данный компонент отображает таймер.

0:07

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

ЗАПУЩЕН

Этот флаг показывает, активен ли таймер.

Имя переменной: running

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
running	Boolean	Запущен

ВРЕМЯ НАЧАЛА

Время, когда таймер начинает отсчет, в секундах.

Имя переменной: startTime

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
startTime	Integer	Время начала

ТОЧКА ОСТАНОВА

Время, когда таймер заканчивает отсчет, в секундах.

Имя переменной: breakpoint

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
breakpoint	Integer	Точка останова

ТИП ТАЙМЕРА

Тип таймера. Свойство имеет два возможных значения:

- **Timer -** таймер прямого отсчета
- **Stopwatch -** таймер обратного отсчета

Имя переменной: componentType

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
componentType	String	Тип таймера

ПЕРИОД СИНХРОНИЗАЦИИ

Частота отправки сообщений о статусе таймера в консоль веб-браузера, в секундах.

Имя переменной: syncTime

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
syncTime	Integer	Период синхронизации

Общие события [? 239]

ДОСТИГНУТА ТОЧКА ОСТАНОВА

Событие возникает, когда таймер достигает точки останова.

Имя события: breakpointReached

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	ID события. Данное поле скрыто.
value	String	Имя компонента, например, *timer0*.

7.9.8.9 Журнал событий

Компонент *Журнал событий* предназначен для просмотра и управления событиями 73 сервера в реальном времени. Он также предоставляет доступ к истории события 75 и позволяет подтверждать 76 события.

Журнал событий состоит из двух главных частей: Текущие события 308 и История событий 309.

Обе части имеют инструментальную панель, которая обеспечивает быстрый доступ к наиболее часто используемым функциям. В левой части обеих инструментальных панелей расположен раскрывающийся список, который позволяет выбрать фильтр событий 762. Изначально, фильтром событий по умолчанию является `<Empty Filter>`, а значит, события не отображаются.

В некоторых случаях компонент Журнал событий отображает ранее определенный набор типов событий. В таком случае, список Фильтров событий и соответствующие кнопки ("Отключить фильтр", "Настроить фильтр") не доступны.

События можно сортировать. Чтобы включить эту функцию, нажмите на заголовок столбца. Последующие нажатия будут менять порядок сортировки для данного столбца с возрастающего на убывающий и обратно.

Нажатие кнопки в столбце Данные около любого события открывает Таблицу данных 49, связанную с ним в Редакторе таблицы данных 282.

Журнал событий использует настройки **Шаблон даты**, **Шаблон времени** и **Временная зона**, определенные в Свойствах учетной записи пользователя 481, для правильного конвертирования и отображения различных временных меток.

Все значения даты/времени отображаются во временной зоне активного пользователя.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

НАСЛЕДУЕМЫЕ ПЕРЕМЕННЫЕ (СВОЙСТВА)

Компонент Журнал событий имеет в основе компонент Таблица данных 282, поэтому наследует общие с ним свойства 285.

РЕЖИМ

Компонент Журнал событий может работать в двух основных режимах:

- **Список событий**. В этом режиме отображаются только события, указанные свойством **События**. Изначально видимые столбцы определяются свойствами компонента Журнал событий (см. ниже). Никакие дополнительные поля не отображаются, а также не выполняется фильтрация/подсвечивание.
- **Фильтр событий**. В этом режиме Журнал событий активирует серверный фильтр событий 762, указанные настройкой **Фильтр событий**, и использует этот фильтр для отбора и подсвечивания событий. Модель столбцов также строится согласно настройкам фильтра.

Имя переменной: mode

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
searchIcon	Integer	Режим

СОБЫТИЯ

Данное свойство определяет, какой тип событий должен отображаться в **Режиме** *Список событий*.

Имя переменной: eventList

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
mask	String	**Context Mask**. Контекст или маска контекста для отбора событий. Отображаются только события, возникшие в контекстах согласно этой маске 44.
event	String	**Event**. Имя события для отображения в журнале событий.

ФИЛЬТР СОБЫТИЙ

Путь контекста фильтра событий 762, который будет использоваться в **Режиме** *Фильтр событий*.

Если данная настройка NULL, компонент Журнал событий позволит выбирать фильтр из выпадающего списка.

Обратите внимание, что можно выбрать разные фильтры для разделов Текущие события и История событий.

Имя переменной: filter

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
filter	String	Путь контекста фильтра событий, например, *users.admin.filters.alerts.*

РАЗДЕЛЫ

Определяет, какие секции будут отображаться компонентом:

- Оба раздела

- Текущие события
- История событий

Имя переменной: sections

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
sections	Integer	Секции

АКТИВНЫЙ РАЗДЕЛ

Определяет, какая секция будет активной по умолчанию, когда выбрана опция *Оба раздела* в свойстве *Разделы*.

Имя переменной: activeSection

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
activeSection	Integer	Активный раздел

АВТОМАТИЧЕСКИ ЗАГРУЖАТЬ ИСТОРИЮ СОБЫТИЙ

Если включено, автоматически загружает историю событий при загрузке компонента.

Имя переменной: preloadHistory

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
searchIcon	Boolean	Автоматически загружать историю событий

ПОКАЗЫВАТЬ ИМЕНА КОНТЕКСТОВ

Флаг, определяющий видимость столбца Контекст.

Имя переменной: showContexts

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
showContexts	Boolean	Показывать имена контекстов

ПОКАЗЫВАТЬ ИМЕНА СОБЫТИЙ

Флаг определяет видимость столбца Событие.

Имя переменной: showNames

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
showNames	Boolean	Показывать имена событий

ПОКАЗЫВАТЬ УРОВНИ СОБЫТИЙ

Флаг определяет видимость столбца Уровень.

Имя переменной: showLevels

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
showLevels	Boolean	Показывать уровни событий

ПОКАЗЫВАТЬ ПОДТВЕРЖДЕНИЯ СОБЫТИЙ

Флаг определяет видимость столбца Подтверждения.

Имя переменной: showAcknowledgements

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
showAcknowledgements	Boolean	Показывать подтверждения событий

ПОКАЗЫВАТЬ ОБОГАЩЕНИЯ СОБЫТИЙ

Флаг определяет видимость столбца Обогащения.

Имя переменной: showEnrichments

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
showEnrichments	Boolean	Показывать обогащения событий

ПОКАЗЫВАТЬ ДАННЫЕ СОБЫТИЙ

Флаг определяет видимость столбца Данные.

Имя переменной: showData

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
showData	Boolean	Показывать данные событий

ТЕКУЩИЕ СОБЫТИЯ

Данное свойство позволяет указать список экземпляров пользовательского события для отображения с разделе Текущие события журнала событий, когда включен **Режим** *Журнал пользовательских событий*.

Имя переменной: currentSection

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
creationtime	Data	Временная метка создания события.
name	String	Имя события.
context	String	Имя соответствующего контекста.
level	Integer	Уровень 75 события.
data	Data Table	Данные события. Можно вставить таблицу данных любого формата.
acknowledgements	Data Table	Подтверждение события. Эта вложенная таблица данных содержит несколько полей: • **Author -** человек, подтверждающий событие • **Time -** временная метка подтверждения события • **Text -** текстовое сообщение подтверждения
enrichments	Data Table	Обогащение события. Эта вложенная таблица данных содержит несколько полей: • **Name -** имя обогащения • **Value -** текстовое сообщение обогащения • **Date -** временная метка обогащения события • **Author -** человек, обогащающий событие

ИСТОРИЯ СОБЫТИЙ

Данное свойство позволяет указать список экземпляров пользовательского события для отображения с разделе История событий журнала событий, когда включен **Режим** *Журнал пользовательских событий*.

Имя переменной: historySection

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
creationtime	Data	Временная метка создания события.
name	String	Имя события.

context	String	Имя соответствующего контекста.
level	Integer	Уровень 75 события.
data	Data Table	Данные события. Можно вставить таблицу данных любого формата.
acknowledgements	Data Table	Подтверждение события. Эта вложенная таблица данных содержит несколько полей: • **Author -** человек, подтверждающий событие • **Time -** временная метка подтверждения события • **Text -** текстовое сообщение подтверждения
enrichments	Data Table	Обогащение события. Эта вложенная таблица данных содержит несколько полей: • **Name -** имя обогащения • **Value -** текстовое сообщение обогащения • **Date -** временная метка обогащения события • **Author -** человек, обогащающий событие

Общие события [? 239]

НАСЛЕДУЕМЫЕ СОБЫТИЯ

Компонент Журнал событий имеет в основе компонент Таблица данных 282, поэтому наследует общие с ним события 294.

СОБЫТИЕ ДОБАВЛЕНО В ЖУРНАЛ

Событие журнала событий возникает при добавлении нового события в журнал.

Имя события: eventAddedToLog

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
eventData	Data Table	**Event Data**. Данные нового добавленного события.

7.9.8.9.1 Текущие события

Таблица Текущие события отображает события, происходящие на сервере в реальном времени. События фильтруются согласно выбранному фильтру событий 762 или при помощи пользовательских правил фильтра, определяемых сервером во время выполнения UI процедуры Показать журнал событий 96.

Количество отображаемых событий ограничено 1000. Если число событий оказывается выше, самое старое событие будет удаляться по мере появления нового.

Инструментальная панель текущих событий

Раскрывающийся список выбирает фильтр событий 762, который будет использоваться для фильтрации событий, отображаемых в журнале.

Отключить текущий фильтр. Данная операция позволяет вводить новые параметры фильтра.

Конфигурировать текущий фильтр. Изменяет настройки выбранного фильтра.

Просмотр статистики серьезности событий. Отображает количество событий каждого уровня 73 в окне Текущие события.

Экспорт. Экспортирует 311 события, отображаемые на данный момент в окне Текущие события, в файл.

Очистить текущие события. Очистка таблицы текущих событий без удаления данных событий, новые события не появляются в таблице.

7.9.8.9.2 История событий

История событий отображает события, произошедшие в прошлом и сохраненные сервером. События фильтруются согласно выбранному на данный момент Фильтру событий 762 или при помощи пользовательских правил фильтрации, заданных сервером во время выполнения GUI процедуры Показать журнал событий 96.

В истории сохраняются не все события. Таким образом, событие, которое появилось в разделе Текущие события может и не появиться в Истории событий.

Инструментальная панель Истории событий

Фильтр событий 762, выбранный из раскрывающегося списка, используется для фильтрации отображаемых событий.

Отключить текущий фильтр. Данная операция позволяет вводить новые параметры фильтра.

Конфигурировать текущий фильтр. Изменяет настройки выбранного фильтра.

Просмотр статистики серьезности событий. Отображает количество событий каждого уровня 73 в загруженной истории событий.

Экспорт. Экспортирует 311 события, отображаемые в окне История событий, в файл.

Экспортируются только события, выбранные в заданном пределе. Для экспорта большего количества событий, увеличьте значение в комбинированном списке **События** (см. далее).

Обновить. Обновляет историю событий. Данная операция приводит к тому, что сервер заново загружает историю согласно выбранному фильтру. Видимые события могут измениться после данной операции.

В начало. Переход на первую страницу истории.

Назад. Загружает предыдущую страницу истории.

Вперед. Загружает следующую страницу истории.

В конец. Переход на последнюю страницу истории.

Комбинированный список **События** определяет, сколько событий будут отображаться одновременно.

Общее количество событий согласно текущему фильтру и пределу отображаемых событий указано в правой части инструментальной панели.

7.9.8.9.3 Контекстное меню

Контекстное меню открывается при нажатии правой кнопкой мыши на одном из событий в Журнале событий и включает в себя следующие пункты:

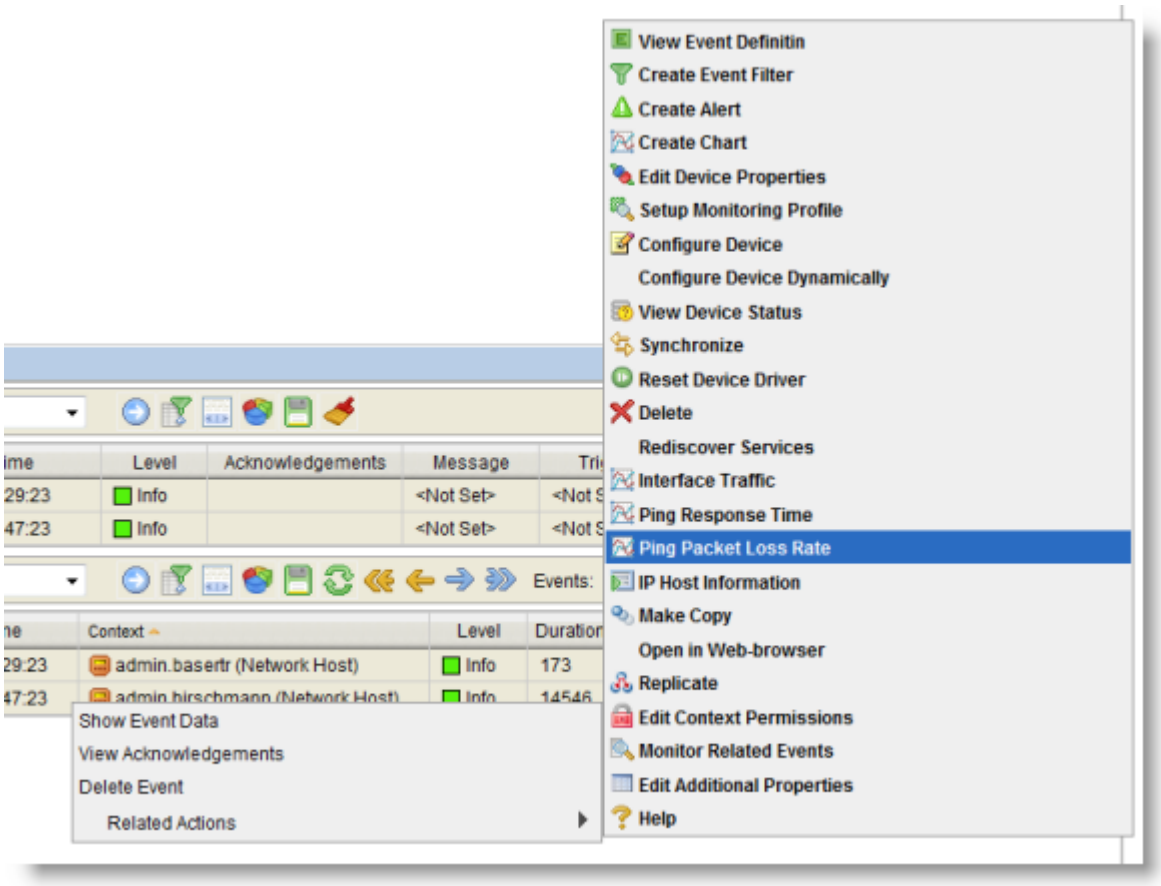
- **Показать данные события.** Открывает Таблицу данных 49, относящуюся к событию, в Редакторе таблицы данных 282.

- **Подтвердить событие.** Позволяет пользователю установить подтверждение события. Данная операция доступна только для постоянных пользователей.
- **Посмотреть подтверждения**. Показывает подтверждения событий в Редакторе таблицы данных 282.
- **Удалить событие.** Удаляет событие безвозвратно из истории событий. Данная операция доступна только для постоянных пользователей.

Контекстное меню события также содержит подменю Относительные действия со списком действий 87, относящихся к контексту 41, где возникло событие или относящиеся к самому событию 102.

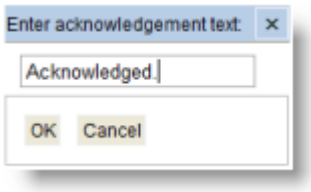
Например, для события, относящегося к устройству, могут быть отображены следующие операции: "Перезагрузить устройство", "Конфигурировать устройство" и др.

Пример контекстного меню события:

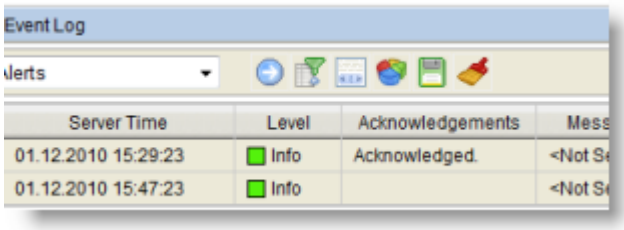


7.9.8.9.4 Подтверждение событий

Функция подтверждения 73 события доступна только для постоянных пользователей. При активации пункта меню **Подтвердить событие**, пользователю предлагается ввести текст подтверждения:



Когда подтверждение выполнено, его время, автор и текст появляются в столбце Подтверждения таблицы событий:



7.9.8.9.5 Экспорт событий

Экспорт журнала событий в AtomMind Web Desktop похож на экспорт журнала событий[402] AtomMind Client.

7.9.8.10 Сниппет HTML

Сниппет HTML - это компонент, который отображает содержимое, отформатированное с помощью HTML.

Общие переменные (свойства) [?[231]]

Общие переменные: Имя[232], Видимый[232], Свойства расширения[232], Ширина сетки[233], Высота сетки[233], Выход за пределы контента[233], Стиль контейнера[234], Пользовательские классы[234], Стиль[234]

СОДЕРЖИМОЕ HTML

Содержимое HTML для отображения. Заключать содержимое в тэги *<html></html>* не требуется.

Имя переменной: htmlContent

Записи: 1

Формат[50] записи:

Имя поля	Тип поля	Примечания
htmlContent	String	Содержимое HTML

СОЗДАВАТЬ СОБЫТИЯ "НАЖАТИЕ МЫШИ"

Этот флаг показывает, будет ли компонент создавать события Нажатие мыши.

Имя переменной: generateMouseClicked

Записи: 1

Формат[50] записи:

Имя поля	Тип поля	Примечания
generateMouseClicked	Boolean	Создавать события Нажатие мыши

СОЗДАВАТЬ СОБЫТИЯ "ДВОЙНОЙ ЩЕЛЧОК МЫШИ"

Этот флаг показывает, будет ли компонент создавать события Двойной щелчок мыши.

Имя переменной: generateMouseDoubleClicked

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
generateMouseDoubleClicked	Boolean	Создавать события Двойной щелчок мыши

СОЗДАВАТЬ СОБЫТИЯ "НАВЕДЕНИЕ УКАЗАТЕЛЯ МЫШИ"

Этот флаг показывает, будет ли компонент создавать события Наведение указателя мыши.

Имя переменной: generateMouseEntered

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
generateMouseEntered	Boolean	Создавать события Наведение указателя мыши

СОЗДАВАТЬ СОБЫТИЯ "ВЫВОД КУРСОРА МЫШИ"

Этот флаг показывает, будет ли компонент создавать события Вывод курсора мыши.

Имя переменной: generateMouseExited

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
generateMouseExited	Boolean	Создавать события Вывод курсора мыши

СОЗДАВАТЬ СОБЫТИЯ "НАЖАТИЕ КЛАВИШИ"

Этот флаг показывает, будет ли компонент создавать события Нажатие клавиши.

Имя переменной: generateKeyPressed

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
generateKeyPressed	Boolean	Создавать события Нажатие клавиши

СОЗДАВАТЬ СОБЫТИЯ "ОТПУСКАНИЕ КЛАВИШИ"

Этот флаг показывает, будет ли компонент создавать события Отпускание клавиши.

Имя переменной: generateKeyReleased

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
generateKeyReleased	Boolean	Создавать события Отпускание клавиши

Общие события [? 239]

НАЖАТИЕ МЫШИ

Данное событие возникает, когда пользователь кликает мышью внутри компонента.

Имя события: mouseClicked

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	ID события.
value	Data Table	Содержит информацию, относящуюся к событию. Эта вложенная таблица данных имеет одно поле: • **ID** - имя соответствующего компонента, например, *htmlSnippet0*.

ДВОЙНОЙ ЩЕЛЧОК МЫШИ

Данное событие возникает при двойном щелчке мыши внутри компонента.

Имя события: mouseDoubleClicked

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	ID события.
value	Data Table	Содержит информацию, относящуюся к событию. Эта вложенная таблица данных имеет одно поле: • **ID** - имя соответствующего компонента, например, *htmlSnippet0*.
level	Integer	Уровень 75 события по умолчанию.

НАВЕДЕНИЕ УКАЗАТЕЛЯ МЫШИ

Данное событие возникает, когда пользователь наводит курсор мыши на область компонента.

Имя события: mouseEntered

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	ID события.
value	Data Table	Содержит информацию, относящуюся к событию. Эта вложенная таблица данных имеет одно поле: • **ID** - имя соответствующего компонента, например, *htmlSnippet0*.
level	Integer	Уровень 75 события по умолчанию.

ВЫВОД КУРСОРА МЫШИ

Данное событие возникает, когда пользователь выводит курсор мыши из области компонента.

Имя события: mouseExited

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	ID события.
value	Data Table	Содержит информацию, относящуюся к событию. Эта вложенная таблица данных имеет одно поле: • **ID** - имя соответствующего компонента, например, *htmlSnippet0*.
level	Integer	Уровень 75 события по умолчанию.

НАЖАТИЕ КЛАВИШИ

Данное событие возникает при нажатии пользователем на клавишу, когда компонент находится в фокусе.

Имя события: keyPressed

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	ID события.
value	Data Table	Содержит информацию, относящуюся к событию. Эта вложенная таблица данных имеет одно поле: • **ID** - имя соответствующего компонента, например, *htmlSnippet0*.
level	Integer	Уровень 75 события по умолчанию.

ОТПУСКАНИЕ КЛАВИШИ

Данное событие возникает при нажатии пользователем клавиши, когда компонент находится в фокусе.

Имя события: keyReleased

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	ID события.
value	Data Table	Содержит информацию, относящуюся к событию. Эта вложенная таблица данных имеет одно поле: • **ID** - имя соответствующего компонента, например, *htmlSnippet0*.
level	Integer	Уровень 75 события по умолчанию.

7.9.9 Управление вводом

Этот раздел описывает элементы **Управления вводом**.

7.9.9.1 Текстовое поле

Текстовое поле - это компонент, который позволяет редактировать отдельную строку текста.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

ТЕКСТ

Текст, содержащийся в текстовом поле.

Имя переменной: text

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
text	String	Текст

ЗАПОЛНИТЕЛЬ

Сообщение, которое отображается в текстовом поле, когда никакой другой текст не введен.

Имя переменной: placeholder

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания

placeholder	String	Заполнитель

МАСКА ВВОДА

Позволяет указать допустимые символы для ввода с использованием регулярного выражения, которое может включать следующие шаблоны:

- `'9'` - соответствует любой цифре, т.е. [0-9]
- `'a'` - соответствует любой букве, т.е. [a-zA-Z]
- `'*'` - соответствует любой букве или цифре, т.е. [a-zA-Z0-9]

Имя переменной: inputMask

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
inputMask	String	Маска ввода

ИКОНКА СЛЕВА

Изображение, отображаемое слева от поля ввода.

Имя переменной: leftIcon

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
leftIcon	Data Block	Иконка слева

ИКОНКА СПРАВА

Изображение, отображаемое справа от поля ввода.

Имя переменной: rightIcon

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
rightIcon	Data Block	Иконка справа

ИСПОЛЬЗОВАТЬ СОБЫТИЕ KEYUP

Если включено, компонент будет создавать событие **keyUp**, содержащее текущие входные данные, каждый раз, когда пользователь отпускает клавишу клавиатуры.

Имя переменной: useKeyUp

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
useKeyUp	Boolean	Использовать событие keyUp

ЗАДЕРЖКА ПОСЛЕ СОБЫТИЯ KEYUP

Определяет задержку между отпусканием клавиши клавиатуры и созданием события **keyUp**. Если в это время будет нажата любая другая клавиша, событие не возникнет, и отсчет времени задержки начнется заново.

Имя переменной: keyUpDelay

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
keyUpDelay	Integer	Задержка после события keyUp

МАКСИМАЛЬНАЯ ДЛИНА

Максимальное число символов, которое можно ввести в текстовое поле

Имя переменной: maxLength

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
maxLength	Integer	Максимальная длина

НЕАКТИВНЫЙ

Этот флаг показывает, активен ли компонент. Неактивные компоненты не отвечают на ввод пользователя и недоступны для выбора.

Имя переменной: disabled

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
disabled	Boolean	Неактивный

ТИП ВВОДА

Определяет тип ввода. Данное свойство имеет несколько возможных значений:

- **Search -** добавляет иконку Поиск в поле ввода. Вводимый текст будет отображаться как обычно.
- **Text -** вводимый текст будет отображаться как обычно.
- **Password -** вводимый текст будет скрыт за звездочками.

Имя переменной: inputType

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
inputType	String	Тип ввода

ИСПОЛЬЗОВАТЬ ИКОНКУ "ОЧИСТИТЬ"

Если включено, добавляет кнопку Очистить, которая появляется только если введен какой-либо текст.

Имя переменной: addClearIcon

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
addClearIcon	Boolean	Использовать иконку Очистить

СТИЛЬ КОНТЕЙНЕРА КАРТИНКИ

Стиль CSS для применения к изображению контейнера.

Имя переменной: iconsStyle

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
iconsStyle	Boolean	Стиль контейнера картинки

ВАЛИДАТОРЫ

Это свойство позволяет определить валидаторы для поля ввода.

Имя переменной: validators

Записи: 0...не ограничено

Формат записи:

Имя поля	Тип поля	Примечания
validatorType	String	**Type**. Тип валидатора. Это свойство имеет несколько возможных значений: • **Non-null Validator -** вводимые данные не могут быть Null, т.е. нужно ввести хотя бы один символ; • **Limits Validator -** вводимые данные должны находиться в рамках, указанных в поле *options*. Например, значение "*0, 100*" (без кавычек) ограничивает ввод любых значений меньше 0 и больше 100; • **Regex Validator -** вводимые данные должны соответствовать регулярному выражению, указанному в поле *options*.
options	String	**Options**. Дополнительные опции для выбранного типа валидатора.
message	String	**Message**. Сообщение, показывающее, соответствует ли введенный символ текущему валидатору.

ДЕЙСТВИТЕЛЬНЫЙ ПО УМОЛЧАНИЮ

Определяет состояние компонента по умолчанию. Если отключено, компонент будет отображаться с недействительным состоянием ввода по умолчанию.

Имя переменной: componentValid

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
componentValid	Boolean	Действительный по умолчанию

Общие события [? 239]

Общие события: Изменение входного значения 240

7.9.9.2 Текстовая область

Текстовая область - это компонент, который отображает множество редактируемых строк текста.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

ТЕКСТ

Текст, содержащийся в текстовой области.

Имя переменной: text

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
text	String	Текст

ЗАПОЛНИТЕЛЬ

Сообщение, которое отображается в текстовой области, когда никакой другой текст не введен.

Имя переменной: placeholder

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
placeholder	String	Заполнитель

НЕАКТИВНЫЙ

Этот флаг показывает, активен ли компонент. Неактивные компоненты не отвечают на ввод пользователя и недоступны для выбора.

Имя переменной: disabled

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
disabled	Boolean	Неактивный

ИЗМЕНЯЕМЫЙ

Определяет возможность изменять размер текстовой области путем перетаскивания. Данное свойство имеет несколько возможных значений:

- **Both** - пользователь может менять текстовой области с обоих измерениях
- **Horizontal -** пользователь может менять только горизонтальный размер текстовой области
- **Vertical -** пользователь может менять только вертикальный размер текстовой области
- **None -** изменение размера отключено

Имя переменной: resize

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
resize	String	Изменяемый

Общие события [? 239]

Общие события: Изменение входного значения 240

7.9.9.3 Числовое поле

Однострочное поле ввода, которое позволяет пользователям выбирать число или значение из упорядоченной последовательности. В числовом поле имеются две маленькие кнопки со стрелками для изменения числового значения. Для этой цели также можно использовать клавиши со стрелками вверх/вниз на клавиатуре. Пользователь также может ввести (допустимое) значение непосредственно в числовое поле.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

МИНИМАЛЬНОЕ ЗНАЧЕНИЕ

Минимальное числовое значение, которое пользователь может ввести в числовое поле.

Имя переменной: minValue

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
minValue	Double	Минимальное значение

МАКСИМАЛЬНОЕ ЗНАЧЕНИЕ

Максимальное числовое значение, которое пользователь может ввести в числовое поле.

Имя переменной: maxValue

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
maxValue	Double	Максимальное значение

ШАГ

Количество, на которое уменьшается или увеличивается текущее значение при использовании кнопок со стрелками при вводе.

Имя переменной: step

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
step	Double	Шаг

ЕДИНИЦА ИЗМЕРЕНИЯ

Текст единицы изменения, который будет отображаться после числа.

Имя переменной: unit

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
unit	String	Единица измерения

НЕАКТИВНЫЙ

Этот флаг показывает, активен ли компонент. Неактивные компоненты не отвечают на ввод пользователя и недоступны для выбора.

Имя переменной: disabled

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
disabled	Boolean	Неактивный

ЗНАЧЕНИЕ

Числовое значение, содержащееся в числовом поле.

Имя переменной: value

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
value	Double	Значение

ТОЧНОСТЬ

Количество знаков после запятой.

Имя переменной: presicion

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
presicion	Integer	Точность

ОБщие события [? 239]

Общие события: отсутствуют

7.9.9.4 Выпадающий список

Этот компонент совмещает в себе редактируемое поле и выпадающий список. Пользователь может выбрать значение из выпадающего списка. Если сделать редактируемым поле со списком, то пользователь сможет вводить в него значение.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

ЗАПОЛНИТЕЛЬ

Сообщение, которое отображается в поле ввода, когда никакой другой текст не введен.

Имя переменной: placeholder

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
placeholder	String	Заполнитель

ВЫБРАТЬ РАЗМЕР

Определяет размер компонента. Данное свойство имеет несколько возможных значений:

- **Small Size**
- **Default Size**
- **Large Size**

Имя переменной: selectSize

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
selectSize	String	Выбрать размер

ТРЕБУЕТСЯ ПОДТВЕРЖДЕНИЕ

Если включено, отображает окно подтверждения после начальной выборки.

Имя переменной: confirmationRequred

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
confirmationRequred	Boolean	Требуется подтверждение

ТЕКСТ ПОДТВЕРЖДЕНИЯ

Определяет текст сообщения о подтверждении.

Имя переменной: confirmationMessage

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
confirmationMessage	String	Текст подтверждения

МНОЖЕСТВЕННЫЙ ВЫБОР

Если включено, разрешает выбирать несколько элементов из выпадающего списка.

Имя переменной: multi

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
multi	Boolean	Множественный выбор

МАКСИМАЛЬНОЕ КОЛИЧЕСТВО ТЕГОВ

Определяет максимальное количество тегов для отображения.

Имя переменной: maxTagCount

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
maxTagCount	Integer	Максимальное количество тегов

ЗАПОЛНИТЕЛЬ ДЛЯ СКРЫТЫХ ТЕГОВ

Текст дополнительного тега, который отображается при превышении значения свойства **Максимальное количество тегов**.

Имя переменной: maxTagPlaceholder

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
maxTagPlaceholder	String	Заполнитель для скрытых тегов

ПОИСК

Если включено, разрешает поиск текста ввода по элементам выпадающего списка.

Имя переменной: searchable

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
searchable	Boolean	Поиск

ОПЦИИ

Это свойство определяет набор допустимых значений в выпадающем списке.

Имя переменной: options

Записи: 0..не ограничено

Формат записи:

Имя поля	Тип поля	Примечания
value	String	**Value**. Уникальное значение элемента. Обычно используется для получения доступа к конкретному элементу с помощью привязок.
label	String	**Label**. Текст, описывающий элемент, для отображения в выпадающем списке.
tooltip	String	**Tooltip**. Текст всплывающей подсказки для элемента.

ВЫБРАННЫЕ ЭЛЕМЕНТЫ

Это свойство определяет одно допустимое значение или набор допустимых значений, выбираемых по умолчанию.

Имя переменной: currentOptions

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
value	String	**Value**. Уникальное значение элемента. Обычно используется для получения доступа к конкретному элементу с помощью привязок.
label	String	**Label**. Отображаемый текст допустимого значения.
tooltip	String	**Tooltip**. Текст всплывающей подсказки для допустимого значения.

ВСЕГДА ПОКАЗЫВАТЬ ЗАПОЛНИТЕЛЬ

Если включено, заполнитель будет отображаться всегда, независимо от наличия текста.

Имя переменной: alwaysShowPlaceholder

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
alwaysShowPlaceholder	Boolean	Всегда показывать заполнитель

РАЗРЕШИТЬ СБРОС

Этот флаг определяет, можно ли сбросить поле ввода.

Имя переменной: allowClear

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
allowClear	String	Разрешить сброс

ПОКАЗЫВАТЬ СТРЕЛКУ

Этот флаг определяет, будет ли отображаться значок со стрелкой внутри поля ввода.

Имя переменной: showArrow

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания

showArrow	Boolean	Показывать стрелку

НЕАКТИВНЫЙ

Этот флаг показывает, активен ли компонент. Неактивные компоненты не отвечают на ввод пользователя и недоступны для выбора.

Имя переменной: disabled

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
disabled	Boolean	Неактивный

Общие события [? 239]

Общие события: Изменение входного значения 240.

7.9.9.5 Выпадающее дерево

Компонент Выпадающее дерево представляет собой редактор иерархических данных, отрисованных в виде дерева. Это дает возможность выбирать несколько элементов, а также создавать пользовательские узлы дерева.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

ДАННЫЕ ДЕРЕВА

Данное свойство определяет элементы и структуру выпадающего дерева.

Имя переменной: treeData

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
value	String	**Value**. Уникальное значение элемента. Обычно используется для получения доступа к конкретному элементу с помощью привязок.
title	String	**Title**. Текст, описывающий элемент, для отображения в выпадающем дереве.
children	Data Table	**Children**. Определяет дочерние элементы определенного элемента.

ЗАПОЛНИТЕЛЬ

Сообщение, которое отображается в поле ввода, когда никакой другой текст не введен.

Имя переменной: placeholder

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
placeholder	String	Заполнитель

ЗАПОЛНИТЕЛЬ В ПОИСКЕ

Сообщение, которое отображается в поле ввода, когда никакой другой текст не введен в режиме поиска.

Имя переменной: searchPlaceholder

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
searchPlaceholder	String	Заполнитель в поиске

ПОКАЗЫВАТЬ ПОИСК

Если включено, разрешает поиск текста ввода по элементам выпадающего дерева.

Имя переменной: showSearch

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
showSearch	Boolean	Показывать поиск

ВЫБОР В ДЕРЕВЕ

Если включено, добавляет ячейки с флажками выбора к элементам выпадающего дерева.

Имя переменной: treeCheckable

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
treeCheckable	Boolean	Выбор в дереве

МНОЖЕСТВЕННЫЙ ВЫБОР

Если включено, позволяет выбирать несколько элементов выпадающего дерева.

Имя переменной: multi

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания

multi	Boolean	Множественный выбор

РАЗВОРАЧИВАТЬ ВСЕ УЗЛЫ ПО УМОЛЧАНИЮ

Если включено, все элемента выпадающего дерева будут развернуты по умолчанию.

Имя переменной: treeDefaultExpandAll

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
treeDefaultExpandAll	Boolean	Разворачивать все узлы по умолчанию

НЕАКТИВНЫЙ

Этот флаг показывает, активен ли компонент. Неактивные компоненты не отвечают на ввод пользователя и недоступны для выбора.

Имя переменной: disabled

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
disabled	Boolean	Неактивный

РАЗВЁРНУТЫЕ УЗЛЫ

Определяет элементы выпадающего дерева, которые будут развернуты по умолчанию.

Имя переменной: expandedNodes

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
value	String	**Value**. Значение разворачиваемого узла.

ОТМЕЧЕННЫЕ УЗЛЫ

Определяет элементы выпадающего дерева, которые будут выбираться по умолчанию.

Имя переменной: checkedNodes

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
value	String	**Value**. Значение выбираемого узла.

СТИЛЬ ВЫПАДАЮЩЕГО ДЕРЕВА

Стиль CSS для применения к выпадающему дереву.

Имя переменной: dropdownStyle

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
dropdownStyle	String	Стиль выпадающего дерева

Общие события [? 239]

Общие события: отсутствуют

7.9.9.6 Выпадающее меню

Данный компонент позволяет создавать кнопку с выпадающим меню, которое появляется при нажатии мышью.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

МЕТКА

Текст метки для кнопки с выпадающим меню.

Имя переменной: label

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
label	String	Метка

ИЗОБРАЖЕНИЕ

Изображение для отображения на кнопке с выпадающим меню.

Имя переменной: image

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
image	Data Block	Изображение

НЕАКТИВНЫЙ

Этот флаг показывает, активен ли компонент. Неактивные компоненты не отвечают на ввод пользователя и недоступны для выбора.

Имя переменной: disabled

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
disabled	Double	Неактивный

РАЗМЕЩЕНИЕ

Определяет, в каком положении будет появляться выпадающее меню. Данное свойство имеет несколько возможных значений:

- Снизу слева
- Снизу по центру
- Снизу справа
- Сверху слева
- Сверху по центру
- Сверху справа

Имя переменной: placement

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
placement	String	Размещение

ТРИГГЕР

Определяет действие, при котором будет появляться выпадающее меню. Данное свойство имеет несколько возможных значений:

- **Click -** всплывающее меню появится при нажатии на левую кнопку мыши
- **Hover -** всплывающее меню появится при наведении мыши
- **Context Menu -** всплывающее меню появится при нажатии на правую кнопку мыши

Имя переменной: trigger

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
trigger	Boolean	Триггер

ЭЛЕМЕНТЫ ВЫПАДАЮЩЕГО СПИСКА

Данное свойство определяет перечень элементов меню, отображаемых компонентом.

Имя переменной: dropDownItems

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
image	Data Block	**Image**. Изображение, которое будет отображаться на определенном элементе выпадающего меню.
key	String	**Key**. Уникальный ключ определенного элемента выпадающего меню, который можно использовать для определения графической презентации и поведения группы элементов меню с тем же ключом, используя CSS.
type	String	**Type**. Определяет тип определенного элемента выпадающего меню. Данное свойство имеет два возможных значения: • **Text -** элемент меню будет вести себя как регулярный текстовый элемент меню • **Link -**элемент меню будет вести себя как элемент с URL и открывать указанный URL в новой вкладке веб=браузера.
value	String	**Value**. Значение определенного элемента выпадающего меню. Если выбран тип *link*, здесь указывается URL.
description	String	**Description**. Описание элемента выпадающего меню, которое будет отображаться в выпадающем меню.
disabled	Boolean	**Disabled**. Этот флаг определяет, активен ли определенный элемент выпадающего меню. Неактивные компоненты не отвечают на ввод пользователя и недоступны для выбора.

Общие события [? 239]

Общие события: Изменение входного значения 240

7.9.9.7 Кнопка

Компонент, используемый для отображения обычной кнопки.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

МЕТКА

Текст метки кнопки.

Имя переменной: label

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
label	String	Метка

ИЗОБРАЖЕНИЕ

Изображение для отображения на кнопке.

Имя переменной: image

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
image	Data Block	Изображение

НЕАКТИВНЫЙ

Этот флаг показывает, активен ли компонент. Неактивные компоненты не отвечают на ввод пользователя и недоступны для выбора.

Имя переменной: disabled

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
disabled	Boolean	Неактивный

ТИП ТЕМЫ

Позволяет переключаться между вариантами графического представления кнопок. Это свойство имеет следующие возможные значения:

- **Main -** белый текст, бледно-голубой цвет кнопки
- **Secondary -** серый текст и границы, белый цвет кнопки
- **Text -** только серый цвет

Каждый из стилей кнопки можно менять при помощи относящихся к CSS свойств, например Стиль.

Имя переменной: buttonTheme

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
buttonTheme	String	Theme Type

СТИЛЬ ПРИ НАВЕДЕНИИ

CSS стиль для применения к кнопке при наведении.

Имя переменной: hoverStyle

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
hoverStyle	String	Стиль при наведении

АКТИВНЫЙ СТИЛЬ

CSS стиль для применения к кнопке в статусе Активна.

Имя переменной: activeStyle

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
activeStyle	String	Активный стиль

СТИЛЬ ПРИ ФОКУСЕ

CSS стиль для применения к кнопке при фокусе.

Имя переменной: focusStyle

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
focusStyle	String	Стиль при фокусе

СТИЛЬ КОНТЕЙНЕРА КАРТИНКИ

CSS стиль для применения к контейнеру изображения, указанного в свойстве *image*.

Имя переменной: imageButtonContainerStyle

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
imageButtonContainerStyle	String	Стиль контейнера картинки

СТИЛЬ ТЕКСТА КНОПКИ

CSS стиль для применения к тексту метки кнопки, указанному в свойстве *buttonName*.

Имя переменной: textButtonContainerStyle

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
textButtonContainerStyle	String	Стиль текста кнопки

Общие события [? 239]

Общие события: Нажатие мыши 239

7.9.9.8 Группа кнопок

Этот компонент может использоваться для группировки связанных кнопок.

Общие переменные (свойства) [?231]

Общие переменные: Имя232, Видимый232, Свойства расширения232, Ширина сетки233, Высота сетки233, Выход за пределы контента233, Стиль контейнера234, Пользовательские классы234, Стиль234

КНОПКИ

Это свойство определяет перечень кнопок, отображаемых компонентом.

Имя переменной: buttons

Записи: 0..не ограничено

Формат50 записи:

Имя поля	Тип поля	Примечания
id	String	**ID**. Уникальный текстовый идентификатор определенной кнопки.
image	Data Block	**Image**. Изображение для отображения на кнопке.
text	String	**Text**. Текст метки кнопки.
buttonCustomClasses	String	**Custom Classes**. Позволяет указать пользовательские классы CSS, которые могут использоваться в свойствах Стиль234, **Стиль при наведении**, **Активный стиль** и **Стиль при фокусе**.

ВЫБРАННЫЕ ЭЛЕМЕНТЫ

ID кнопки для добавления к HTML свойству *selected*. Может использоваться для определения графического представления и поведения данной группы кнопок с использованием CSS.

Имя переменной: selectedButtons

Записи: 0..не ограничено

Формат50 записи:

Имя поля	Тип поля	Примечания
id	String	ID

СТИЛЬ ПРИ НАВЕДЕНИИ

CSS стиль для применения к кнопке при наведении.

Имя переменной: hoverStyle

Записи: 1

Формат50 записи:

Имя поля	Тип поля	Примечания
hoverStyle	String	Стиль при наведении

АКТИВНЫЙ СТИЛЬ

CSS стиль для применения к кнопке в статусе Активна.

Имя переменной: activeStyle

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
activeStyle	String	Активный стиль

СТИЛЬ ПРИ ФОКУСЕ

CSS стиль для применения к кнопке при фокусе.

Имя переменной: focusStyle

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
focusStyle	String	Стиль при фокусе

ОБщие события [? 235]

Общие события: отсутствуют

7.9.9.9 Дата/время

Этот компонент используется для ввода определенного значения даты. Он сочетает в себе редактируемое поле ввода и выпадающий список с возможностью выбора даты.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

ФОРМАТ

Определяет формат отображения даты. Строка формата может содержать следующие токены:

	Токен	Выход
Месяц	M	1 2 ... 11 12
	Mo	1st 2nd ... 11th 12th
	MM	01 02 ... 11 12
	MMM	Jan Feb ... Nov Dec
	MMMM	January February ... November December
Квартал	Q	1 2 3 4
	Qo	1st 2nd 3rd 4th

День месяца	D	1 2 ... 30 31
	Do	1st 2nd ... 30th 31st
	DD	01 02 ... 30 31
День года	DDD	1 2 ... 364 365
	DDDo	1st 2nd ... 364th 365th
	DDDD	001 002 ... 364 365
День недели	d	0 1 ... 5 6
	do	0th 1st ... 5th 6th
	dd	Su Mo ... Fr Sa
	ddd	Sun Mon ... Fri Sat
	dddd	Sunday Monday ... Friday Saturday
День недели (Locale)	e	0 1 ... 5 6
День недели (ISO)	E	1 2 ... 6 7
Неделя года	w	1 2 ... 52 53
	wo	1st 2nd ... 52nd 53rd
	ww	01 02 ... 52 53
Неделя года (ISO)	W	1 2 ... 52 53
	Wo	1st 2nd ... 52nd 53rd
	WW	01 02 ... 52 53
Год	YY	70 71 ... 29 30
	YYYY	1970 1971 ... 2029 2030
	YYYYYY	-001970 -001971 ... +001907 +001971 Важно: Расширенные годы (Охватывают полный диапазон значений времени примерно 273,790 лет до и после от 01 января 1970 г.)
	Y	1970 1971 ... 9999 +10000 +10001 Важно: соответствует стандарту ISO 8601 для дат после 9999 года
Год эры	y	1 2 ... 2020 ...
Эра	N, NN, NNN	BC AD Важно: аббревиатура названия эры
	NNNN	Before Christ, Anno Domini Важно: полное название эры
	NNNNN	BC AD Важно: краткое название эры
Недель в году	gg	70 71 ... 29 30
	gggg	1970 1971 ... 2029 2030
Недель в году (ISO)	GG	70 71 ... 29 30
	GGGG	1970 1971 ... 2029 2030
AM/PM	A	AM PM
	a	am pm

Час	H	0 1 ... 22 23
	HH	00 01 ... 22 23
	h	1 2 ... 11 12
	hh	01 02 ... 11 12
	k	1 2 ... 23 24
	kk	01 02 ... 23 24
Минута	m	0 1 ... 58 59
	mm	00 01 ... 58 59
Секунда	s	0 1 ... 58 59
	ss	00 01 ... 58 59
Доля секунды	S	0 1 ... 8 9
	SS	00 01 ... 98 99
	SSS	000 001 ... 998 999
	SSSS ... SSSSSSSSS	000[0..] 001[0..] ... 998[0..] 999[0..]
Временная зона	z or zz	EST CST ... MST PST
	Z	-07:00 -06:00 ... +06:00 +07:00
	ZZ	-0700 -0600 ... +0600 +0700
Временная метка Unix	X	1360013296
Временная метка Unix в миллисекундах	x	1360013296123

Имя переменной: format

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
format	String	Формат

НАЧАЛЬНАЯ ДАТА

Значение по умолчанию отдельного указателя даты или начальная дата, если активировано свойство **Диапазон дат**.

Имя переменной: startDate

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
startDate	Date	Начальная дата

ДИАПАЗОН ДАТ

Если включено, позволяет выбирать диапазон даты (т.е. начальную и конечную даты) вместо единичной даты.

Имя переменной: range

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
range	Boolean	Диапазон дат

КОНЕЧНАЯ ДАТА

Значение конечной даты по умолчанию на указателе даты. Это свойство имеет смысл только при активированном свойстве **Диапазон дат**.

Имя переменной: endDate

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
endDate	Date	Конечная дата

КНОПКИ ДИАПАЗОНА

Данное свойство позволяет создавать кнопки диапазона на всплывающем списке с возможностью выбора даты. При нажатии на эти кнопки выбирается преднастроенный диапазон дат.

Имя переменной: rangeButtons

Записи: 0..не ограничено

Формат записи:

Имя поля	Тип поля	Примечания
name	String	**Name**. Текст метки кнопки.
range	Integer	**Range**. Число элементов для выбора.
direction	String	**Direction**. Определяет направление выбора даты. Данное свойство имеет несколько возможных значений: • **Forward -** выбирает число элементов, начиная с текущей даты; • **Today -** выбирает только текущую даты. Свойства *Range*, *step* и *dynamicStep* будут проигнорированы; • **Back -** выбирает число элементов до текущей даты.
step	String	**Step**. Определяет единицу измерения, используемую в свойстве *range*. Данное свойство имеет несколько возможных значений: • **Hour** • **Day** • **Month**
dynamicStep	Boolean	**Dynamic Step**. Позволяет сдвигать определенный диапазон на значение, указанное в свойстве *range,* при каждом нажатии кнопки.

НЕАКТИВНЫЙ

Этот флаг показывает, активен ли компонент. Неактивные компоненты не отвечают на ввод пользователя и недоступны для выбора.

Имя переменной: disabled

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
disabled	Boolean	Неактивный

ОБщие события [? 239]

Общие события: отсутствуют

7.9.9.10 Время

Этот компонент используется для ввода определенного значения времени. Он сочетает в себе редактируемое поле ввода и выпадающий список с возможностью выбора времени.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

ЗНАЧЕНИЕ

Значение времени, отображаемое/редактируемое компонентом.

Имя переменной: value

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
value	Date	Значение

ФОРМАТ

Определяет формат отображения времени (например, `HH:mm:ss` или `mm-ss`). Может включать различные токены 335.

Имя переменной: format

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
format	String	Формат

РАЗРЕШИТЬ СБРОС

Этот флаг разрешает очищать поле ввода.

Имя переменной: allowClear

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
allowClear	Boolean	Разрешить сброс

ИСПОЛЬЗОВАТЬ 12 ЧАСОВ

Этот флаг разрешает отображать время в 12-часовом формате. Обратите внимание, чтобы различать время до и после полудня, необходимо добавлять `a` к формату времени (например, `HH:mm:ss a`).

Имя переменной: use12Hours

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
use12Hours	Boolean	Использовать 12 часов

ШАГ ЧАСОВ

Определяет интервал между часами на указателе времени.

Имя переменной: hourStep

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
hourStep	Integer	Шаг часов

ШАГ МИНУТ

Определяет интервал между минутами на указателе времени.

Имя переменной: minuteStep

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
minuteStep	Integer	Шаг минут

ШАГ СЕКУНД

Определяет интервал между секундами на указателе времени.

Имя переменной: secondStep

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
secondStep	Integer	Шаг секунд

ЗАПОЛНИТЕЛЬ

Определяет, какой текст будет отображаться в незаполненном поле ввода.

Имя переменной: placeholder

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
placeholder	String	Заполнитель

НЕАКТИВНЫЙ

Этот флаг показывает, активен ли компонент. Неактивные компоненты не отвечают на ввод пользователя и недоступны для выбора.

Имя переменной: disabled

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
disabled	Boolean	Неактивный

ОБщие события [? 239]

Общие события: отсутствуют

7.9.9.11 Триггерная кнопка

Данный компонент позволяет переключаться между двумя состояниями настройки.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

АКТИВНЫЙ

Определяет, включена ли триггерная кнопка.

Имя переменной: checked

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания

checked	Boolean	Активный

РАЗМЕР

Определяет размер триггерной кнопки:

- **Default** - будет отрисована триггерная кнопка обычного размера
- **Small -** будет отрисована триггерная кнопка маленького размера

Имя переменной: size

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
size	String	Размер

НЕАКТИВНЫЙ

Этот флаг показывает, активен ли компонент. Неактивные компоненты не отвечают на ввод пользователя и недоступны для выбора.

Имя переменной: disabled

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
disabled	Boolean	Неактивный

ОБщие события [? 239]

Общие события: отсутствуют

7.9.9.12 Флаговая кнопка

Элемент, который может быть выбран, или чей выбор может быть отменен. Статус показывается пользователю.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

МЕТКА

Текст ярлыка флаговой кнопки.

Имя переменной: label

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания

label	String	Метка

СОСТОЯНИЕ ВЫБОРА

Состояние флаговой кнопки по умолчанию. Данное свойство имеет несколько возможных значений:

- **Selected ("selected") -** флаговая кнопка будет отрисована с пометкой внутри
- **Unselected ("unselected") -** флаговая кнопка будет отрисована с без пометок
- **Partially Selected ("partiallySelected") -** флаговая кнопка будет отрисована с квадратом внутри (обычно используется, когда часть дочерних флажков выбрана, а остальные - нет)

Имя переменной: state

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
state	String	Состояние выбора

НЕАКТИВНЫЙ

Этот флаг показывает, активен ли компонент. Неактивные компоненты не отвечают на ввод пользователя и недоступны для выбора.

Имя переменной: disabled

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
disabled	Boolean	Неактивный

ОБщие события [? 239]

Общие события: отсутствуют.

7.9.9.13 Группа флаговых кнопок

Данный компонент представляет собой набор флаговых кнопок 342, объединенных в одну группу.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

ДАННЫЕ ФЛАГОВЫХ КНОПОК

Это свойство определяет перечень флаговых кнопок, отображаемых компонентом.

Имя переменной: data

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	String	**ID**. Уникальный текстовый идентификатор определенной флаговой кнопки.
value	String	**Value**. Текст ярлыка флаговой кнопки.
selected	Boolean	**Selected**. Этот флаг указывает, что флаговая кнопка активна.
Disabled	Boolean	**Disabled**. Этот флаг указывает, что флаговая кнопка неактивна.

ЗАБЛОКИРОВАТЬ ВСЕ

Этот флаг блокирует все флаговые кнопки. Блокированные флаговые кнопки недоступны для выбора.

Имя переменной: disabledAll

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
disabledAll	Boolean	Заблокировать все

ОБщие события [? 239]

Общие события: отсутствуют

7.9.9.14 Загрузчик

Данные компонент позволяет выбрать один или несколько файлов для загрузки на сервер 144.

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

ЗАГРУЗОЧНЫЕ ДАННЫЕ

Предварительно загруженный файл. Этот файл будет отображаться на компоненте как загруженный сразу после его загрузки на веб-страницу.

Имя переменной: uploadData

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
data	Data Block	Данные

ТЕКСТ КНОПКИ

Текст ярлыка кнопки Загрузить.

Имя переменной: buttonText

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
buttonText	String	Текст кнопки

СТИЛЬ ОТОБРАЖЕНИЯ

Определяет стиль отображения компонента. Данное свойство имеет несколько возможных значений:

- **Text -** загруженные файлы будут отображены как текстовые строки, включающие имя файла и расширение.
- **Picture -** загруженные файлы будут отображены как небольшие прямоугольники с текстовыми строками, включающими имя файла и расширение.
- **Picture Card -** загруженные файлы будут отображены как небольшие квадраты с предпросмотром изображений. Если загруженный файл не является изображением, отобразится текстовая строка, включающая имя файла и расширение.

Имя переменной: listType

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
listType	String	Стиль отображения

НЕАКТИВНЫЙ

Этот флаг показывает, активен ли компонент. Неактивные компоненты не отвечают на ввод пользователя и недоступны для выбора.

Имя переменной: disabledUploader

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
disabled	Boolean	Неактивный

ЛИМИТ ФАЙЛОВ

Определяет максимальное допустимое для загрузки число файлов.

Имя переменной: limitFiles

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
limitFiles	Integer	Лимит файлов

ИЗОБРАЖЕНИЕ

Изображение значка кнопки Загрузить.

Имя переменной: image

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
image	Data Block	Изображение

ОБщие события [? 239]

Общие события: отсутствуют

7.9.9.15 Зависимая кнопка

Данный компонент представляет собой набор зависимых кнопок, объединенных в одну группу. Пользователи могут выбрать только одну кнопку за раз внутри группы зависимых кнопок

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

ВЫБРАННОЕ ЗНАЧЕНИЕ

Определяет зависимую кнопку, выбираемую по умолчанию. Значение данного свойства должно совпадать с полем значения переменной данных для определенной зависимой кнопки.

Имя переменной: selectedValue

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
selectedValue	String	Выбранное значение

ДАННЫЕ КНОПКИ

Это свойство определяет перечень зависимых кнопок, отображаемых компонентом.

Имя переменной: data

Записи: 0..не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
value	String	**Value**. Определяет значение конкретной зависимой кнопки - уникальных текстовый идентификатор, позволяющий получать доступ к текущей зависимой кнопке с использованием привязок.
label	String	**Label**. Текст ярлыка зависимой кнопки.
disabled	Boolean	**Disabled**. Этот флаг определяет, отключена ли зависимая кнопка. Отключенная зависимая кнопка недоступна для выбора.

МЕТКА

Текст заголовка группы зависимых кнопок.

Имя переменной: label

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
label	String	Метка

ЗАБЛОКИРОВАТЬ ВСЕ

Этот флаг блокирует все зависимые кнопки. Блокированные зависимые кнопки недоступны для выбора.

Имя переменной: disabledAll

Записи: 1

Формат записи:

Имя поля	Тип поля	Примечания
disabledAll	Boolean	Заблокировать все

ОБщие события [?]

Общие события: отсутствуют

7.9.10 Диаграммы

В этом разделе описываются **Графики** и **Диаграммы**.

7.9.10.1 Круговая диаграмма

Enter topic text here.

7.9.10.2 Столбчатая диаграмма

Enter topic text here.

7.9.10.3 Линейная диаграмма

Enter topic text here.

7.9.11 Разное

В этом разделе описываются компоненты **Разное**.

7.9.11.1 Виджет

Компонент, позволяющий отображать виджеты 943 внутри инструментальных панелей 223 Web UI .

Общие переменные (свойства) [? 231]

Общие переменные: Имя 232, Видимый 232, Свойства расширения 232, Ширина сетки 233, Высота сетки 233, Выход за пределы контента 233, Стиль контейнера 234, Пользовательские классы 234, Стиль 234

ССЫЛКА

Путь контекста виджета 1627, соответствующего виджету для загрузки и отображения внутри инструментальной панели.

Имя переменной: reference

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
reference	String	Ссылка

КОНТЕКСТ ПО УМОЛЧАНИЮ

Выражение 112, возвращающее строку с путем контекста, который станет контекстом по умолчанию 946 для виджета.

Имя переменной: defaultContext

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
defaultContext	String	Контекст по умолчанию

Общие события [? 239]

Общие события: Информация, Видимая информация изменена, Добавлен видимый дочерний элемент, Удален видимый дочерний элемент, Нажатие мыши.

7.10 Конструктор Web UI

Конструктор Web UI AtomMind - это среда low-code разработки пользовательских интерфейсов. Конструктор помогает создавать веб-интерфейсы решений, сервисов и продуктов на базе платформы AtomMind.

Конструктор Web UI создавался с учетом одной важной концепции: он должен позволять разрабатывать веб-интерфейсы, которые с пиксельной точностью соответствуют любому дизайн-макету или эскизу. В то же время, конструктор Web UI - это инструмент low-code, поэтому даже самые сложные интерфейсы можно реализовать без единой строки кода. Клиентские интерфейсы, созданные в конструкторе UI, поддерживаются единой моделью данных 41 AtomMind .

Технически, конструктор UI редактирует инструментальные панели 912, которые становятся отдельными веб-страницами. Каждая панель имеет максимально гибкую компоновку на основе многоуровневых контейнеров 240. Каждый контейнер использует свою собственную компоновку 240 (сетку 241 или абсолютное позиционирование 241) для отрисовки компонентов 231 и вложенных контейнеров.

Большинство компонентов и контейнеров - обычные React-компоненты, упакованные в модель данных AtomMind, чтобы обеспечивать унифицированное взаимодействие с данными сервера. Широкий набор преднастроенных

компонентов, поддерживаемых комплектом разработчика и возможностью полной CSS кастомизации, гарантирует, что созданные вашими дизайнерами прототипы будут успешно реализованы как часть UI вашего веб-приложения. В стандартный комплект конструктора UI входят такие компоненты, как различные контейнеры, элементы управления вводом, компоненты отображения данных, компоненты навигации, графики/диаграммы и многое другое.

Динамическое поведение интерфейсов возможно благодаря привязкам. Привязки позволяют компонентам реагировать на события сервера и веб-страницы, а также выполнять периодические операции и инициализацию при запуске. После активации привязка выполняет некоторое преобразование данных, полученных от компонентов инструментальной панели и модели данных сервера. Полученное значение используется для корректировки свойств компонента UI или обновления каких-либо значений на сервере.

Созданные с помощью конструктора веб-интерфейсы поддерживают большинство актуальных технологий веб-приложений, такие как маршрутизация, гибкий дизайн, кэширование данных и другие.

7.10.1 Тулбар

Конструктор UI имеет отдельный тулбар в дополнение к основному тублару Web UI.

Этот тулбар расположен в левой части окна конструктора UI и дает доступ к следующим окнам:

- Редактору свойств компонентов, позволяющему просматривать и редактировать свойства избранных компонентов
- Палитре компонентов, которая служит источником новых компонентов
- Дереву компонентов, которое визуализирует иерархию вложенных контейнеров и компонентов
- **Истории редактирования свойств**

После открытия страницы конструктора UI, основной тублар Web UI также дополняется следующими элементами:

- Кнопкой Предварительный просмотр инструментальной панели
- **Выбором расширения**, используемым для редактирования гибких интерфейсов

7.10.2 Рабочая область

Рабочая область используется для просмотра и редактирования компоновки контейнера/компонента во время создания UI. Она является основной частью окна конструктора UI.

Рабочая область всегда отображает основной контейнер инструментальной панели - ее корневую панель. Если она содержит дочерние компоненты или контейнеры, в рабочей форме отображается полная иерархия. Другими словами, инструментальная панель в рабочей области очень напоминает то, как она будет выглядеть в готовом приложении.

Можно перетаскивать новые компоненты из палитры в рабочую форму, перемещать компоненты внутри текущего контейнера или в другие контейнеры, изменять размер компонентов, создавать новые привязки и так далее. Более подробно об этом см. в разделе Редактирование расположения компонентов.

Рабочая область выглядит следующим образом при редактировании инструментальной панели:

Компоненты конструктора UI выглядят почти так же, как в будущем готовом приложении, за исключением нескольких различий:

- Реальные данные на стороне сервера в большинстве случаев не отображаются
- Большая часть вводимых пользователем данных игнорируется
- Все контейнеры выделены рамками, содержащими их названия и функции быстрого редактирования
- Отображается сетка ячеек в макете сетки и вспомогательная сетка в абсолютном позиционировании.

Контекстное Меню

Клик правой кнопкой мыши на компонент внутри рабочей области вызовет контекстное меню со следующими пунктами:

- **Компоненты**. Подпункты этого пункта меню позволяют выбрать один из соответствующих компонентов (например, компонент/контейнер и любой из его контейнеров более высокого уровня)
- **Клон**. Этот пункт меню создает копию компонента.

7.10.3 Редактор свойств компонентов

Это окно содержит редактор свойств 256, используемый для изменения свойств выбранного UI компонента.

Изменения свойств сразу сохраняются и применяются к компоненту.

7.10.4 Палитра компонентов

Палитра компонентов используется для добавления компонентов на инструментальную панель путем перетаскивания на выбранный компонент внутри рабочей области 349.

Палитра отображает все доступные компоненты 231 и контейнеры 240, разделенные на несколько групп.

7.10.5 Дерево компонентов

Дерево компонентов отображает иерархию компонентов инструментальной панели. Корневая панель 241 образует корневой узел дерева. Обычные компоненты и контейнеры отображаются как узлы с разными значками. Компоненты, входящие в контейнер или область (например, вкладка панели со вкладками) отображаются как дочерние узлы.

Иерархия компонентов в дереве компонентов инструментальной панели не соответствует иерархии контекстов компонентов в модели данных 223 панели.

Дерево компонентов можно использовать для выбора компонентов или областей контейнеров с множеством областей. Когда компонент или область выбраны в окне ресурсов, они также выбираются в рабочей области 349. Их свойства отображаются в окне Свойства компонента 350.

Каждый компонент в дереве имеет следующие операции, активируемые соседними значками:

- **View/Hide**. По сути, активирует свойство Видимость компонента.
- **Rename**. Помогает переименовать компонент/контейнер/область.
- **Delete**. Позволяет удалить выбранный компонент/контейнер/область.

Переименование компонентов

Имя компонента служит для ссылки на компонент через привязки 226. Переименовать компонент можно при помощи пункта меню **Rename**.

Если у переименованного компонента были привязки, их необходимо переделать вручную, чтобы они отражали новое имя компонента.

Поиск/фильтрация компонентов

Используйте поле фильтра сверху дерева компонентов, чтобы выделить компоненты, в названиях которых присутствует определенная строка.

7.10.6 Редактирование расположения компонентов

Инструментальные панели могут включать множество вложенных контейнеров, и каждый из них имеет собственную компоновку 240. Компоновка может быть комбинированного типа, например, корневая панель с абсолютным расположением может иметь дочернюю панель с вкладками 246, чьи вкладки используют сетку, а одна из ячеек во вкладке может иметь вложенную панель 246 с абсолютной компоновкой, и так далее.

Компоновка каждого контейнера редактируется независимо от других. В то же время, можно перемещать компоненты между контейнерами.

Процесс редактирования компоновки состоит из следующих шагов:

- Добавление новых компонентов на инструментальную панель путем перетаскивания их с палитры 350 в рабочую область 349
- Перемещение компонентов из одного контейнера в другой
- Перемещение компонентов между двумя локациями внутри одного контейнера

- Изменение размеров компонентов
- Клонирование компонентов
- Редактирование ограничений компонентов (например, количество занимаемых столбцов)

Пожалуйста, внимательно ознакомьтесь с тем, как организована компоновка инструментальной панели 240, прежде чем приступить к чтению статей о редактировании сетки 351 и абсолютного 351 позиционирования.

Декорации режима редактирования

Когда расположение компонента отображается в рабочей области, его элементы могут иметь дополнительные *декорации*, которые не будут видны в готовом приложении. Эти декорации упрощают редактирование компоновки.

Следующие дополнительные визуальные элементы отображаются при включенных декорациях:

1. Пустые ячейки Сетки показаны как темно-розовые прямоугольники
2. У абсолютной компоновки имеется серо-зеленая вспомогательная сетка, помогающая размещать компоненты в пространстве
3. Каждый компонент/вложенный контейнер в компоновке Сетка имеет серый пунктирный контур, чтобы было удобнее находить отдельные и объединенные ячейки
4. Выбранный компонент/вложенный контейнер в компоновке Сетка имеет зеленый контур, чтобы показывать, свойства какого компонента редактируются в настоящий момент
5. Каждый компонент/вложенный контейнер в компоновке Сетка при наведении на него курсора мыши отображает маленькие зеленые прямоугольники. Эти треугольники позволяют менять положение компонента и вращать его внутри сетки родительского контейнера
6. У компонентов отображаются их названия, а названия контейнеров появляются при наведении на них курсора мыши
7. У некоторых контейнеров имеется значок **Add**, позволяющий создавать новую область контейнера (т.е. вкладку/шаг/слой/ и т.д.)
8. Выбранный компонент/вложенный контейнер имеет значки **Edit** и **Delete**

7.10.6.1 Редактирование сетки

Добавлять новые компоненты с палитры компонентов 350, а также перемещать и менять размер компонентов, которые уже расположены в рабочей области 349, можно простым перетаскиванием.

Компонент можно перетащить с палитры компонентов 350 (чтобы создать новый компонент выбранного типа), из рабочей области 349, а также из дерева компонентов 350. Компонент можно поместить в любую свободную ячейку рабочей области.

Когда компонент выбран в рабочей области, при наведении на него мыши появляются маленькие зеленые прямоугольники. Эти прямоугольники позволяют менять расположение компонента и вращать его внутри сетки.

7.10.6.2 Редактирование абсолютного позиционирования

Редактирование абсолютного позиционирования намного проще, чем редактирование сетки. Компоненты можно перемещать с помощью мыши по осям X и Y внутри родительского контейнера.

Чтобы изменить размер компонента, потяните за один из маленьких зеленых прямоугольников, которые появляются в его углах при наведении мыши.

Контейнеры с абсолютным позиционированием имеют вспомогательную сетку для размещения элементов. Эта сетка отображается только в режиме редактирования инструментальной панели, а во время работы с готовой панелью не видна.

7.10.7 Предварительный просмотр инструментальной панели

Кнопка Предварительный просмотр инструментальной панели, расположенная в тулбаре конструктора UI, открывает текущую версию инструментальной панели в отдельной вкладке браузера. Это позволяет протестировать инструментальную панель в так называемом режиме *preview*, очень похожем на нормальный рабочий режим (*run* mode).

Режим предпросмотра отличается от рабочего режима лишь несколькими моментами:

- **Hot Reload**. Все изменения инструментальной панели в редакторе UI сразу применяются к ее экземплярам, запущенным в режиме предпросмотра.
- **Bindings Log**. Выполнения всех привязок 226 журналируются в dev tools консоли браузера.
- **No Caching**. Кэширование 224 инструментальной панели отключено.

В режиме предпросмотра можно открыть только одну копию инструментальной панели.

7.10.8 Журналирование операций UI

Web UI журналирует свои операции и действия в консоль браузера. Эта система очень схожа с журналированием 166 на стороне сервера.

Чтобы получить доступ к журналам Web UI, откройте *dev tools* браузера и переключитесь на *console*.

Уровни записей журнала UI соответствуют уровням журналирования 171 на сервере.

Можно получить список категорий журналирования уровня UI, вызвав метод `showLoggers()` JavaScript объекта `LoggersManager`, доступного из глобальной области видимости.

Большинство категорий соответствуют категориям журналирования на стороне сервера 168. Однако, некоторые категории специфичны для Web UI:

Категория	Описание
ag.dashboard.reducers	Сообщения, относящиеся к редактированию компоновки инструментальной панели.
ag.network.requests	Сообщения, относящиеся к различным подключениям AtomMind Server.
ag.redux.actions	Сообщения, относящиеся к изменениям статуса приложения UI (например, изменениям событий или свойств компонента).

По умолчанию все логгеры настроены на журналирование сообщений на уровне **Info** или выше. Чтобы изменить уровень журналирования, вызовите метод `setLoggersLevel(string level, string logger)` JavaScript объекта `LoggersManager`.

Пример: Команда `LoggersManager.setLoggersLevel("debug", "ag.context")` переключает журналирование в категории `ag.context` на уровень `debug`.

7.11 Производительность

Инструментальные панели - активные ресурсы, которые могу значительно влиять на производительность. "Двигатель" инструментальной панели - это ее привязки 226.

Запущенная в браузере инструментальная панель может вызвать значительную нагрузку на процессор и использование большого объема памяти самого браузера, AtomMind Server, или и того, и другого.

Нагрузка на процессор, создаваемая инструментальной панелью, растет многократно из-за следующих факторов:

- Количество экземпляров инструментальной панели, запущенных оператором.
- Количество привязок инструментальной панели.
- Частота обработки привязок инструментальной панели. Частота может быть прямо определена в опциях привязки (для Периодических привязок), либо неявно определена частотой событий и изменений статуса, которые запускают выполнение привязки (для привязок При событии). Более подробно см. раздел Производительность привязок 742.
- Сложность и воздействие выражений привязок. Более подробно см. раздел Производительность выражений 141.

- Влияние записи целей привязок. В большинстве случаев запись цели привязки - это запись переменной контекста или вызов функции контекста. Более подробно см. Производительность переменных 70 и Производительность функций 73.
- Использование и параметры механизма кэширования инструментальных панелей 224.

Запуск инструментальных панелей с сотнями сложных динамических компонентов (таких как таблицы 282, журналы событий 303 или карты 265) может вызвать значительную постоянную загрузку памяти браузера.

Запуск инструментальных панелей может также привести к кратковременному резкому увеличению используемой памяти, если привязки и графики инструментальной панели загружают большие объемы данных (например, множество исторических событий 75).

Оптимизация запуска инструментальной панели

Современные веб-приложения предполагают очень короткое время загрузки страницы. Чтобы максимально сократить время отрисовки веб-инструментальной панели, рекомендуется минимизировать количество привязок при запуске.

В режиме запуска следует выполнять только привязки, отображающие информацию, которую пользователь должен увидеть в первые секунды. Все остальные привязки инициализации инструментальной панели должны запускаться при событии показ 244 корневой панели, которое, по сути, возникает сразу после отображения страницы в окне браузера.

Такая настройка производительности особенно важна для привязок, которые отображают информацию, изначально скрытую от глаз пользователя, например, содержание нестандартных вкладок панели со вкладками 246.

7.12 Настройки

Плагин 207 Web UI использует следующие настройки глобальной конфигурации.

Инструментальные панели

Настройки в этой группе конфигурируют поведение инструментальной панели.

- **Enable Caching**. Включает/выключает кэширование 224 для всех веб инструментальных панелей. Конфигурация кэша, специфичного для инструментальной панели 914, может перекрывать эту опцию и отключать кэширование для определенных инструментальных панелей.
- **Cache Sizing Method**. Позволяет определять, ограничен ли максимальный размер кэша настройками **Dashboard Instance Count** или **Memory Footprint**.
- **Dashboard Instance Count**. Максимальное количество экземпляров 223 инструментальной панели в кэше.
- **Memory Footprint**. Максимальное пространство "кучи", отведенное экземплярам инструментальной панели.

Настройки Recaptcha

Настройки в этой группе конфигурируют использование reCAPTCHA.

- **Enable reCAPTCHA**. Дополняет страницу входа аутентификацией reCAPTCHA v3.
- **reCAPTCHA Server Key**. Ключ сервера reCAPTCHA.
- **reCAPTCHA Secret Key**. Секретный ключ reCAPTCHA
- **Number of reCAPTCHA Attempts**. Максимальное количество попыток авторизации.

Кастомизация

Настройки в этой группе конфигурируют вкладку браузера Web UI.

- **Свой заголовок**. Позволяет задать свой префикс вкладки браузера.
- **Своя иконка**. Позволяет задать свою иконку сайта.

7.13 Встроенный веб сервер

AtomMind Server включает *встроенный веб сервер,* который дает доступ через браузер администраторам AtomMind и пользователям приложений/продуктов/решений на базе платформы, а также доступ по API через интернет для сторонних систем.

На встроенном веб сервере расположены несколько основных веб сервисов:

- Web UI 220 используется для администрирования системы, разработки приложений на базе платформы и для ежедневной работы с этими приложениями
- HTTP сервер 1414 предлагает сторонним приложениям получить доступ и оперировать моделью данных 41 AtomMind Server через настраиваемый REST API
- Веб контрольный центр 354 выступает как очень простой веб-портал верхнего уровня для всех доступных веб сервисов

7.13.1 Центр управления Web

Центр управления веб AtomMind Server - это веб-портал, используемый для получения доступа к различным веб-интерфейсам AtomMind Server. Центр управления и другие веб-приложения, размещенные на AtomMind Server, опираются на встроенный веб сервер 354, который запускается автоматически при старте AtomMind Server.

Для получения доступа к центру управления, введите следующий URL в браузере: `https://hostname:8443/web/cc`, где `hostname` - имя хоста или IP-адрес машины AtomMind Server.

команды центра управления

В открытом центре управления можно выбрать одну из следующих опций:

- **Web UI**. Дает доступ к Web UI 354.
- **Веб-проигрыватель виджетов**. Открывает один виджет 943 внутри определенной веб-страницы. Требует отдельного логина, указания контекстного пути виджета и пути контекста по умолчанию 946 виджета.
- **Скачать автономный AtomMind Client**. Перенаправляет на вебсайт ТВЭЛ для скачивания установочного комплекта AtomMind Client 359.

7.13.2 Настройка веб сервера

Перечисленные ниже опции управляют поведением *Встроенного веб сервера* и веб приложений, работающих на нем.

Включить встроенный веб сервер

Имя переменной в конфигурационном файле: webServerEnabled

Тип значения: Boolean

Возможные значения: true или false

Значение по умолчанию: true

Определяет, включен ли интегрированный с AtomMind Server'ом веб сервер. Если нет, то все веб-услуги (Web UI 220, Web Services 1417 и пр.) недоступны.

Включить веб-сервис (Web Service)

Имя переменной в конфигурационном файле: webServiceEnabled

Тип значения: Boolean

Возможные значения: true или false

Значение по умолчанию: true

Определяет, доступен ли Web Service 1417 AtomMind Server для других приложений.

Включить незащищенный доступ к веб-приложениям

Имя переменной в конфигурационном файле: webNonSecureAccessEnabled

Тип значения: Boolean

Возможные значения: true или false

Значение по умолчанию: true

Определяет, можно ли получить доступ к *веб-приложениям* AtomMind Server, используя небезопасный протокол HTTP. Обычно эту опцию следует отключать, чтобы доступ осуществлялся только по защищенному HTTPS протоколу (с шифрованием на основе SSL).

Включить клиентский апплет и Java Web Start

Имя переменной в конфигурационном файле: webAppletAndJavaWSEnabled

Тип значения: Boolean

Возможные значения: true или false

Значение по умолчанию: false

Определяет, доступны ли клиентский апплет и Java Web Start для запуска.

Тип соединения

Имя переменной в конфигурационном файле: webConnectionType

Тип значения: Integer

Возможные значения: Не разрешать удаленные соединения, Любые удаленные соединения и Только заранее заданные соединения

Значение по умолчанию: Не разрешать удаленные соединения

Опция включает и контролирует работу в режиме Автономный веб сервер 357.

Заранее заданные соединения

Имя переменной в конфигурационном файле: webConnectionPreconfigured

Тип значения: Data Table

Эта таблица определяет список заданных соединений, которые может устанавливать Автономный веб сервер 357 с внешними серверами AtomMind. Таблица используется только если **Тип соединения** установлен на **Только заранее заданные соединения**.

Список алиасов имен хоста сервера, разделенных запятым

Имя переменной в конфигурационном файле: webAppsAliases

Тип значения: String

Возможные значения: Одно ли более имен хостов сети, разделенных запятой

Значение по умолчанию: "" (пустое)

Определяет имя (имена) хоста, по которым можно получить доступ к *веб-приложениям AtomMind Server (*Web Admin, Web Service, HTTP Proxy 209 и пр.). Эта опция будет работать, если означенные имена хоста правильно установлены в DNS. Опцию следует настроить дополнительно во время конфигурации вашего сервера DNS.

Номер порта для приема защищенных соединений (HTTPS)

Имя переменной в конфигурационном файле: webAppsSslPort

Тип значения: Integer

Возможные значения: 1-65535

Значение по умолчанию: 8443

Определяет номер порта, по которому будет доступно приложение *Web Admin*. Вы можете поменять значение этой опции на 443, если нет других веб серверов, прослушивающих этот порт. 443 - это номер порта HTTPS по умолчанию. Не рекомендуется использовать *80* в качестве значения для этой опции, потому что 80 - это номер порта по умолчанию для небезопасного протокола HTTP.

Номер порта для приема незащищенных соединений (HTTP)

Имя переменной в конфигурационном файле: webAppsNonSslPort

Тип значения: Integer

Возможные значения: 1-65535

Значение по умолчанию: 8080

Определяет номер порта, по которому можно получить доступ к *веб-приложениям AtomMind Server*. Вы можете выставить значение этой опции в 80, если нет других веб серверов, прослушивающих этот порт. *80* - это номер порта HTTP по умолчанию.

Размер буфера чтения

Key name in the configuration file: appReadBufferSize

Тип значения: Long

Возможные значения: 1 или более

Значение по умолчанию: 8192

Каждое соединение, которое открывается веб сервером, связано с байтовым буфером чтения. Этот атрибут управляет размером данного буфера. По умолчанию буфер чтения имеет размер 8192 байта. Для снижения параллелизма вы можете увеличить размер, чтобы буферизировать больше данных. Для большого количества поддерживаемых соединений уменьшите это число или увеличьте размер кучи.

Размер буфера записи

Имя переменной в конфигурационном файле: appWriteBufferSize

Тип значения: Long

Возможные значения: 1 или более

Значение по умолчанию: 8192

Каждое соединение, которое открывается веб сервером, связано с байтовым буфером записи. Этот атрибут управляет размером данного буфера. По умолчанию буфер записи имеет размер 8192 байта. Для снижения параллелизма вы можете увеличить размер, чтобы буферизировать больше данных ответа. Для большого количества поддерживаемых соединений уменьшите этот параметр или увеличьте размер кучи. Значение по умолчанию здесь довольно низкое, вам следует его использовать, если вы не имеете дело с десятками тысяч одновременных подключений.

Максимальное количество потоков

Имя переменной в конфигурационном файле: maxThreads

Тип значения: Integer

Возможные значения: 1 или более

Значение по умолчанию: 200

Ограничивает число потоков, которые могут использоваться встроенным веб-сервером.

Путь к хранилищу ключей

Имя переменной в конфигурационном файле: webAppsKeyStoreFile

Тип значения: String

Возможные значения: Любой действительный путь к папке

Значение по умолчанию: Отсутствует

Определяет файл, содержащий SSL сертификаты, которые веб сервер будет пытаться использовать.

Пароль хранилища ключей

Имя переменной в конфигурационном файле: webAppsKeyStorePassword

Тип значения: String

Возможные значения: Любая строка

Значение по умолчанию: Отсутствует

Определяет пароль для открытия файла хранилища ключей.

Пароль ключа

Имя переменной в конфигурационном файле: webAppsKeyPassword

Тип значения: String

Возможные значения: Любая строка

Значение по умолчаниюvalue: Отсутствует

Определяет пароль для открытия ключа SSL.

Режим выполнения виджетов

Имя переменной в конфигурационном файле: webAppsWidgetsMode

Тип значения: Integer

Возможные значения: 0 (виджеты будут запускаться на машине сервера) или 1 (виджеты будут запускаться на машине клиента).

Значение по умолчанию: 0

Определяет, где будут выполняться виджеты при работе с Web UI 220 - на сервере или на устройстве-клиенте .

7.13.3 Автономный веб сервер

Некоторые развернутые системы AtomMind включают сотни серверов, работающих на периферийных устройствах с ограниченными техническими ресурсами, например, на одноплатных ПК, ПЛК на Linux, сенсорных панелях и др.

Таким устройствам часто не хватает оперативной памяти и мощности процессора для работы встроенного веб сервера AtomMind Server. В то же время, удаленный доступ к подобным миниатюрным устройствам очень важен.

Эту проблему решает развертывание **Автономного AtomMind веб сервера**, который будет играть роль "прокси-сервера", т.е. посредника для пользовательских соединений с другими серверами AtomMind, используя протокол AtomMind.

Технически автономный AtomMind веб сервер - это специальный выпуск AtomMind Server без необязательных модулей, а только со встроенным модулем веб-сервера. Этот модуль настроен для "автономного" использования.

Для установки и настройки Автономного AtomMind веб сервера выполните следующие шаги:

- Скачайте **Автономный AtomMind веб сервера** с вебсайта АО "ТВЭЛ" и установите его, следуя стандартным процедурам установки 152 AtomMind Server
- Запустите сервер и подключитесь к нему через десктопный или веб-клиент
- Откройте общие настройки 207 плагина Веб сервер и поменяйте **Тип соединения** на:
  - **Любые удаленные соединения**, чтобы включить переадресацию входящих соединений на любой внешний AtomMind Server, чей адрес/порт указан в окне входа Автономного AtomMind веб сервера
  - **Только заранее заданные соединения**, чтобы включить возможность выбора определенного AtomMind Server в окне входа Автономного AtomMind веб сервера
- Во втором случае, заполните таблицу **Заранее заданные соединения** описаниями, адресами и портами AtomMind серверов, куда могут переадресовываться соединения
- Сохраните настройки и, если необходимо, перезапустите сервер.

7.13.4 Установка SSL сертификата для веб-приложений

Эта статья объясняет, как установить доверенный сертификат для веб-приложений AtomMind Server для обеспечения безопасности соединения с ними без предупреждений браузера.

1. Сначала, вам необходимо приобрести подписанный сертификат ЦС (`CA.crt`) на основе сгенерированного вами секретного ключа (`private.key`).

2. Следующий шаг - это установка инструмента OpenSSL. Если вы используете OpenSSL на Windows, вам может понадобиться задать путь к файлу конфигурации, запустив комманду консоли:

```
set OPENSSL_CONF=c:\OpenSSL-Win32\bin\openssl.cfg
```

3. Используйте OpenSSL для создания хранилища PKCS #12:

```
openssl pkcs12 -export -in CA.crt -inkey private.key -out server.p12
```

где

- `CA.crt` – сертификат, который вы получили от ЦС

- `private.key` – секретный ключ для CA.crt
- `server.p12` – имя файла в хранилище PKCS #12

4. Затем, вы можете импортировать ваше хранилище ключей на основе Java PKCS #12, используя утилиту **keytool**, входящую в состав набора для разработки Java-приложений (Java Development Kit; JDK). Keytool находится в закладке `%JDK%\bin`:

```
keytool -importkeystore -srckeystore server.p12 -destkeystore server.jks -srcstoretype pkcs12
```

где

- `server.p12` – имя файла в хранилище ключей PKCS #12
- `server.jks` – имя хранилища ключей Java

5. Поместите ваше хранилище ключей в папку `%AtomMind%\admin` и задайте следующие опции в разделе Веб-приложения 354 общей конфигурации сервера:

- Путь к файлу хранилища ключей – путь к хранилищу ключей Java (по умолчанию папка AtomMind\admin)
- Пароль для хранилища ключей – пароль для хранилища ключей Java
- Пароль для ключа – пароль для секретного ключа (если это поле не заполнено, AtomMind пытается использовать пароль хранилища ключей для чтения секретного ключа из хранилища ключей)

8 Десктопное клиентское приложение

AtomMind Client - это десктопное ПО с графическим пользовательским интерфейсом, которое служит для взаимодействия с AtomMind Server.

Ранее AtomMind Client использовался как интерфейс пользователя AtomMind по умолчанию, однако сейчас в основном используется web UI 354.

Данный продукт можно использовать для:

1. Конфигурации и администрирования сервера
2. Управления и настройки устройств
3. Мониторинга событий и получения уведомлений об ошибках
4. Разработки и эксплуатации вашего продукта/решения, использующего десктопный интерфейс пользователя

Он может использоваться практически для всего, что связано с AtomMind Server. AtomMind Client - это "окно пользователя для работы" в AtomMind Server. Здесь пользователь проводит большую часть времени.

AtomMind Client использует коммуникационный протокол AtomMind 2119 для взаимодействия с AtomMind Server.

8.1 Единая консоль оператора

AtomMind Client может быть установлен как автономное кроссплатформенное приложение для Windows, Linux/Unix и Mac OS. В этом случае AtomMind Client действует как *Единая Консоль Оператора*, поскольку может устанавливать множество одновременных соединений с сервером и объединять данные, собранные распределенной инсталляцией AtomMind.

Когда он запущен как *Единая Консоль Оператора*, AtomMind Client позволяет создавать рабочее пространство 363, которое включает в себя соединения с сервером 363. Таким образом, каждый оператор, использующий AtomMind Client на определенной машине, сначала входит в свое рабочее пространство, и рабочее пространство автоматически подключает его к одному или более серверам.

8.2 Технические требования

AtomMind Client - это Java-приложение, которое работает на любой ОС (операционной системе) с Java.

Системные требования

- **Операционная система**: любая ОС, поддерживающая Java, включая встроенные среды (например, ARM Linux)
- **RAM:** минимально 256 MB
- **HDD:** 200 MB свободного пространства
- **Рекомендуемое разрешение экрана**: 1280x1024 или выше

При работе со сложными инструментальными панелями 912/виджетами 943 или с большими таблицами данных 49 в AtomMind Client может потребоваться увеличение размера памяти виртуальной машины Java (параметр `-Xmx`). В некоторых случаях необходимо увеличить этот параметр в несколько раз по сравнению со значением по умолчанию. Более подробно об этом см. раздел Настройка параметров Java VM 363.

8.3 Инсталляция

Инсталлятор AtomMind Client объединен в пакет с Виртуальной Машиной Java 7, поэтому на Вашем компьютере необходимо иметь установленную Java.

Инсталляция необходима только для работы AtomMind Client в режиме единая консоль оператора 359.

ТВЭЛ предлагает входящие в пакет файлы установки для следующих операционных систем:

- *Windows 98, NT, 2000, XP, 2003 Server, Vista, 2008 Server, 7 и 2012 Server* (совместимы со встраиваемым Windows)
- Различные версии *Linux* (совместимы со встраиваемым Linux)
- *Mac OS X*

Инсталляторы для других операционных систем на основе Java (FreeBSD и другие *nix, Solaris и т.д.) доступны по запросу.

Для установки продукта просто запустите выполняемый файл инсталлятора и следуйте инструкциям. На некоторых системах вам необходимо сделать пакет инсталляции 'выполняемым' до его запуска.

8.4 Деинсталляция

AtomMind Client деинсталлируется при запуске исполняемого файла `uninstall` из директории установки AtomMind Client. Перед деинсталляцией следует закрыть приложение.

Деинсталлятор AtomMind Client не удаляет файлы, созданные или измененные AtomMind Client во время работы (рабочие пространства, логи и пр.). Чтобы полностью стереть все данные AtomMind Client, следует удалить установочную директорию со всем ее содержимым, когда деинсталлятор закончит работу. Это может быть необходимо для выполнения "чистой" переустановки.

8.5 Запуск и остановка

Эта статья объясняет, как запустить/остановить AtomMind Client, установленный в режиме единая консоль оператора 359.

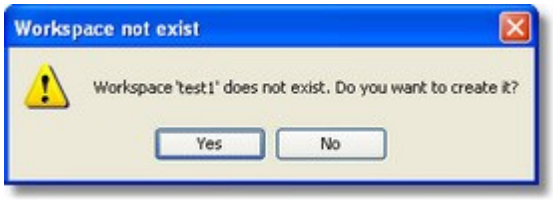
AtomMind Client запускается специальным *Загрузчиком,* созданным во время установки. Тип загрузчика зависит от версии ОС: загрузчик в *Unix* - это сценарий `am_client`, в то время как в *Windows* это маленький исполняемый файл, доступный из меню *Пуск* (*Пуск > Программы > AtomMind > Клиент) или через иконку на рабочем столе*:

Вход в рабочее пространство

После загрузки AtomMind Client следует зарегистрироваться в рабочем пространстве 363. Выберите рабочее пространство и введите пароль в диалоговом окне:

Если Вы используете AtomMind Client в первый раз, просто введите **любое** имя и пароль, чтобы создать новое рабочее пространство.

Если рабочее пространство с данным именем не существует, система предложит создать новое имя/пространство:



Прямая авторизация

Вы можете авторизоваться с удаленного AtomMind Client на любом AtomMind Server через графический интерфейс пользователя.

Используя параметр `-r`, можно ввести адрес сервера, порт, логин и пароль.

Можно также авторизоваться в Простом режиме 362, используя параметры `-r` и `-s`.

Используя параметры `-address` и `-port`, можно указать адрес сервера и порт, чтобы пользователь не мог поменять эти настройки при авторизации.

8.5.1 Параметры командной строки

AtomMind Client можно запускать с любыми из следующих указанных параметров командной строки:

Параметр	Описание
`-a`	Активирует режим администратора 362.
`-c`	Автоматически создает рабочее пространство 363, если его нет.
`-l <context> <action>`	Автоматически запускает действие `<action>` из контекста `<context>`.
	Эта опция позволяет открывать необходимую информацию при запуске AtomMind Client. Например, если нажать на устройство в сторонней системе (например, системе биллинга), запустится AtomMind Client, и у оператора сразу же отобразится инструментальная панель выбранного устройства.
`-u <username>`	Присваивает имя рабочему пространству 363 для загрузки во время начала работы.
	Если выбраны обе опции `-u` и `-p`, диалоговое окно авторизации не выводится на экран, а рабочее пространство, выбранное опцией `-u`, загружается автоматически. Эту комбинацию следует использовать лишь в безопасном окружении, поскольку она предоставляет мгновенный доступ к Вашей инсталляции AtomMind Server.
	Вы можете использовать опции `-u` и `-p` для запуска AtomMind Client и незамедлительной демонстрации нескольких виджетов 943 (используя Автозапуск 1468), отслеживая информацию для контекстов (таких как, например, станция мониторинга на предприятии). Т.е. AtomMind Client может быть запущена с иконки на рабочем столе (или автоматически во время запуска компьютера), она показывает различные датчики и заголовки на экране, что позволяет определить статус той или иной машины.
`-p <password>`	Пароль рабочего пространства.
`-s`	Запускает AtomMind Client в простом режиме 362.
`-t`	Запускает AtomMind Client в стационарном режиме 362.
`-scr <number>`	Запускает AtomMind Client на специальном экране. Нумерация начиная с нуля.
`-x <number>`	X-координата окна приложения на экране.
`-y <number>`	Y-координата окна приложения на экране.
`-w <number>`	Ширина окна приложения.
`-h <number>`	Высота окна приложения.
`-ac <number>`	Период автоподключения сервера: 0 - отключить автоподключения, 1 - автоподключение только при начале работы, >1000 - период попыток автоподключения с миллисекундах

8.5.2 Простой режим

Операторы AtomMind часто не нуждаются ни в каких стандартных компонентах AtomMind Client (таких как системное дерево 370 и журнал событий 398) для осуществления ежедневных обязанностей. Они используют лишь специально разработанные виджеты 943, сгруппированные в инструментальные панели 912, для осуществления задач мониторинга и контроля, таких как управление сегментом сети производственного процесса.

Такие операторы могут запускать AtomMind Client в так называемом *Простом режиме*. В этом режиме окна не открываются автоматически при запуске. Однако устанавливаются соединения со всеми серверами, зарегистрированными в рабочем пространстве 363. При соединении с сервером AtomMind Client запускает все автоматические действия 941 из сервера в нормальном режиме. Эти действия должны открывать необходимые инструментальные панели, обеспечивающие доступ к ежедневным операциям.

Простой Режим активируется переключателем командной строки 164 `-s`.

8.5.3 Стационарный режим

Операторы AtomMind часто работают в среде, которая запрещает им доступ к операционной системе их рабочей станции, конфигурации рабочего пространства 363 AtomMind Client и конфигурации AtomMind Server. Они выполняют ежедневные обязанности, такие как мониторинг сети или контроль производственного процесса, используя только специально разработанные виджеты 943, сгруппированные в инструментальные панели 912.

Такие операторы могут входить в операционную систему и в AtomMind Client в так называемом *Стационарном режиме*. В этом режиме разрешено только приложение AtomMind Client, большинство функциональных клавиш отключены, а окна не открываются автоматически при запуске. Редактирование рабочего пространства 363 запрещено. При подключении к серверу AtomMind Client запускает действия автозапуска 941 для открытия необходимых инструментальных панелей, составляющих операторский интерфейс.

В операционных системах семейства Windows стационарный режим активируется при помощи:

- Отключения Диспетчера задач Windows и замена среды Windows выполняемым AtomMind Clientom
- Определения переключателей командной строки 361 `-t` (стационарный режим) и `-s` (простой режим)

Для активации стационарного режима для определенного пользователя Windows войдите в систему от имени этого пользователя и импортируйте в регистр следующий файл:

```
Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Winlogon]
"Shell"="C:\\Program Files\\AtomMind\\am_client.exe -t -s"

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\System]
"DisableTaskMgr"=dword:00000001
```

8.5.4 Режим администратора

AtomMind Client может работать в двух режимах:

- Обычный режим
- Режим администратора

Режим администратора включается, если выбрать опцию командной строки 361 `-a` . Она наделяет администраторов AtomMind расширенными возможностями. Режим Администратора не требует особых привилегий, однако расширенные возможности недоступны для пользователя, у которого нет права доступа к серверу.

В настоящий момент режим расширенных возможностей дает доступ к следующим функциям:

- Интерактивный гид-редактор ("Macro recorder). Он доступен через **Главное меню** 365 > **Вид** > **Запустить макрорегистратор**. Для его запуска не требуется разрешения сервера, но любые внесенные изменения будут применяться лишь к локальной установке AtomMind Client. На работу сервера это не повлияет.
- Действия **Открыть журнал сервера** и **Открыть браузер устройств** в узле AtomMind Server'а 376 Системного Дерева.

8.5.5 Настройка параметров Java VM

Язык интерфейса пользователя, использование памяти и некоторые другие параметры AtomMind Client можно сконфигурировать при помощи Файлов Свойств Загрузчика (`*.vmoptions`), чей формат такой же, как Файл Свойств Загрузчика AtomMind Server 162.

8.6 Рабочие пространства и соединения с сервером

Существуют два термина, которые можно легко перепутать: *Рабочее пространство* и *Соединение с сервером*. Оба эти термина часто употребляются в данном руководстве.

Рабочие пространства используются только если AtomMind Client работает в режими единая консоль оператора 359. В этом режиме соединения с сервером также управляются вручную.

Во всех других режимах у AtomMind Client нет рабочих пространств, и соединение идет с сервером, откуда был запущен AtomMind Client.

Рабочее пространство - это специальная директория в AtomMind Client, в которой хранятся данные, используемые одним пользователем AtomMind Client. Обратите внимание, что ***пользователь*** *AtomMind Client* - это человек, который использует AtomMind Client, чтобы управлять одним или несколькими AtomMind Server. Это не то же самое, что Учетная запись пользователя 478 на AtomMind Server. Хотя у каждого пользователя AtomMind Client есть учетная запись в AtomMind Server, важно различать эти термины. Пользователь AtomMind Client может иметь больше одной учетной записи в AtomMind Servere, и теоретически, может запускать AtomMind Client без соответствующей пользовательской учетной записи в AtomMind Server (для тестирования и пр.). AtomMind Client при этом будет работать, несмотря на снижение производительности.

Рабочее пространство:

- Содержит от нуля или более подключений сервера,
- Содержит "оконную" компоновку (размер "окна", расположение, закрепленный/плавающий статус и пр.),
- Находится в установочной папке AtomMind Client,
- Зашифровано (*пароль рабочего пространства раз*блокирует *его*),
- Никогда не переносится на сервер полностью или частично.

Серверное соединение:

- Хранится на рабочем пространстве (зашифровано для обеспечения безопасности),
- Включает адрес сервера, порт, имя пользовательского аккаунта на сервере и пароль для этого аккаунта,
- Это всего лишь локальное определение для конфигурации - AtomMind Client *попытается* установить соединение с AtomMind Server, используя эти регистрационные данные. Если данные правильные (сервер находится там, где предполагается, а аккаунту соответствуют регистрационные данные), соединение будет установлено.

Рабочие пространства

Рабочие пространства хранятся в папке `/workspaces` в директории AtomMind Client, то есть, `/John/.ag//workspaces/test/*.*` содержит "**тестовое**" рабочее пространство. Рабочие пространства можно перемещать между инсталляциями AtomMind Client, но рабочие пространства, сохраненные в более старых версиях AtomMind Client могут неправильно загрузиться на более новых версиях.

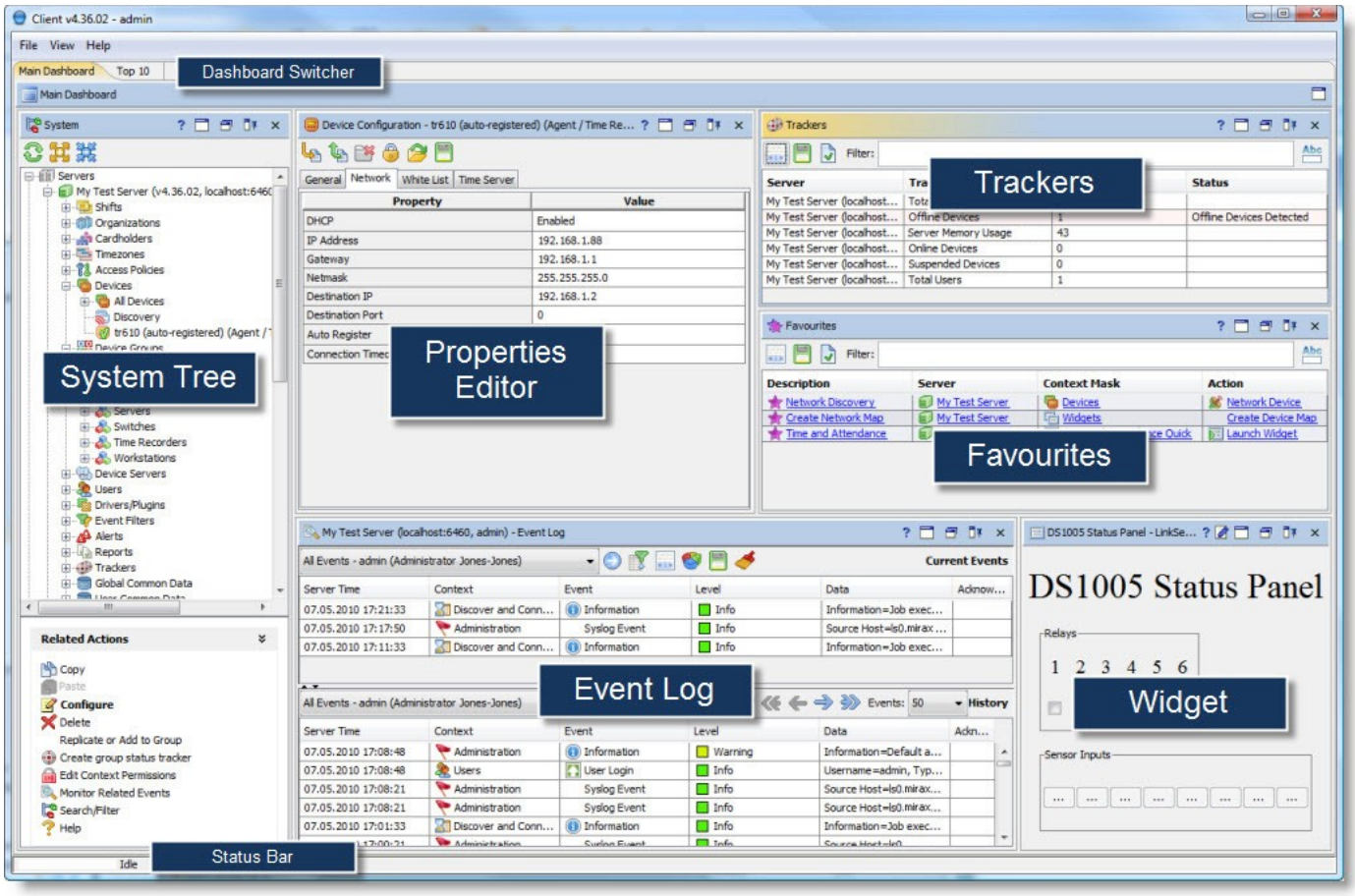
8.7 Главное окно

Основная форма AtomMind Client состоит из Главного меню 365, *Инструментальных панелей* с различными окнами и Строки текущего состояния. По умолчанию имеется лишь одно окно инструментальных панелей, именуемое **Основная Инструментальная Панель**.

Раскладка основной инструментальной панели по умолчанию включает следующие окна:

- Системное дерево 370
- Один или несколько Журналов регистрации событий 398
- Окно Избранное 414
- Окно Датчики 414

Другие окна, такие как Редактор свойств 377, Виджет 943 или Просмотрщик отчетов 415 можно добавить в процессе работы.

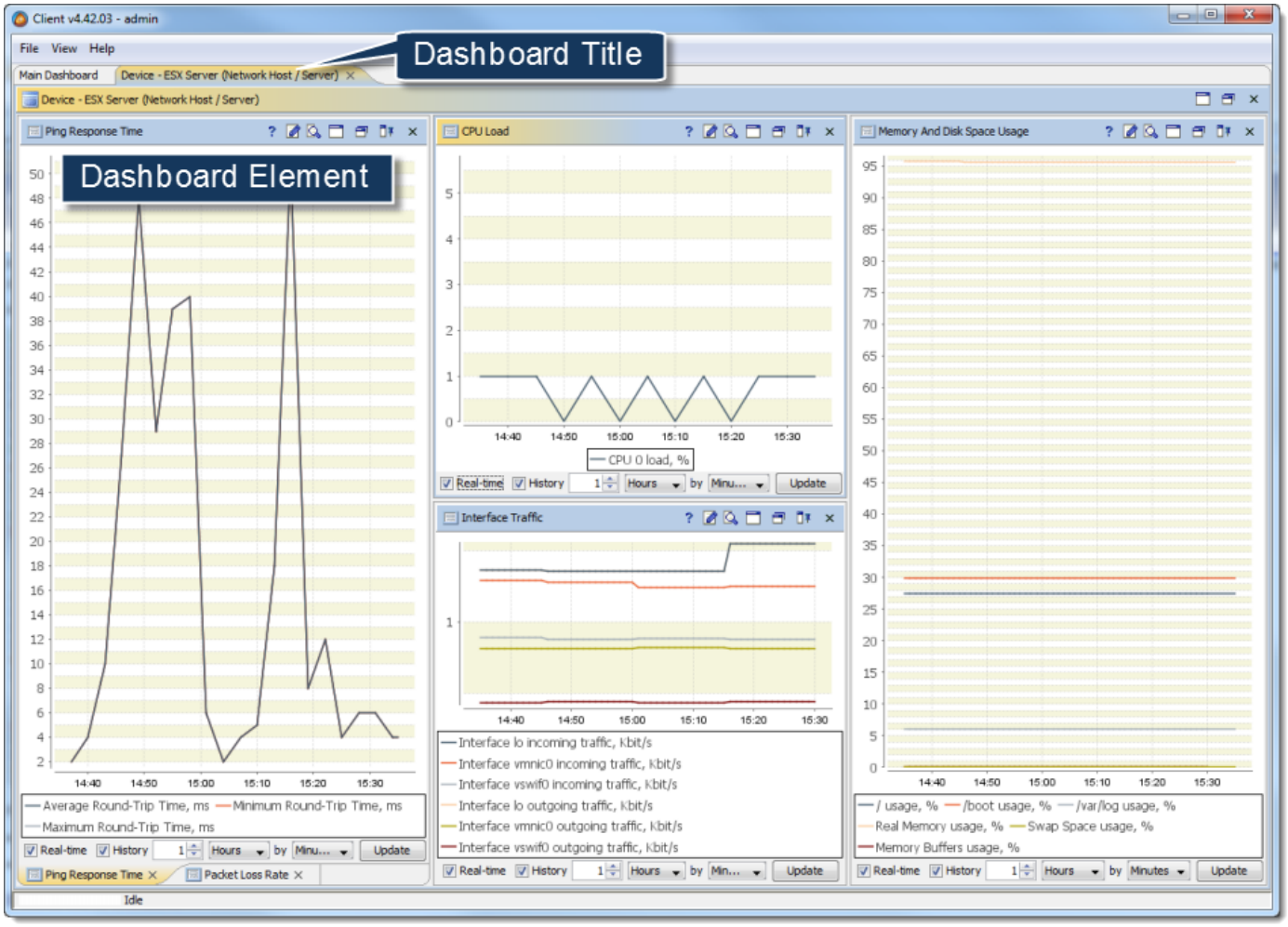


Системное дерево, Редактор свойств, Датчики, Журнал регистрации события и другие окна являются плавающими 366.

окна инструментальных панелей

Каждая инструментальная панель 912 открывается в отдельном окне с вкладками в любом пользовательском интерфейсе AtomMind Server. После закрытия окна инструментальной панели все ее компоненты останавливаются (например, если были запущены виджеты 943) и скрываются.

Так выглядит инструментальная панель в AtomMind Client:



Каждый элемент инструментальной панели можно настроить, нажав на иконку Configure в заголовке элемента.

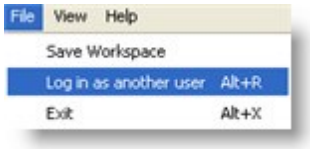
Если какой-либо элемент закрывается путем нажатия на иконку Close, пользователь получает подсказку окончательно удалить соответствующий элемент из конфигурации инструментальной панели.

Тулбар панели инструментов включает следующие элементы:

- **Save layout** сохраняет расположения элементов в конфигурации панели инструментов на сервере
- **Load layout** загружает расположения элементов из конфигурации панели инструментов на сервере
- **Show palette** показывает список компонентов, которые можно добавить на панель с прокруткой и настроить на ходу
- **Reset layout** откатывает расположения элементов до расположений по умолчанию

8.7.1 Главное меню

В Главном Меню AtomMind Client есть три пункта: Файл, Вид и Справка.



Меню Файл

- **Сохранить рабочее пространство**. Сохраняет текущее рабочее пространство незамедлительно, без ожидания выключения.
- **Сменить рабочее пространство**. Сохраняет и закрывает текущее рабочее пространство 363, позволяет загрузить еще одно. **(Alt+R)**
- **Выход**. Сохранить текущее рабочее пространство и выйти из AtomMind Client. **(Alt+X)**

Меню Вид

- **Сбросить компоновку окон**. Установить размеры и положения всех окон и компонентов по умолчанию.
- **Системное дерево**. Показать/скрыть Системное дерево 370. Когда Системное дерево показано, этот элемент отмечен "галочкой".
- **Избранное**. Показать/скрыть Избранное 414. Когда Избранное показано, этот элемент отмечен "галочкой".
- **Датчики**. Показать/скрыть Датчики 414. Когда Датчики показаны, этот элемент отмечен "галочкой".
- **Тревоги**. Отключите этот флажок в меню для блокировки всплывающих и звуковых уведомлений о тревоге 791. Этот флажок всегда включен по умолчанию, его состояние не сохраняется в рабочем пространстве.
- **Автозапуск**. Отключите этот флажок для остановки выполнения действия автозапуск 941 для всех учетных записей серверов. Действия автозапуска не запустятся, когда это рабочее пространство откроется в следующий раз. Этот флажок всегда включен по умолчанию, его состояние сохраняется в рабочем пространстве.
- **Показать все журналы событий**. Показывает Журнал регистрации событий для каждого активного соединения сервера.
- **Скрыть все журналы событий**. Скрывает все журналы регистрации событий.
- **Использовать модальные диалоги**. Если функция включена, большая часть диалогов редактора свойств 377 открыта в отдельных модальных диалоговых окнах. Такое поведение по умолчанию может быть неудобным, если необходимо копировать-вставить данные в окнах или проверять данные в одном окне, а вносить изменения в другом. Отключение этой функции открывает большую часть редакторов свойств в обычных плавающих окнах. Если необходимо, нажать на кнопку Сохранить на панели инструментов редактора свойств, чтобы сохранить изменения.

Меню справка

- **Открыть справку**. Открывает руководство AtomMind в окне браузера (HTML).
- **Открыть интерактивный самоучитель**. Запускает Интерактивный проводник 476.
- **О программе**. Содержит информацию о версии программного обеспечения и авторских правах.

8.7.2 Строка текущего состояния

Строка состояния содержит анимированный индикатор выполнения с описанием. Строка показывает текущее состояние AtomMind Client.

- **Ожидание**. Показывает, что ни одна команда не выполняется, и сетевой ввод/вывод отсутствует.
- **Загрузка**. Указывает, что AtomMind Client в настоящий момент выполняет операции с одним или несколькими серверами, и производится передача данных.

Некоторые операции могут уведомлять о работе или текущем статусе, отправляя сообщения с строку текущего состояния. Например, Автообнаружение внешних 2096 Device Server уведомляет о статусе следующим образом:

```
- Auto-Discovery started
- Found admin.c1 at 0.1.2.3.4.5 (192.168.1.100)
- Found admin.c2 at 0.1.2.3.4.6 (192.168.1.101)
...
```

8.7.3 Управление плавающими окнами

У каждого окна AtomMind Client есть небольшая строка, содержащая информацию об окне, она может быть "плавающей" или фиксированной. У строки есть несколько кнопок:

Справка по данному окну. Открывает документацию для компонентов, расположенных в плавающем окне.

Развернуть окно. Окно будет **занимать** весь фрейм AtomMind Client.

Восстановить размер окна. Изменяет увеличенный размер окна на обычный.

Включить плавающую панель. Разрешает окну располагаться поверх фрейма AtomMind Client.

Отключить плавающую панель. Возвращает плавающее окно в его предыдущее расположение в рамке AtomMind Client.

Включить/отключить режим боковой панели. Когда окно располагается на боковой панели, оно уменьшается до размера иконки с одной из сторон фрейма AtomMind Client. Наведение курсора на эту иконку отобразит окно.

Скрыть активное окно боковой панели. Скрывает видимое окно боковой панели в боковой панели.

Закрыть окно. Закрывает плавающее окно.

Для некоторых окон некоторые из этих операций могут быть недоступны.

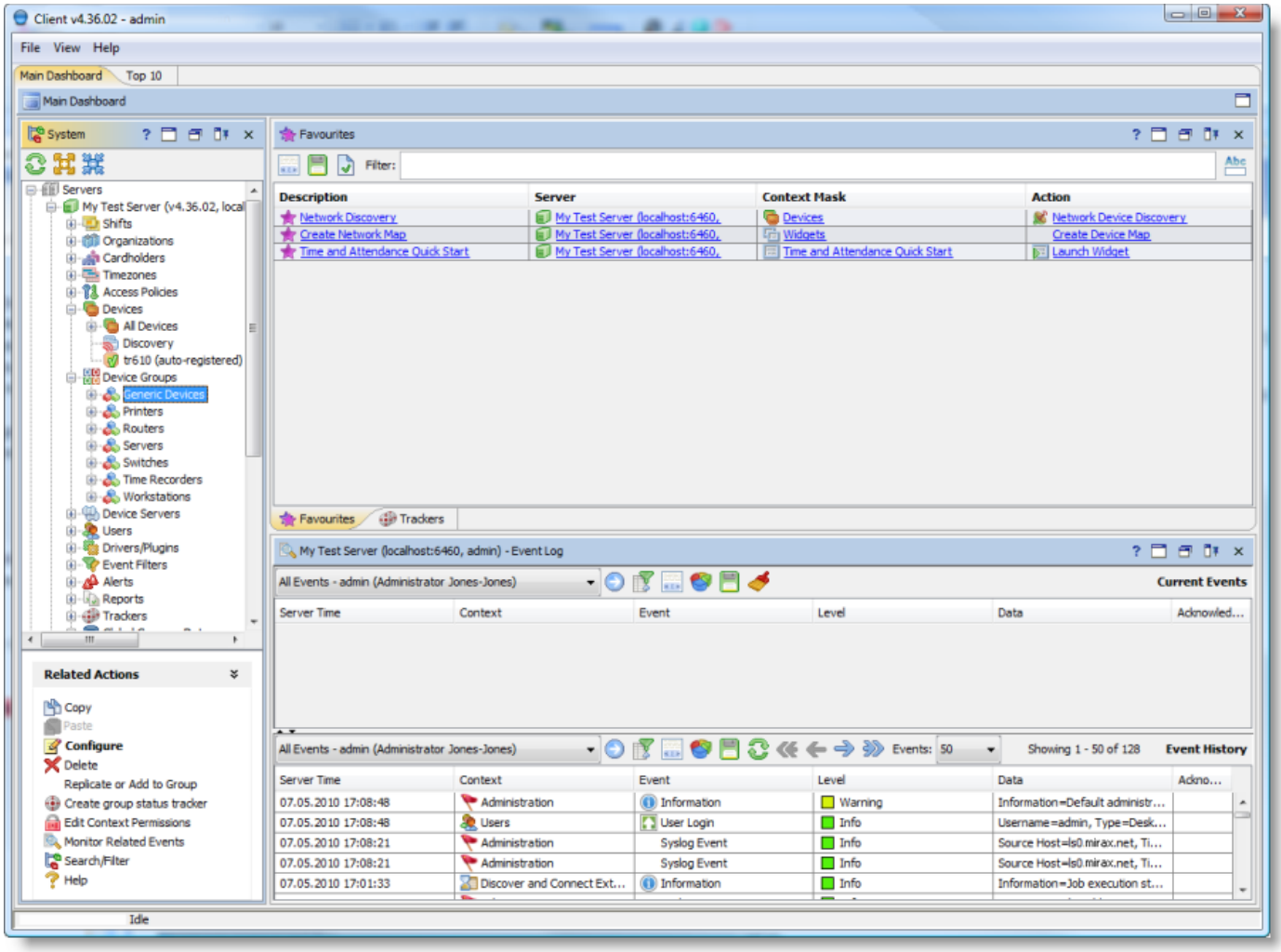
Многие операции позволяют задать месторасположение окна (окон) как параметр и сохранить его на сервере. См. дополнительную информацию в параграфе Расположение окна.

Операции с окнами

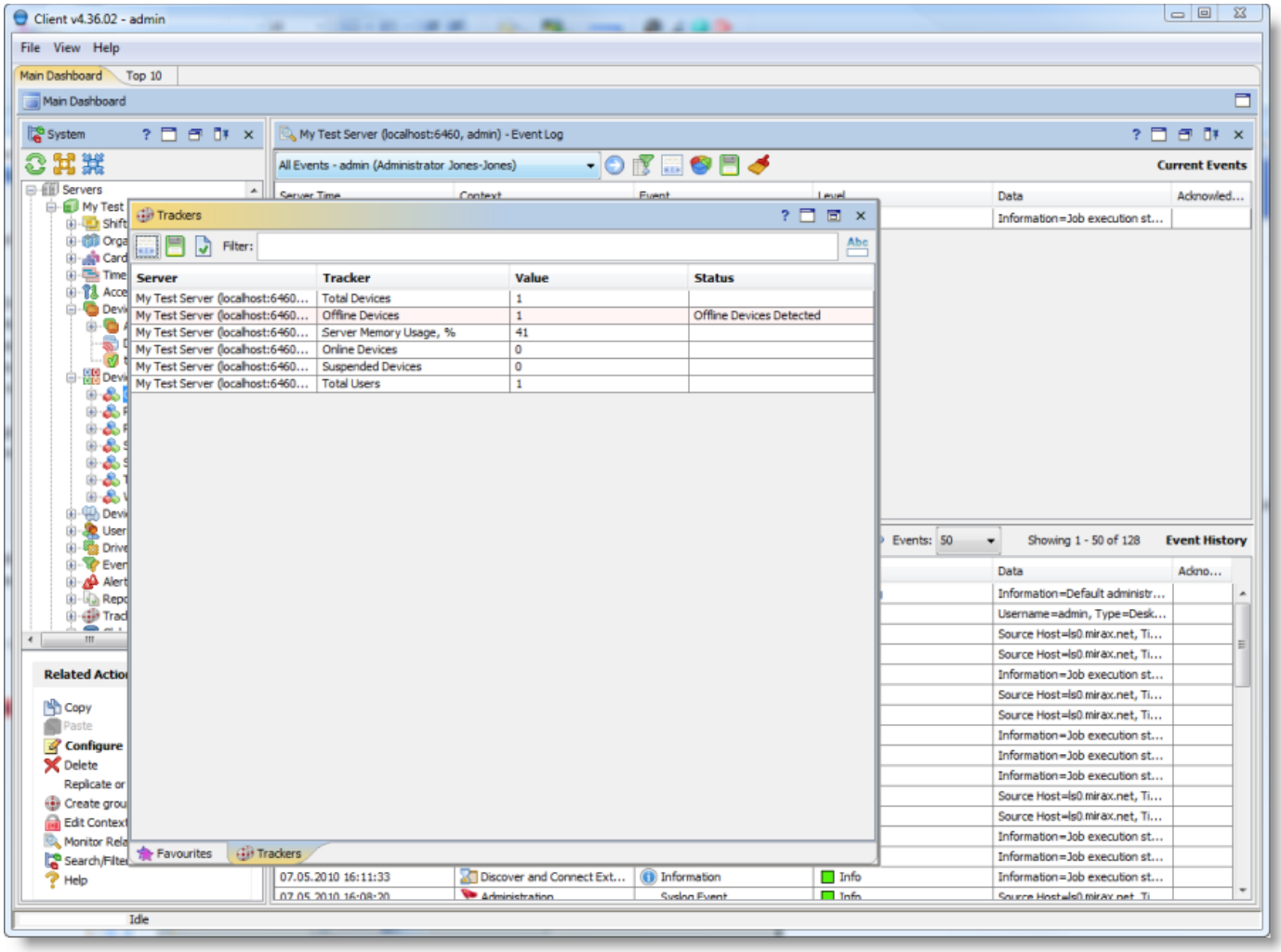
Любое из окон можно перетащить за его строку заголовка и оставить в следующем виде:

- Как плавающее окно, состыкованное с другими окнами
- Как плавающее окно, которое может "плавать" поверх других окон.
- Как боковую панель, которая обычно появляется в виде иконки в углу фрейма клиента"; эту панель можно распахнуть, если по ней кликнуть мышкой.
- Как новую закладку для окон с несколькими страницами. Это происходит, когда окно оставляют непосредственно поверх другого окна.
- Множество окон (таких как окно Системного дерева 370 или окно Журнала регистрации событий 398) можно скрыть (закрыть), а затем восстановить при помощи соответствующих элементов Главного меню 365.

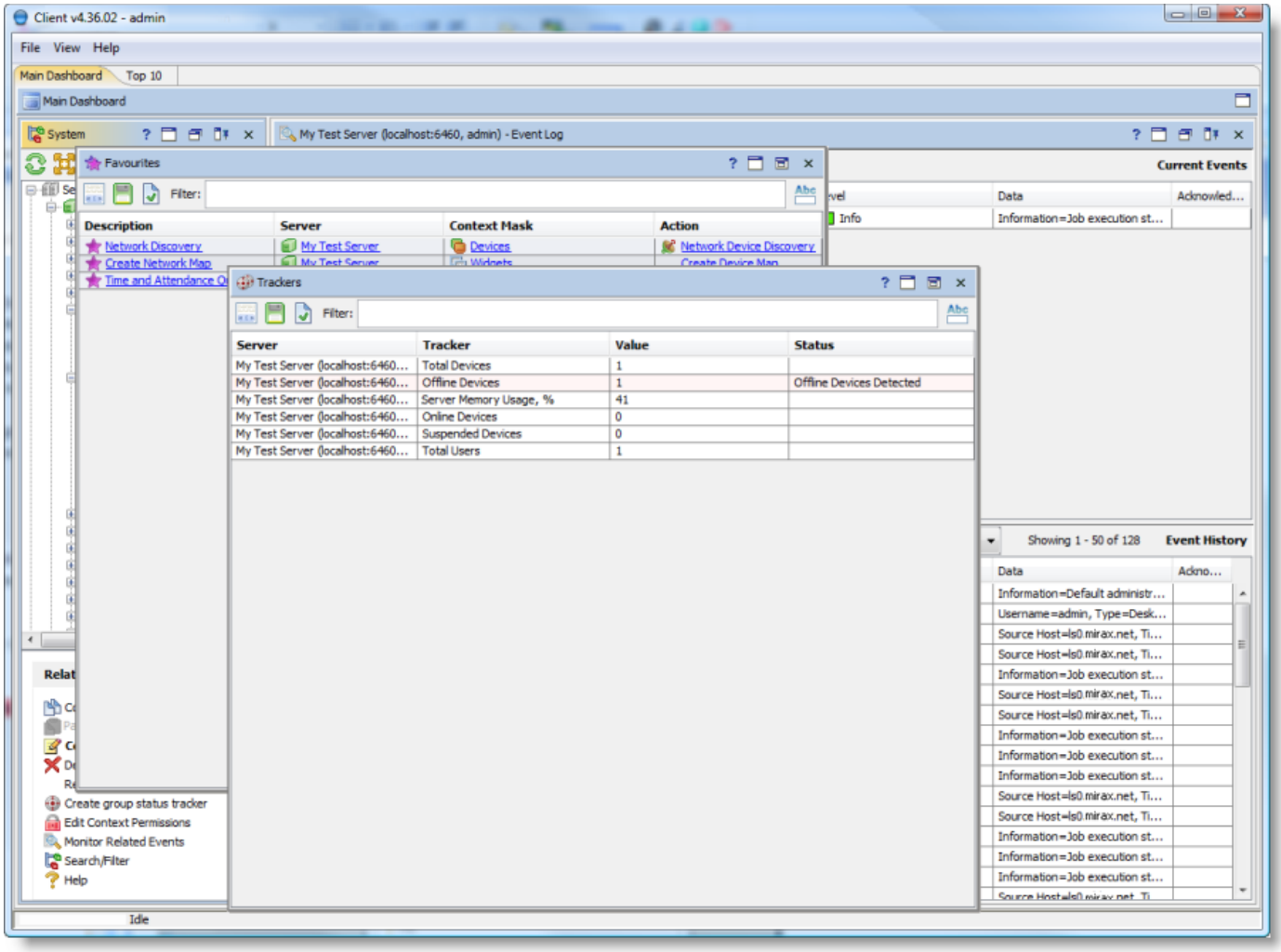
Окно по умолчанию выглядит следующим образом:



Вот как выглядят окна, когда они не закреплены и размещаются одно поверх другого для создания плавающей панели с вкладками. Обратите внимание на вкладки, располагающиеся снизу:



Так выглядят незакрепленные окна, представляющие собой плавающие мини-окна и располагающиеся каскадом:



Если окна имеют беспорядочный вид, можно вернуться к начальным настройкам, используя Главное меню 365.

8.8 Системное дерево

Системное дерево - это наиболее важный компонент AtomMind Client. Он используется, чтобы:

- Управлять соединениями AtomMind Server.
- Предоставлять доступ к настройкам рабочего пространства.
- Осуществлять поиск и управлять различными ресурсами сервера и устройствами (т.е. контекстами 41 сервера).

Системное дерево состоит из панели инструментов, самого дерева и контекстно - зависимого списка *соответствующих действий*. Каждый узел дерева представляет контекст[41] (который может иметь субконтексты).

Кликните здесь[45], чтобы узнать, в чем разница между контекстным деревом в Системном дереве и серверным контекстным деревом.

Панель инструментов системного дерева

Через панель инструментов осуществляется доступ к часто используемым операциям:

	Обновить	Обновляет текущий выбранный узел и все под-узлы.

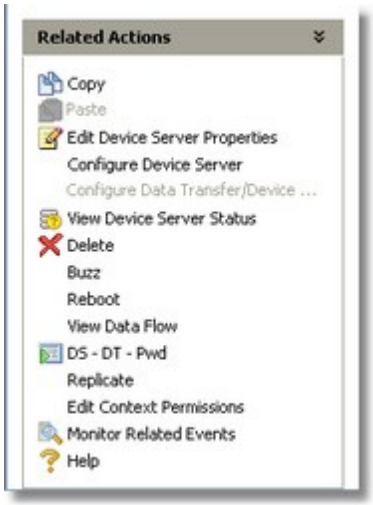
		Если используется для Узла сервера 376, эта кнопка заставляет AtomMind Client переподключиться к серверу.
	Развернуть рекурсивно	Раскрыть выбранный узел и его дочерние узлы рекурсивно. Эта кнопка не действует, если выбран листовой узел (т.е. дочерние узлы отсутствуют).
	Свернуть рекурсивно	Скрывает выбранный узел и все его дочерние узлы рекурсивно. Эта кнопка не действует, если выбран листовой узел.
	Искать/Фильтровать текстовое поле	Включает поиск/фильтрацию узла 375. Функция поиска/фильтра производит поиск только среди открытых узлов, т.е. узлов, которые были автоматически расширены или хотя бы раз расширены оператором.

Существует два типа раскрытия и сворачивания узла: обычный и рекурсивный. Обычное раскрытие и сворачивание выполняются при нажатии на иконку [+] или [-], расположенную рядом с узлом. Операции рекурсивного раскрытия и сворачивания активируются, когда используется контекстное меню узла или инструментальная панель Системного дерева. Обычный режим раскрывает лишь сам узел, в то время как дочерние узлы, которые могли бы быть развернуты, остаются свернутыми. При рекурсивном режиме раскрывается не только узел, но и его дочерние узлы. То же самое касается и операций сворачивания.

Панель инструментов также используется для поиска и фильтрации 375 узлов.

Соответствующие действия

Список *соответствующих действий* такой же как и в нижней части (после горизонтального разделителя) контекстного меню 372 для выбранного (-ых) в настоящий момент узла (-ов) . Вы можете использовать его, чтобы запустить действие в один клик.



Подсказки для элементов списка действий не являются описаниями, а показывают *имена* действий. Эти имена необходимы для указания действий во многих компонентах системы.

8.8.1 Контекстное меню

Когда вы кликаете правой кнопкой мыши по узлу Системного Дерева, появится контекстное меню. Контекстное меню каждого узла включает несколько элементов:

- **Обновить узел**
- **Раскрыть узел рекурсивно**
- **Свернуть узел рекурсивно**

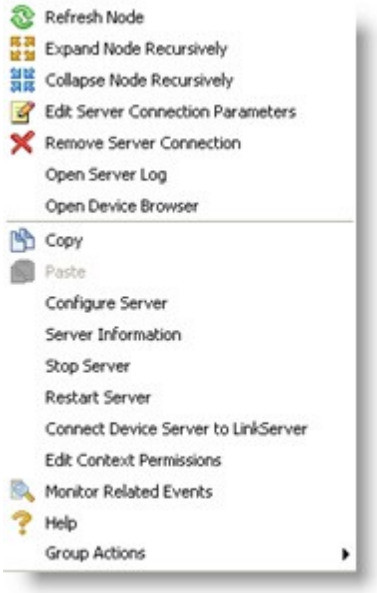
Другие опции являются специфичными для узлов. Операции, которые появляются до горизонтального разделителя, специфичны для узла текущего Системного дерева и задаются в самом AtomMind Client (они не являются действиями 87 AtomMind Server).

Операции, которые видны после горизонтального разделителя, запускают действия 87 из контекста 41 AtomMind Server, относящиеся к узлу Системного дерева. Данные действия позволяют запускать различные элементы пользовательского интерфейса 88.

Элемент меню, относящийся к действию по умолчанию 88 (т.е. действие, запускаемое двойным щелчком по узлу) выделен **жирным шрифтом**.

Для некоторых узлов контекстное меню также содержит подменю "Групповые действия". Оно содержит действие, которое Вы можете применить для каждого дочернего узла текущего узла. См. группировка действий 101.

Так выглядит контекстное меню системного дерева для узла AtomMind Server 376:



Работа с множеством узлов

Чтобы выделить группу узлов, кликните на узел, удерживая **Shift,** кликните на другой узел. Два узла, а также все находящиеся между ними другие узлы, окажутся выделенными. Чтобы выбрать множество разрозненных узлов (т.е. с невыбранными узлами между ними), удерживая **Ctrl,** кликните на узлы, которые Вы хотите выбрать. Если кликнуть правой кнопкой мыши по узлу, на экране появится контекстное меню с операциями, которые относятся ко всем выбранным узлам.

Дополнительную информацию о том, какие операции появляются в контекстном меню, когда выбраны несколько узлов, и как они себя ведут, можно найти в разделе группировка действий 101.

8.8.2 Операции по перетаскиванию мышью

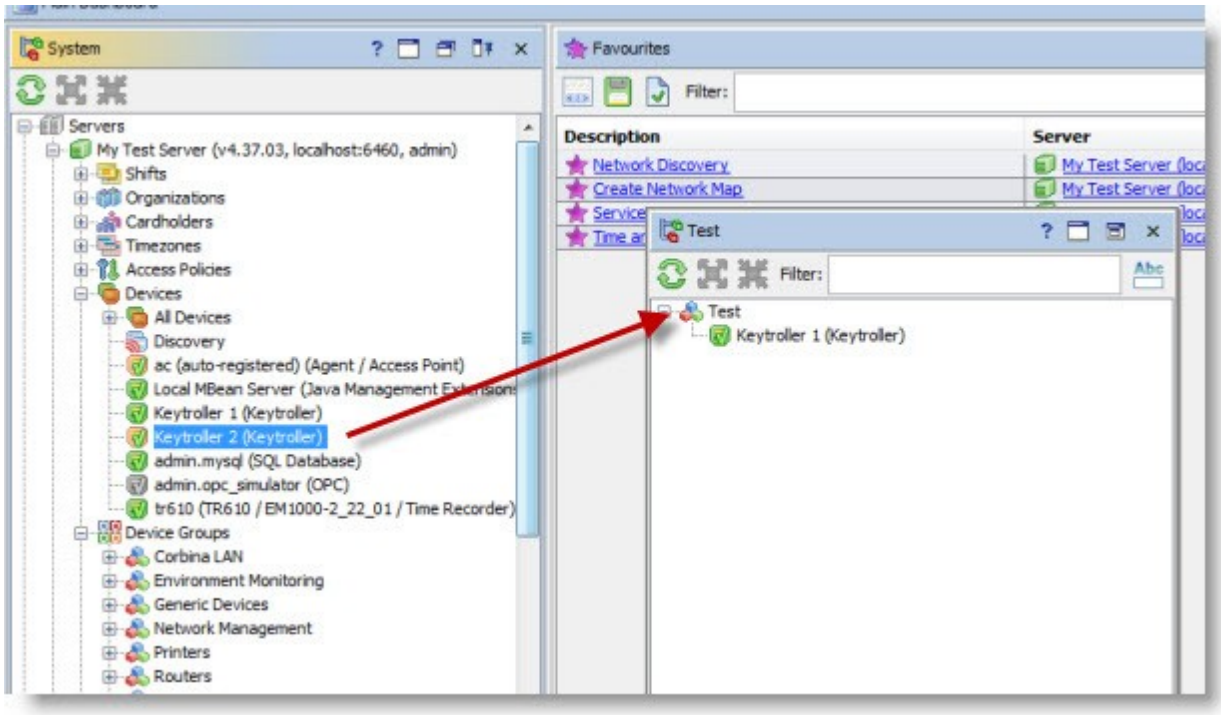
Системное дерево поддерживает операции *Перетаскивания мышью*.

Для того, чтобы выполнить перетаскивание мышью, кликните левой кнопкой мыши по узлу, удерживайте ее и перетащите на другой узел (эта операция называется перетаскивание) Теперь отпустите кнопку и "бросьте" узел в нужное место. Если действие является недопустимым или бессмысленным, перетащить узел не удастся (например, перетащить Device на узел Уведомления об ошибке - что в этом случае следует делать AtomMind Client ?). Если узел можно перетащить, курсор мыши изменится.

Если узел, на который Вы пытаетесь перетащить (Ваша цель) в настоящий момент находится "внутри" другого свернутого узла, просто удерживайте курсор над родительским узлом некоторое время, и он развернется.

Если целевой узел поддерживает несколько операций по перетаскиванию 101 для удаляемого контекста, пользователю предложат выбрать одно действие из контекстного меню:

Удобно добавлять множественные объекты в группу, открыв группу 751 в отдельном Системном дереве с помощью действия Поиск/фильтр 110 и перетаскивая мышью объектов из "главного" Системного дерева в узел группы в Системном дереве группы.



8.8.3 Операция копировать-вставить

Команды контекстного меню копировать-вставить работают также как и операции по перетаскиванию мышью 373 между узлами. Операция Копирования узла ***A*** и вставки его в узле ***B*** аналогичны операции перетаскивания узла ***A*** в узел ***B***.

Если узел Системного дерева связан с контекстом 41 AtomMind Server (большинство узлов связаны), его копирование и вставка в любом текстовом поле означает вставку полного пути для контекста сервера. Например, если вы заходите в **Запросы > Все пользователи**, кликаете по этому узлу правой кнопкой мыши и выбираете **"копировать"**, в буфере обмена будет содержаться полный путь к этому контексту - `users.admin.queries.all_users`. Это может пригодиться, когда необходимо задать контекстное имя при создании нового уведомления 779 или в любом другом случае, когда требуется полный путь для контекста.

8.8.4 Переупорядочивание узлов

Для некоторых узлов поддерживается режим ручной сортировки, или переупорядочивания их дочерних узлов. Для этих узлов вы можете перетащить дочерний узел и оставить его *между* другими дочерними узлами, когда появляется символ вставки (короткая горизонтальная черта между элементами):

Это свойство помогает:

- Переупорядочить учетные записи сервера

- Изменить последовательность загрузки в действиях авто-запуска 941
- Переупорядочить датчики 2181 и членов группы 751

8.8.5 Поиск и фильтрация узлов

Чтобы включить поиск/фильтрацию узлов, введите какой-либо текст в текстовое поле фильтра, находящееся в панели инструментов 370 системного дерева.

Выпадающее меню текстового поля фильтра позволяет указать следующие опции фильтра:

- Обработка заглавных букв: **с учетом регистра** или **без** учета регистра
- Режим фильтра: **нормальный, подстановочные знаки** или **регулярные выражения** 2142
- Режим соответствия шаблону: **совпадение с начала**, **полное совпадение** или **любое совпадение**
- Сравнивать только краевые узлы
- Скрыть узлы без дочерних элементов
- Сохранять дочерние элементы при совпадении их родительских узлов

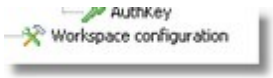
8.8.6 Нестандартные узлы

Некоторые узлы в Системном дереве не относятся напрямую к контекстам AtomMind Server. Эти узлы использует AtomMind Client:

- Настройка рабочего пространства 375
- Серверы 375
- Сервер 376

8.8.6.1 Конфигурация рабочего пространства

Этот узел находится в Системном дереве 370 и позволяет Вам настраивать собственное Рабочее пространство.



Пользовательские действия

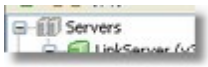
НАСТРОИТЬ РАБОЧЕЕ ПРОСТРАНСТВО (действие по умолчанию)

Позволяет редактировать конфигурацию открытого в настоящий момент рабочего пространства.

В текущий момент, есть лишь один параметр настройки рабочего пространства: пароль для рабочего пространства. Этот пароль используется для шифрования паролей, используемых для соединений с серверами сервера 363 располагающимися на рабочем пространстве.

8.8.6.2 Узел серверов

Предоставляет доступ к списку Учетных записей сервера 363, зарегистрированных на открытом в настоящий момент рабочем пространстве.



Учетные записи сервера, расположенные под Узлом серверов можно реорганизовать, перетащив их мышью.

Действия пользователя

НОВОЕ СОЕДИНЕНИЕ С СЕРВЕРОМ (действие по умолчанию)

Регистрирует новое Соединение с сервером 363 под открытым в настоящий момент рабочим пространством:

- **IP-адрес или DNS-имя**. Адрес по умолчанию **--** 'localhost', используемый когда AtomMind Server устанавливается на одном и том же компьютере, что и AtomMind Client.
- **Номер порта.** Порт AtomMind Server , используемый для соединений с AtomMind Client. Значение по умолчанию 6460.
- **Имя пользователя**. Имя учетной записи пользователя 478 на сервере.
- **Пароль**. Пароль учетной записи пользователя.

8.8.6.3 Узел сервера

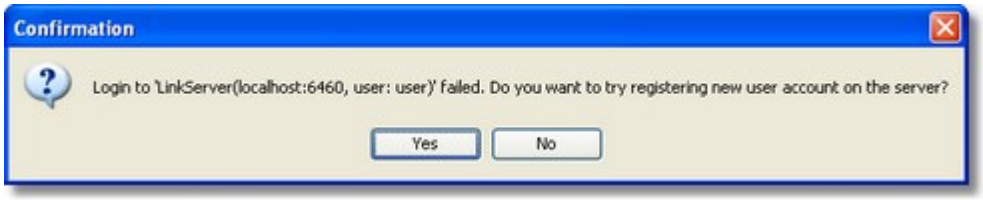
Узел *сервер* Системного дерева 370 - это корень всех узлов, относящихся к определенному AtomMind Server. Он обеспечивает доступ к основным операциям по управлению сервером.

У этого узла есть три состояния:

- Соединенный ( )
- Соединенный с предупреждениями ( )
- Отсоединенный ( )
- Заблокированный ( )

Соединение с сервером автоматически устанавливается, когда добавлен узел, или запущен AtomMind Client. Если соединение или логин по какой-либо причине отклонены (сервер недоступен, пароль неверный и пр.), у узла появляется статус *Отсоединенный*.

Если логин отклонен, Вам предложат зарегистрировать новую учетную запись на сервере:



Регистрация новой учетной записи возможна, лишь если включена опция глобальной конфигурации 179 "Саморегистрация пользователей". Вам предложат эта сделать, даже если эта опция отключена на сервере, потому что у AtomMind Client нет иного способа сообщить, включена она или нет.

AtomMind Client постоянно отслеживает соединение с каждым AtomMind Server. Если сервер оказывается недоступным, у узла появляется статус *Отсоединен*.

Если узел Сервера в Отсоединенном состоянии, AtomMind Client будет периодически пытаться к нему подсоединиться. Чтобы подсоединиться вручную, дважды кликните по узлу.

Действия пользователя

РЕДАКТИРОВАТЬ ПАРАМЕТРЫ СОЕДИНЕНИЯ С СЕРВЕРОМ

Это действие выполняется по умолчанию, когда установлено соединение с сервером. В противном случае, дважды кликните по узлу - это заставит AtomMind Client подключиться к AtomMind Server.

Это действие используется для изменения настроек для текущего Соединения с сервером 363:

- **IP-адрес или DNS-имя**. Адрес сервера.
- **Номер порта** для подключения клиентов. Порт, используемый для соединения с AtomMind Client.
- **Имя пользователя**. Имя учетной записи пользователя 478 на сервере.
- **Пароль**. Пароль пользователя учетной записи.
- **Описание**. Имя сервера или иное текстовое описание учетной записи.

- **Неактивный**. Флажок указывает на то, что учетная запись неактивна. AtomMind Client не пытается установить связь, используя неактивные учетные записи.
- **Время ожидания соединения**. Определяет, как долго клиент будет ожидать ответа от не отвечающего сервера, после чего сервер будет считаться недоступным. Этот параметр следует увеличить для удаленных серверов со слабым или ненадежным соединением.
- **Время ожидания команды**. Определяет максимальное время исполнения для одной команды. Этот параметр следует увеличить в редких случаях, когда у сервера чрезвычайно медленно исполняемые операции, например, составление отчета может занять несколько часов.

УДАЛИТЬ ПОДКЛЮЧЕНИЕ К СЕРВЕРУ

Удаляет подключение к серверу из рабочего пространства.

ОТКРЫТЬ ЖУРНАЛ СЕРВЕРА

Открывает Журнал событий 398 из журнала сервера 166. Отображаются лишь события **информационного** (или выше) уровня 171. Это действие доступно, только если AtomMind Client запущен в режиме администратора 362.

ОТКРЫТЬ БРАУЗЕР УСТРОЙСТВ

Открывает компонент Браузер устройств 475 текущего сервера. Это действие доступно, только если AtomMind Client запущен в режиме администратора 362.

ПРОСМОТРЕТЬ СТАТИСТИКУ СОЕДИНЕНИЯ

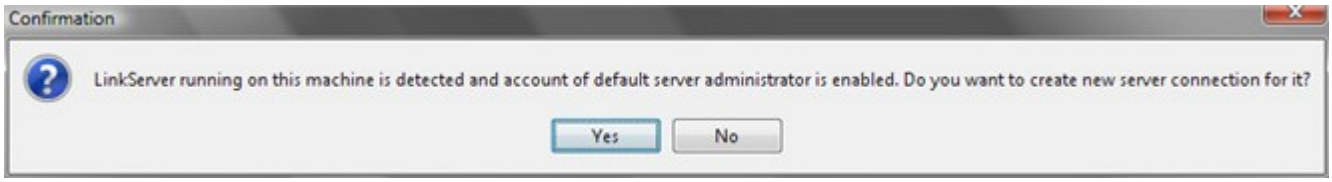
Показывает статистическую информацию о подключении к серверу:

- Длительность соединения
- Число выполненных за время соединения команд
- Среднее время ответа сервера
- Объем входящего и исходящего трафика
- Количество неуспешных (неотвеченных) команд

Это действие доступно, только если AtomMind Client запущен в режиме администратора 362.

8.8.7 Автоподключение к локальному серверу

Если при загрузке AtomMind Client не находит настроенных соединений к серверам (т.е. под узлом Серверы 375 нет узлов Сервер) 376, AtomMind Client пытается соединиться с AtomMind Server, запущенным на порту по умолчанию на локальном компьютере. Если соединение установлено успешно, а AtomMind Client может залогиниться на сервер под учетной записью администратор по умолчанию 479 (с логином/паролем по умолчанию), система предложит Вам установить новое автоматическое Подключение к серверу:



Это свойство позволяет пользователям, только осваивающим систему, получить доступ к AtomMind Server, чтобы его протестировать и выполнить начальную настройку системы.

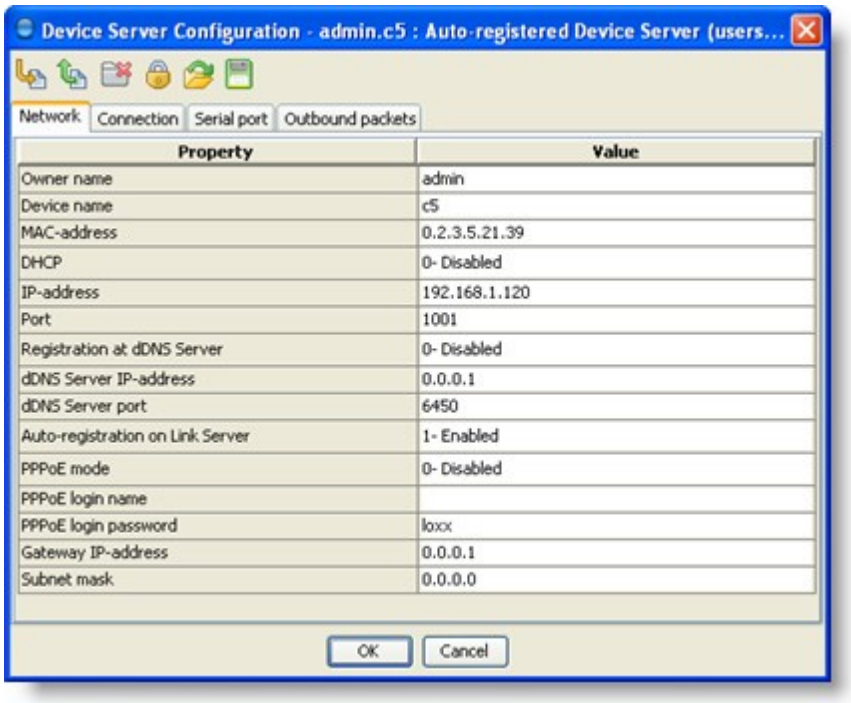
8.9 Редактор свойств

Редактор свойств используется для изменения свойств различных контекстов 41. Например, настройки для AtomMind Server и Device следует редактировать в Редакторе Свойств.

Технически Редактор Свойств позволяет изменять значения одной и более переменных 61 (свойств) контекста. Когда редактор начинает работу с контекстом, он загружает значение каждой переменной для контекста, а затем позволяет пользователю редактировать эти значения и записывает их назад в контекст.

Каждое значение (переменная) редактируются в отдельном компоненте Редактора таблиц данных 382 потому, что каждое значение, в действительности, - это одна Таблица данных 49.

Редактор данных состоит из панели инструментов и окна свойств:



Панель инструментов

	Перезагрузить свойства. Система заново загружает все свойства из контекста источника. Недавно измененные в редакторе значения утрачиваются.
	Сохранить свойства. Система сохраняет значения измененных свойств.
	Включить/отключить закладки. Система переключает редактора свойств с формата с вкладками на постраничный формат и наоборот.
	Включить/отключить редактирование. Временно переключает редактора с/на режим только для чтения 378.
	Импорт свойств из файла. Система импортирует 382 значения свойств из файла.
	Экспорт свойств в файл. Система экспортирует 382 значения свойств в файл.
	Помощь. Показывает документацию/помощь для редактируемых свойств.

Если Редактор Свойств отображается внутри диалогового окна, у него есть кнопки **OK** и **Cancel**. Кнопка **OK** сохраняет значения любых значений, который были изменены, и закрывает диалоговое окно. Кнопка **Cancel** запрещает выполнение операции без сохранения.

У большинства элементов редактора свойств, также как и у свойств, есть подсказки. Они появляются при наведении на элемент курсора мыши.

Контекстное меню

Контекстное меню отображается, если правой кнопкой мыши кликнуть по одному из свойств Редактора свойств. Он содержит список Действий, связанных с переменной 102, которые "знают", что делать с выбранной переменной. Количество и типы действий, связанных с переменной, зависят от типа переменной, для которой отображается контекстное меню.

Режимы редактора свойств

Редактор свойств может работать в двух режимах:

- Обычном режиме
- Режиме Только для чтения

Режим Только для чтения не разрешает редактировать или сохранять свойства.

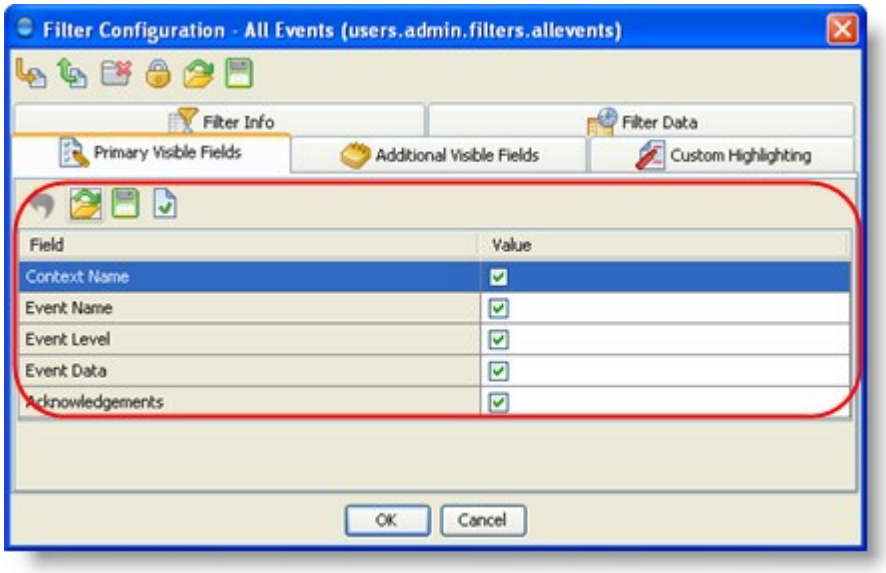
Существуют также два режима отображения Редактора свойств:

- Обычный режим
- Расширенный режим

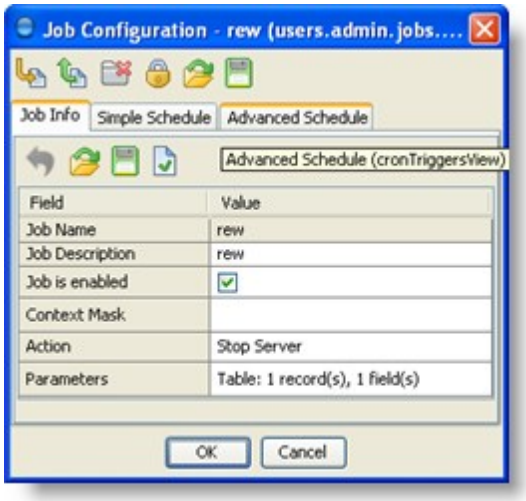
Режимы отображения свойств

ПРОСТОЙ РЕЖИМ

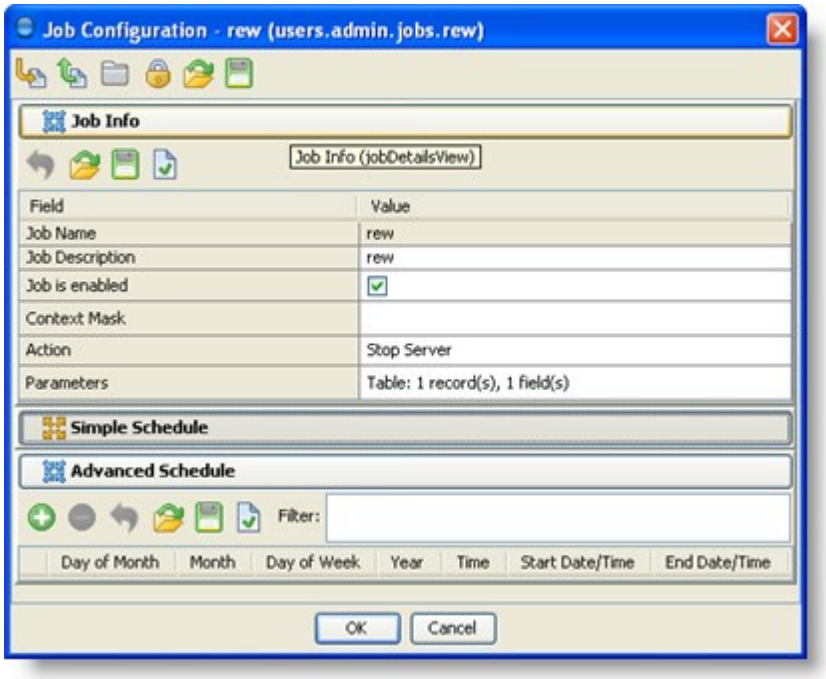
При *Простом режиме,* каждый редактор таблиц данных, отображающий значение отдельного свойства, отображается непосредственно в панели свойств. Отдельное окно при этом не открывается. На рисунке снизу Редактор Таблиц Данных, содержащий значение для отдельного свойства, отмечен красным прямоугольником.



Если включены вкладки, значение каждого свойства отображается в отдельной вкладке. Имя свойства и подробное описание содержатся в подсказке, расположенной на вкладке для этого свойства.

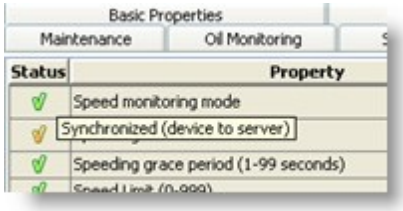


Если вкладки отключены, все свойства отображаются на одной странице. Значения каждого свойства отображаются в отдельной раскрывающейся панели. Чтобы раздвинуть или свернуть эту панель, по ней следует кликнуть мышью. Имя свойства и подробная информация содержатся в подсказке на этой раскрывающейся панели.

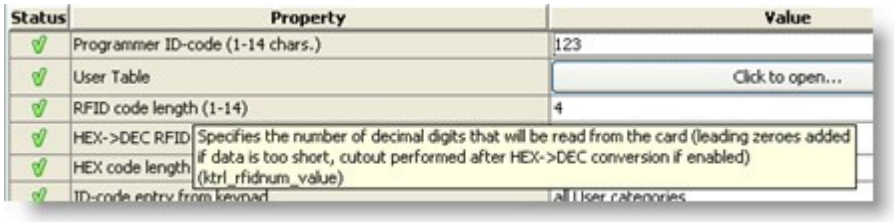


РАСШИРЕННЫЙ РЕЖИМ

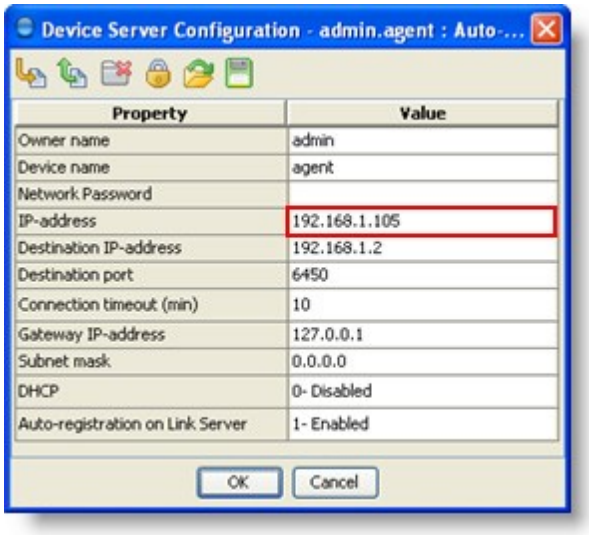
При *Расширенном режиме* свойства группируются согласно тому, к какой группе переменных 61 они относятся. Свойства каждой группы отображаются в виде таблицы с двумя или тремя колонками. Первая колонка опциональна и может содержать иконку, отображающую статус свойства. Редактор свойств постоянно отслеживает статус каждой переменной и обновляет статусную иконку при каждом изменении. Если на иконку навести курсор мыши, появится подсказка с информацией о статусе:



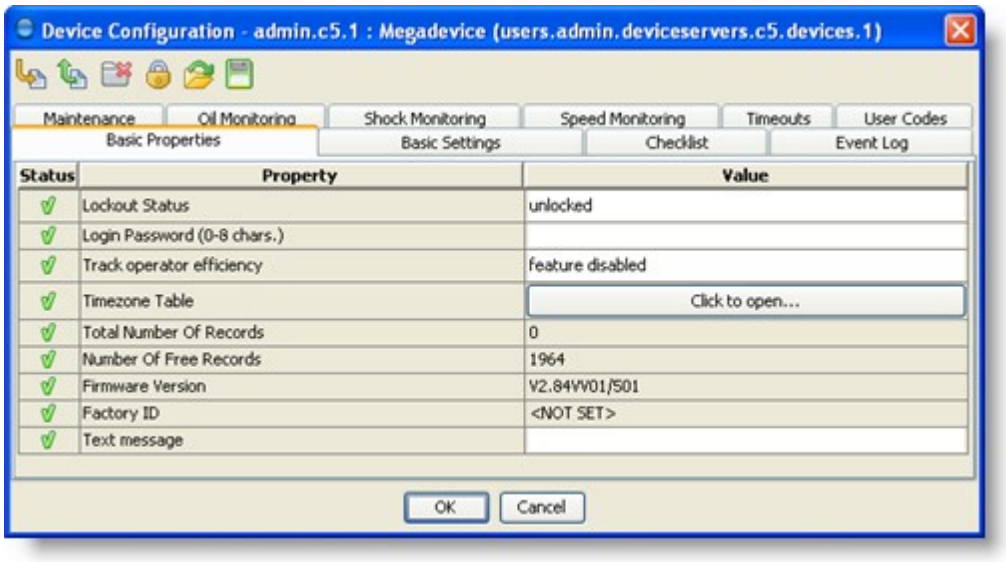
Колонка **свойств** содержит описание свойств. Подсказки в ячейках содержат информацию о каждом имени свойства и его описание:



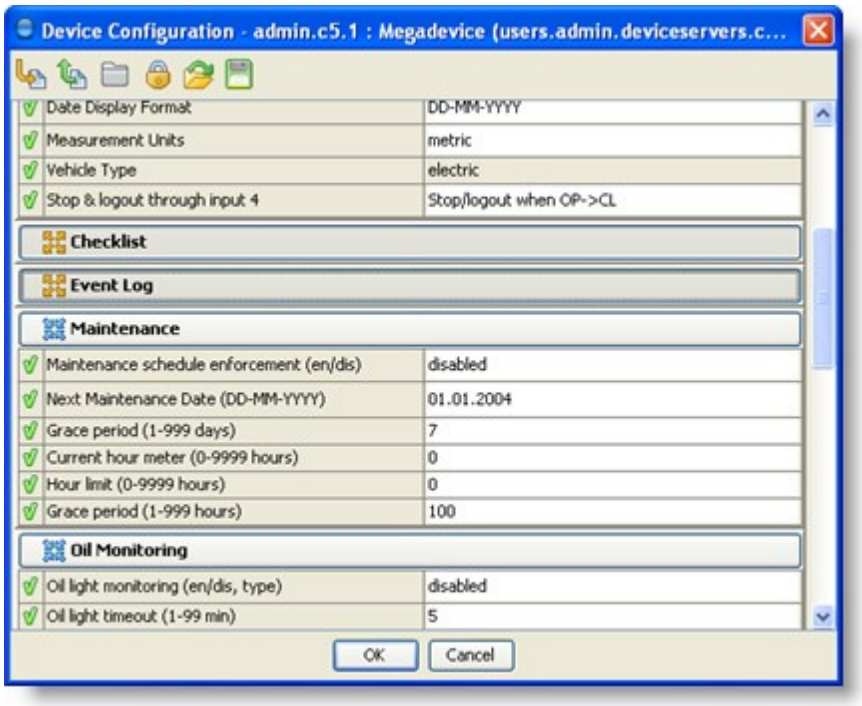
Содержимое третьей колонки изменяется согласно с определениям переменной 61 редактируемого контекста. Если Таблица данных 49, отображающая значение свойства, всегда имеет одно поле с одной записью, Редактор таблицы данных 382 отображается непосредственно в третьей колонке. В других случаях, третья колонка будет содержать кнопку **[Click to open...],** которая открывает редактор таблиц данных в новом окне. Редактор таблиц данных, вставленный в таблицу, отмечен красным:



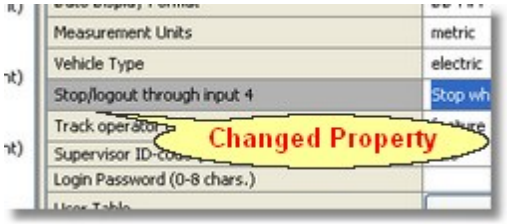
Если выбранная разметка включена, свойства каждой группы отображаются в отдельной вкладке. Имена групп отображены как заголовки вкладок.



Если вкладки отключены, все свойства отображаются на одной странице. Свойства каждой группы отображены в отдельной развертывающейся панели. Щелчком мыши по панели можно ее развернуть или свернуть. Имена групп отображаются в заголовках распахивающихся панелей.



При расширенном режиме цвет фона измененных свойств меняется от серого до темно-серого:



Импорт и экспорт свойств

Свойства можно экспортировать и импортировать в/из внешних файлов. По умолчанию, файлы свойств имеют расширение `.prs`. Свойства в этих файлах кодируются с ипользованием формата XML[2136].

Свойства импортируются по имени: если Редактор Свойств содержит свойство с тем же именем, что и сохраненное в файле, их значения сливаются при выполнении операции Умное копирование таблиц данных.

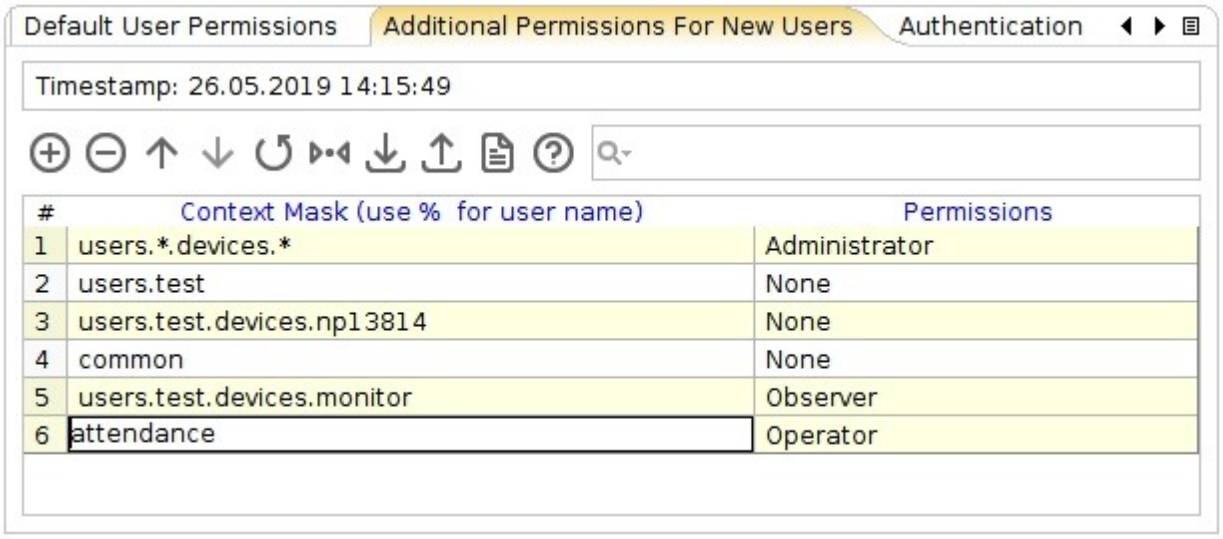
Процедура импортирования свойств не может просто заменить значения, содержащиеся в редакторе, на прочтенные из файла, потому, что они могут иметь разный формат[49]. Но в любом случае, импортируется как можно большее количество данных.

Горячие клавиши

Alt-L	Перегзаружает свойства
Als-S	Сохраняет свойства

8.10 Редактор таблиц данных

Редактор таблицы данных - это компонент, используемый для редактирования или просмотра одной Таблицы данных[49]. Его использует Редактор свойств[377] для редактирования переменных[61] контекста[41], Журнал регистрации событий[398] - для просмотра Таблиц данных, связанных с событиями[73], GUI Процедура Редактировать данные[90] для редактирования параметров ввода функции[70] и просмотра ее вывода и т.д.



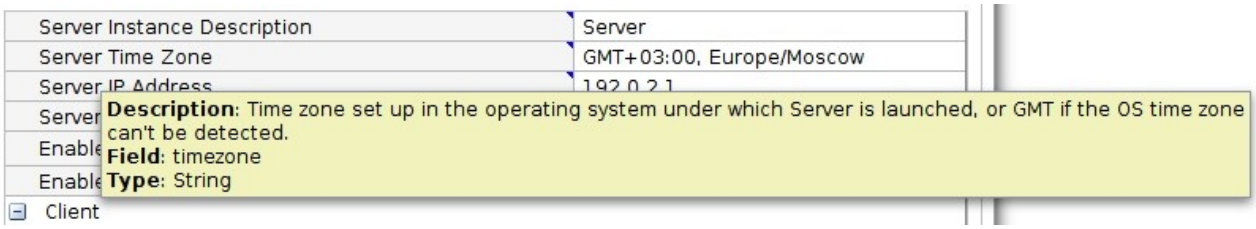
У большинства элементов Редактора Таблиц данных, таких как заголовки колонок и ячейки с данными, обычно есть подсказки, которые появляются, если мышью навести на один из элементов.

Подсказка для заголовка колонки содержит **тип** поля и строку **помощи**, заданную в формате[49] таблицы, если есть таковой.

Подсказка для ячейки содержит строковое представление значения ячейки и тип поля, используемого для этой ячейки. Это может пригодится:

- Если значение ячейки слишком длинное и не вмещается в ячейку,
- У редактора ячейки есть значения выбора[49], и Вы хотите сначала увидеть само значение, а не его описание в раскрывающемся списке.

Если подсказка ячейки содержит важную расширенную информацию, ячейка отмечена небольшим синим треугольником:



У колонок таблицы можно поменять размер и порядок. Чтобы изменить размер колонки, кликните по его полю и перетащите мышью вправо или влево. Чтобы изменить порядок, перетащите заголовок колонки в новое место.

Двойной щелчок мышью по разделителю между заголовками двух колонок автоматически скорректирует ширину колонки слева (т.е. скорректирует ее размер так, чтобы в ней помещались соответствующие данные).

ИЗМЕНЕННОЕ ПРЕДСТАВЛЕНИЕ ЗАПИСЕЙ

AtomMind использует цветовую индикацию для новых записей (зеленый) и для измененных (желтый).

Отображение временных меток

Редактор таблицы данных использует настройки **Шаблон даты**, **Шаблон времени** или **Временная зона**, заданные в Свойствах аккаунта пользователя[481] для преобразования и отображения различных временных меток.

Все значения даты/времени показаны во временной зоне активного пользователя. Если временная зона не задана в настройках пользователя, будет использована временная зона, заданная на ПК, на котором установлен компонент.

Обработка привязок данных

Редактор таблицы данных выполняет привязки данных[741] в формате[49] редактируемой таблицы данных. Когда редактор загружается, он прочитывает список привязок для таблицы и обрабатывает их. Привязки оцениваются и используются для изменения ячеек в редактируемой таблице. Выражения привязок, используемые Редактором Таблицы данных могут содержать ссылки на ячейки редактируемой таблицы. а также ссылки на различные

контексты 41. Чтобы узнать, как создавать эти ссылки, ознакомьтесь с информацией о привязках данных здесь 49.

8.10.1 Источники данных

Редактор таблицы данных может использовать два вида источников данных:

Таблица данных

Это наиболее часто используемая опция и опция по умолчанию, которая отображает одну Таблицу данных. В этом режиме недоступна прокрутка. Фильтрация и сортировка могут быть применены только к просматриваемым/редактируемым записам таблицы.

Список объектов

В данном режиме редактируемая таблица представляет собой часть большого списка объектов(таких как экземпляры класса 885). Она доступна на стороне AtomMind Server.

Список объектов используется как источник данных, прокрутка и кнопки обновления доступны на панели инструментов. Если применены правила фильтрации или сортировки, объекты на стороне сервера фильтруются/сортируются вместо записей в таблице, видимой в данный момент.

8.10.2 Режимы

Редактор таблиц данных может работать в различных режимах, описанных в этом разделе.

РЕЖИМЫ РЕДАКТИРОВАНИЯ

Доступно два режима редактирования:

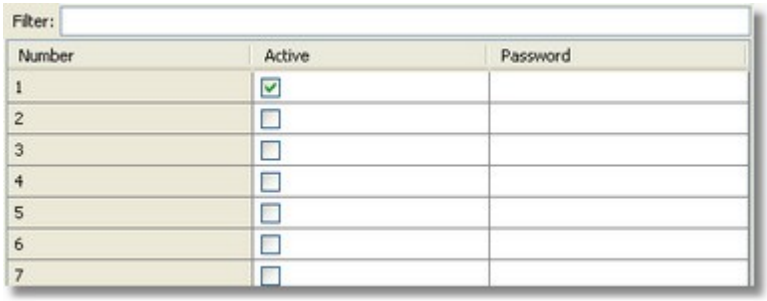
- **Стандартный** режим. Этот режим позволяет редактировать значения.
- Режим **Только для чтения**. В этом режиме не разрешено редактирование данных.

РЕЖИМ ПРЕДСТАВЛЕНИЯ ДАННЫХ

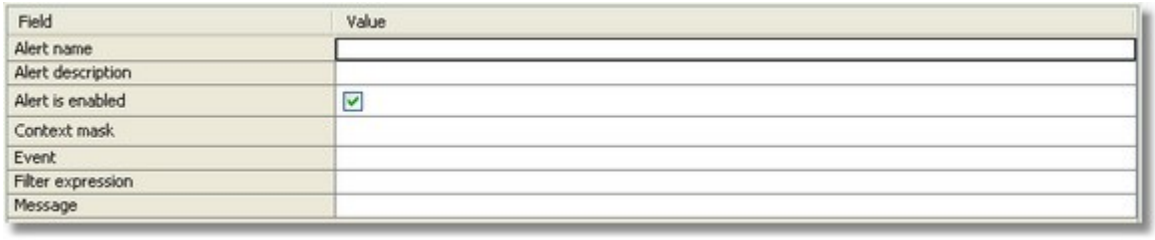
Существует два режима представления данных:

- Режим **Нескольких записей**
- Режим **Одной записи** (также названный **Вертикальным** режимом)

В режиме нескольких записей, заголовок таблицы показывает описание для каждой колонки, и каждый ряд представляеь собой одну запись Таблицы данных:



Режим одной записи используется Редактором таблиц данных, когда редактируемая таблица содержит только одну запись, и дополнительных записей не может быть добавлено. Этот режим предназначается для компоновки из двух колонок: левая колонка показывает имена полей, а правая содержит значения:



8.10.3 Панель инструментов

Инструментальная панель Редактора таблиц данных предоставляет доступ к операциям, которые могут быть выполнены с таблицей данных 49.

Показать/Скрыть дополнительные свойства	Данная кнопка-переключатель используется для показа и скрытия полей колонок, отмеченных как дополнительные.
Обновить	Доступно только в режиме **Список объектов на стороне сервера**. Кнопка Обновить перезагружает весь список и обновляет текущую страницу.
Добавить ряд	Вставить новый ряд перед выбранным рядом. Если ряд не выбран, добавляется новый ряд в конце таблицы. Эта кнопка неактивна или скрыта, если ряды не могут быть добавлены в таблицу. Если компонент работает в режиме **Список объектов на стороне сервера**, эта кнопка вставляет новый объект в список объектов на стороне сервера, а не добавляет запись в видимую таблицу. Другой редактор таблиц данных позволяет заполнить поля созданного объекта. Новый объект станет видимым, только если он соответствует текущим правилам фильтрации и текущей странице. В большинстве случаев требуется клик по кнопке Обновить для просмотра нового объекта.
Удалить выбранные ряды	Удалить выбранный ряд(ы). Данная кнопка неактивна или скрыта, если ряды не могут быть удалены. Если этот компонент работает в режиме **Список объектов на стороне сервера**, то эта кнопка удаляет объект(ы) из списка объектов на стороне сервера в дополнение к удалению соответствующих записей из видимой таблицы.
Первый	Доступно только в режиме **Список объектов на стороне сервера.** Прокручивает список к первой странице.
Предыдущий	Доступно только в режиме **Список объектов на стороне сервера**. Прокручивает список к предыдущей странице.
Следующий	Доступно только в режиме **Список объектов на стороне сервера**. Прокручивает список к следующей странице.
Последний	Доступно только в режиме **Список объектов на стороне сервера**. Прокручивает список к последней странице.
Количество рядов	Доступно только в режиме **Список объектов на стороне сервера**. Определяет количество объектов, показываемых на одной странице. Верхние кнопки прокрутки могут быть использованы для навигации длинного списка.
Переместить ряд наверх	Переместить текущий ряд наверх. Эта кнопка не показывается, если ряды не могут перемещаться. Она изменяет актуальные данные в таблице (по сравнению с операцией сортировки, которая просто изменяет то, как показываются данные).
Переместить ряд вниз	Переместить текущий ряд вниз. Эта кнопка не показывается если ряды не могут перемещаться. Она изменяет актуальные данные в таблице (по сравнению с операцией сортировки, которая просто изменяет то, как показываются данные).
Отменить изменения	Отменить все сделанные в таблице изменения. Эта кнопка становится неактивной только после того, как произошло изменение.
Включить/отключить горизонтальную прокрутку	Если горизонтальная прокрутка неактивна, ширина всех колонок автоматически выравнивается для того, чтобы соответствовать ширине окна Редактора таблиц данных. Если прокрутка активна, Редактор таблиц данных имеет горизонтальную полосу прокрутки, а ширина всех колонок откорректирована для соответствия заполнению. Эта опция неактивна по умолчанию.
Импортировать таблицу данных	Импортирует 394 данные из файла.
Экспортировать таблицу данных	Экспортирует 394 данные в файл.
Сделать отчет	Формирует отчет 397 на основе просматриваемой/редактируемой таблицы.

	Помощь	Откройте раздел документации, посвященный просмотру/редактированию данных.

Некоторые (или даже все) кнопки панели инструментов могут быть скрыты или неактивны, если соответствующие операции недоступны в редакторе. Кнопка скрыта, если операция не применима к текущей таблице. Если операция недоступна в данный момент эта кнопка неактивна (напр. операция Удалить выбранные ряды неактивна, если нет выбранных рядов).

8.10.4 Редактирование данных

Если Редактор таблицы данных находится в режиме редактирования, значения в таблице можно изменять.

Средство отображения и редактор, используемые для отображения и изменения значений в каждой ячейке, зависят от нескольких факторов.

- Тип поля
- Редактор/Средство визуализации указанные в определении поля 49 (это часть формата таблицы)
- Доступность Значений выбора 49, определенных для этого поля в Формате Таблицы
- Нулевое ли значение поля

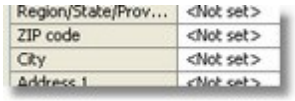
Особые Средства отображения/Редакторы

Есть несколько случаев, когда используются нестандартные редакторы:

СРЕДСТВО ОТОБРАЖЕНИЯ ПУСТОГО ЗНАЧЕНИЯ

Если поле может иметь пустое значение и оно не заполнено, это выглядит в редакторе как <нет значения>. Щелкните мышью по ячейке, чтобы начать редактирование.

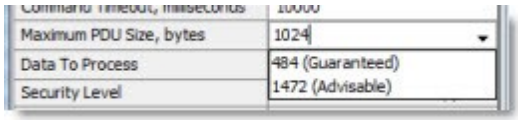
Можно поменять значение на нулевое при помощи операции **Удалить значение** в контекстном меню ячейки.



ВЫБОР ЗНАЧЕНИЙ

Если формат поля содержит выбор значений 49, его редактирование выполняется через раскрывающийся список:

Если флажок расширенного выбора значений 49 настроен на формат поля, в комбинированной ячейке можно ввести любое пользовательское значение:



Стандартные средства отображения/Редакторы

СТРОКОВЫЕ/ЦЕЛОЧИСЛЕННЫЕ/ДЛИННЫЕ ЦЕЛЫЕ/С ПЛАВАЮЩЕЙ ТОЧКОЙ ПОЛЯ

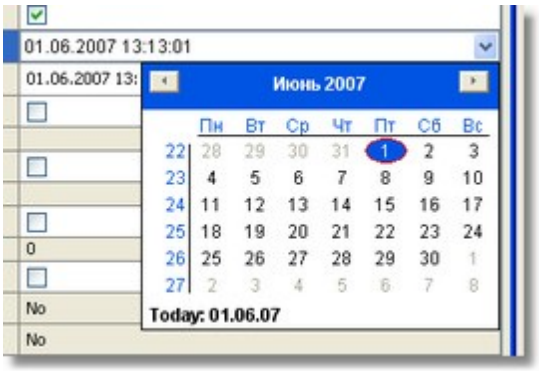
Значения типа *Integer, Long, String* и *Float* редактируются в обычном текстовом поле:

ЛОГИЧЕСКИЕ ПОЛЯ

Значения типа *Boolean* редактируются кнопкой-флажком и представлены как **Yes** (для TRUE) или **No** (для FALSE) в режиме только для чтения:

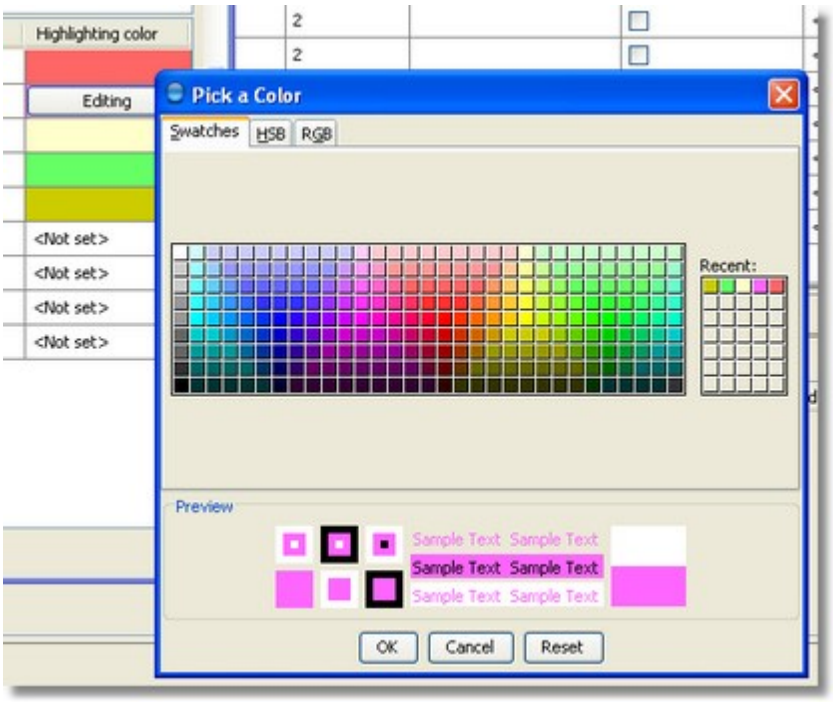
ПОЛЯ ДАТЫ

Значения *Date* и *Date/Time* редактируются в Элементе Выбора Даты:



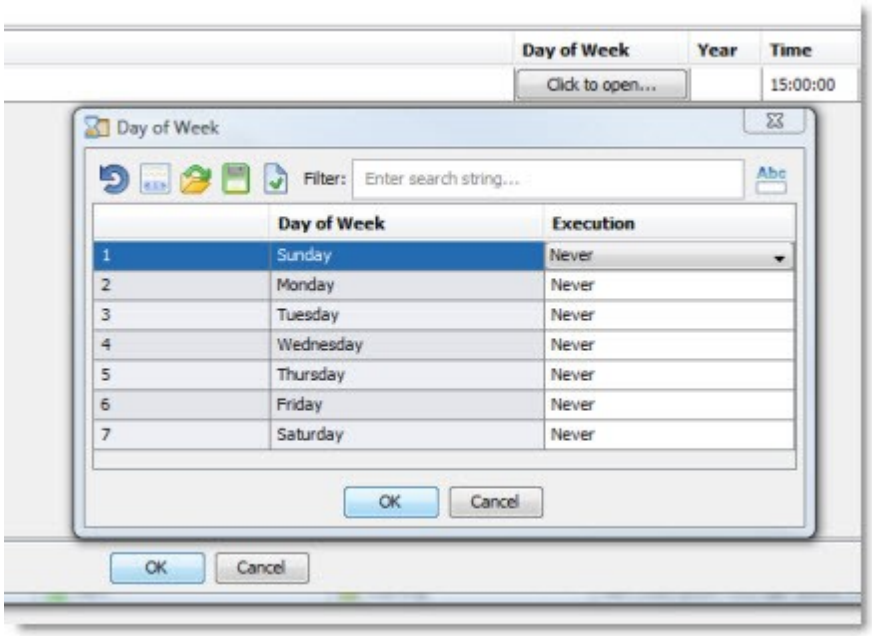
ПОЛЯ ЦВЕТА ИЗОБРАЖЕНИЙ

Цвета можно выбрать в Элементе Выбора Цвета:



ПОЛЯ ТАБЛИЦЫ ДАННЫХ

Поля *Data Table* (т.е. поля, чьи значения включены в таблицы данных) редактируются в вставленном Редакторе Таблицы Данных, который появляется в отдельном диалоговом окне. Диалоговое окно появляется, если кликнуть по ячейке, содержащей Таблицу Данных:



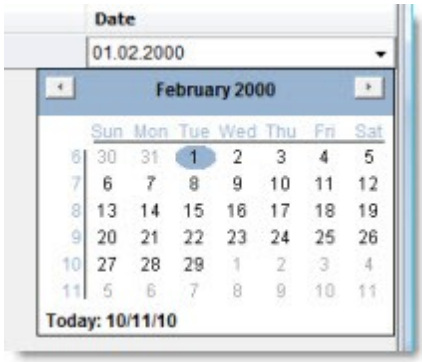
ПОЛЯ БЛОКА ДАННЫХ

Редактор Блока данных по умолчанию позволяет выбирать и сохранять файл любого типа из/в локальной файловой системы AtomMind Client's:

Дополнительные средства визуализации/редакторы

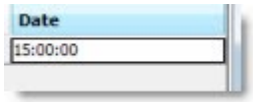
РЕДАКТОР ДАТЫ

Редактор дат позволяет задать дату в Элементе Выбора Даты, но в нем нельзя установить время:



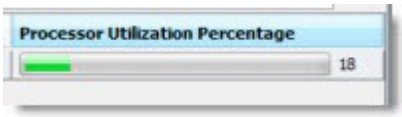
РЕДАКТОР ВРЕМЕНИ

Редактор времени позволяет задать время в форме строки, в нем нельзя выбрать дату:



ОТРИСОВЩИК ИНДИКАТОРА ПРОГРЕССА/СОСТОЯНИЯ

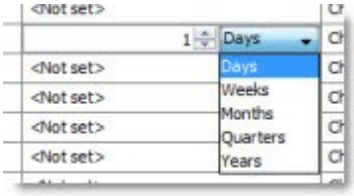
Этот отрисовщик показывает значения *Integer*, *Long*, *Float* и *String* в процентах от максимальных заданных опций редактора:



Строковые значения конвертируются в числовые.

РЕДАКТОР ПЕРИОДА ВРЕМЕНИ

Редактор периода позволяет задать период времени, как определенное число элементов выбранного времени:



РЕДАКТОР ПАРОЛЯ

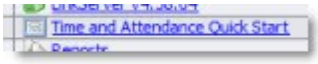
Редактор пароля аналогичен обычному текстовому полю, но символы в нем отображаются:

Значения нельзя скопировать из ячеек, использующих этот редактор.

СРЕДСТВО ВИЗУАЛИЗАЦИИ ССЫЛОК

Редактор ссылок отображает ссылки, активизируемые щелчком мыши, которые подчеркнуты и выделены голубым:

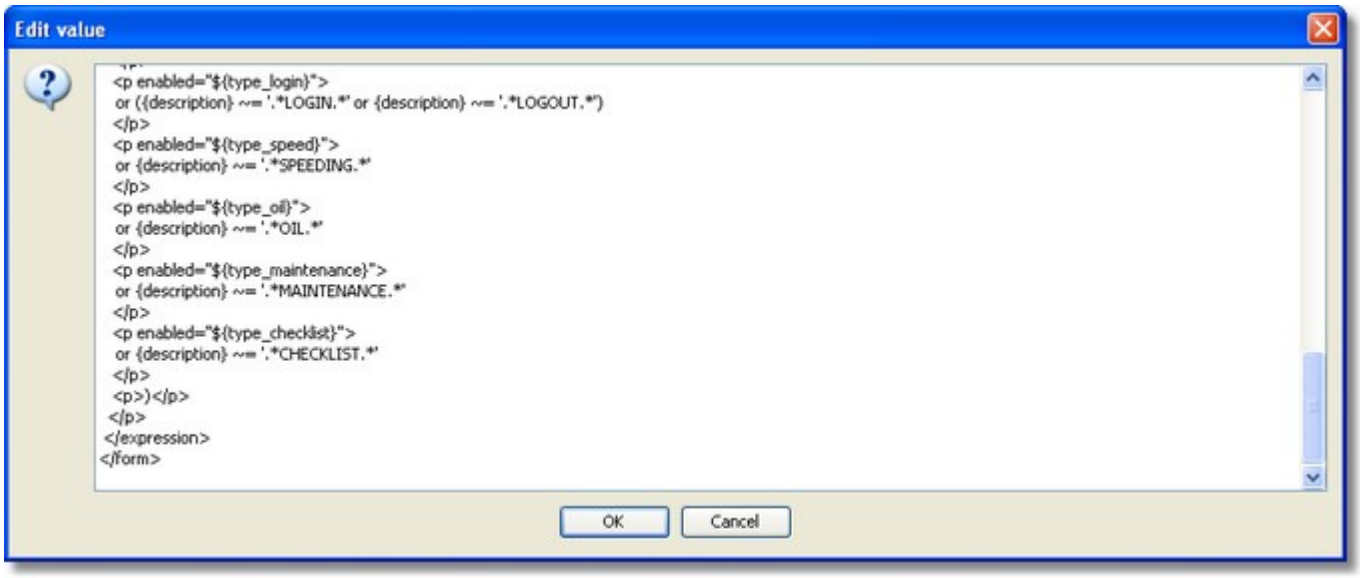
Значения нельзя скопировать из ячеек, которые используют этот редактор.



Щелчок мышью по ссылки инициирует действие со стороны сервера.

ОБЛАСТЬ ТЕКСТА

Значение *String* можно также просмотреть и отредактировать в Редакторе Текстовой области, который отображен в отдельном окне:



ТЕКСТОВЫЙ РЕДАКТОР

Еще одна опция для редактирования значений *String* -- это расширенный Текстовый редактор[408]. Он используется, когда поле *String* содержит код на языке Java, документ XML или другие текстовые данные, подходящие для выделения синтаксических элементов.

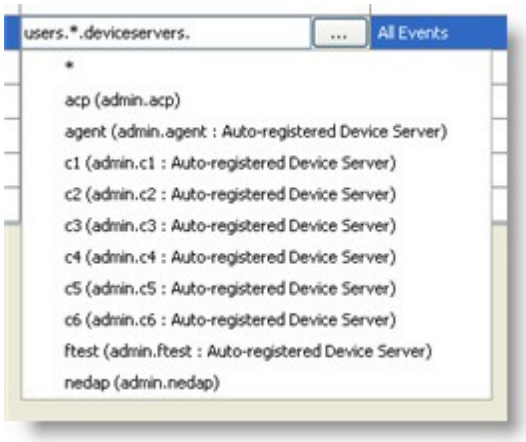
КОДОВЫЙ РЕДАКТОР

Еще одна опция для редактирования значений *Strings* - это Редактор кода 409. Она используется, когда поле *String* содержит исходный код Java, т.е. класс Java, представляющий скрипт сервера 879 или скрипт виджета 1308.

КОНТЕКСТНЫЙ РЕДАКТОР И РЕДАКТОР МАСКИ КОНТЕКСТА

Значения *типа String,* относящиеся к пути Контекста 41 или маскам 44 задаются через компонент элемента **Выбор Маски Контекста**. Он выглядит, как обычное текстовое поле, с расположенной справа кнопкой:

Во время редактирования пути/маски, можно нажать *пробел,* чтобы получить список контекстов, которые можно добавить к редактируемой в настоящей момент маске. Выберите контекст из раскрывающегося списка, чтобы добавить его к текущей маске:



Если нажать кнопку **[...],** откроется компонент Селектора логического объекта 402, который позволяет выстраивать маску, теперь ее можно выделять и активизировать щелчком мыши.

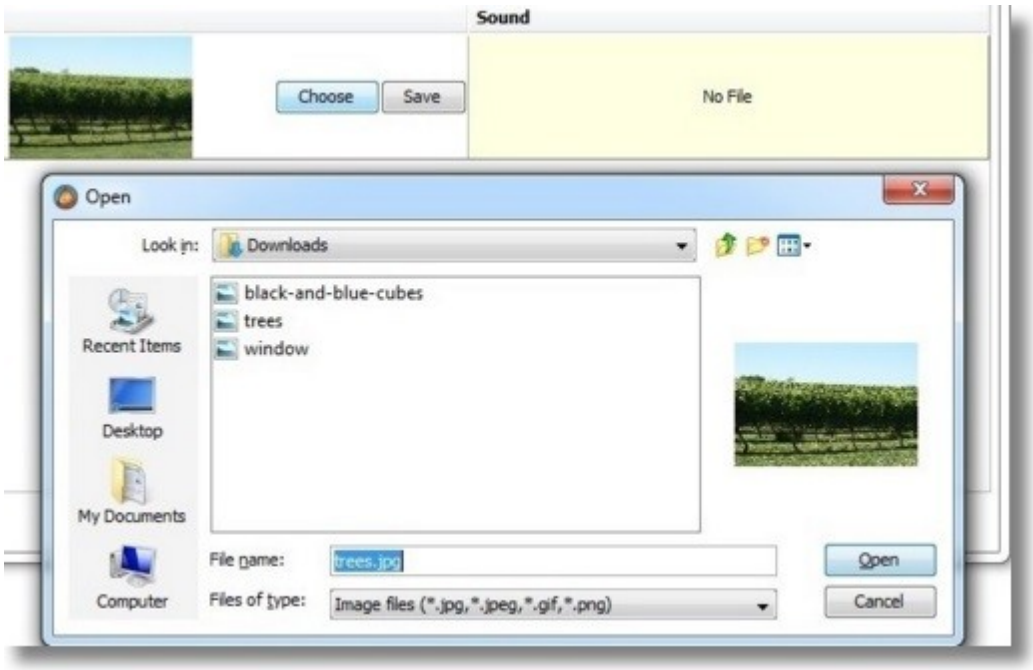
РЕДАКТОР ВЫРАЖЕНИЙ

1. Выражения 112 редактируются в текстовом поле посредством кнопки **[...],** которая открывает Редактор выражений 404, позволяющего создавать и подтверждать правильность выражения в режиме point-and-click.

РЕДАКТОРЫ ЗВУКА/ИЗОБРАЖЕНИЙ/ФАЙЛОВ

Поля *Data Block* могут содержать изображения, звуки или универсальные файлы.

Файлы, звуки и изображения вставляют в ячейки при помощи диалогового окна открытия файла:



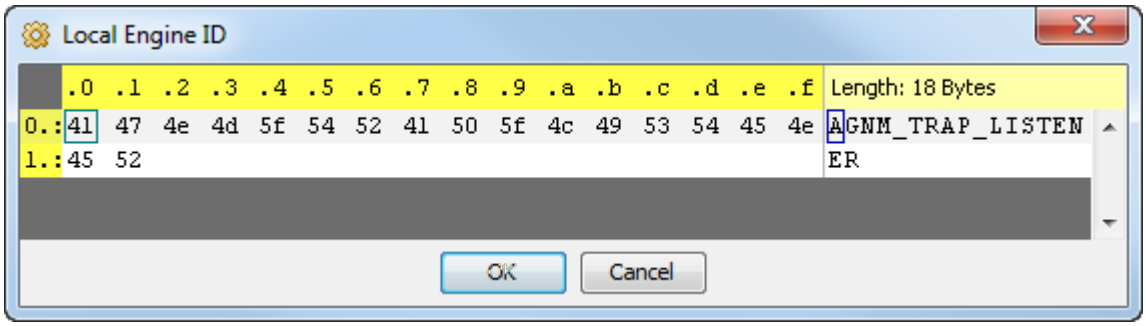
Изображения, содержащиеся в Таблице данных, отображаются в виде уменьшенного изображения (см. предыдущий скриншот). Если кликнуть мышью по уменьшенному изображению, оно раскроется до полного размера.

Векторные изображения (в формате SVG) могут также открываться для редактирования в текстовой редакторе с выделением синтаксиса XML.

Звуки можно воспроизвести непосредственно из ячейки таблицы:

РЕДАКТОР БИНАРНЫХ ДАННЫХ

Обычные бинарные данные редактируются в специальном редакторе, который позволяет задавать байты в форме HEX или ACSII:



8.10.5 Контекстное меню

Редактор таблиц данных предоставляет два разных контекстных меню:

- Контекстное меню ячейки для управления данными таблицы
- Контекстное меню заголовка для контроля видимости и группирования ряда/колонки

Контекстное меню ячейки

Контекстное меню ячеек Редактора таблиц данных включает в себя следующие операции:

Копировать	Копировать значение ячейки.
Вставить	Вставить значение из буфера обмена в ячейку. Если значение в буфере обмена имеет другой формат, отличный от значения ячейки, AtomMind Client попробует его преобразовать. Если значение невозможно преобразовать, оно не будет вставлено.
Скопировать записи	Скопировать выбранные ряды в буфер обмена.
Вставить записи	Вставить скопированные записи после выбранного ряда.
Удалить значение	Присваивает значению ячейки значение NULL (<нет значения>). Эта операция выполняется лишь в том случае, если текущее поле поддерживает отсутствие значения.
Отменить изменения в ячейках	Сбрасывает значение ячейки до одного, которое она содержит, когда открыт Редактор Таблиц данных. Если ячейка не существовала, когда был открыт редактор (т.е. ячейка принадлежит к только что созданному ряду), эта операция сменит значение ячейки на значение по умолчанию (см. ниже).
Изменить значения ячеек на значения по умолчанию	Сменить значение ячейки на значение по умолчанию для этой колонки (обычно заданной сервером, как часть описания формата таблицы).

Заполнить колонку значением из ячейки	Копирует значение из выбранной ячейки в другие ячейки текущей колонки.
Дублировать запись	Создает копию текущей записи.
Отменить изменения в таблице	Отменяет все изменения в таблице, выполненные с момента ее открытия.
Просмотреть формат таблицы	Показывает формат 49 текущей таблицы.

ДЕЙСТВИЯ С ПЕРЕМЕННЫМИ

Когда Редактор Таблиц Данных используется для изменения значения переменной 61 контекста (в отличие от сырых данных, поступающих их другого источника), контекстное меню в редакторе содержит дополнительные действия, относящиеся к этой переменной 102.

Контекстное меню заголовка

Контекстное меню заголовка в Редакторе Таблиц Данных предоставляет доступ к следующим свойствам:

Автоматически изменить размер колонки	Автоматически изменяет размер выбранной колонки так, чтобы в нее вмещались все соответствующие данные.
Автоматически изменить размер всех колонок	Автоматически изменяет размер всех колонок с тем, чтобы в них вмещался максимальный объем данных.
Сгруппировать эту колонку (в порядке возрастания)	Разрешает группирование рядов в порядке возрастания значений текущей колонки.
Сгруппировать эту колонку (в порядке убывания)	Разрешает группирование рядов в порядке убывания значений текущей колонки.
Раскрыть все	Раскрывает все группы рядов.
Свернуть все	Сворачивает все группы рядов.
Скрыть эту колонку	Скрывает текущую колонку.
Показать все скрытые колонки	Показывает все невидимые в настоящий момент колонки.
Установить для колонок значение по умолчанию	Изменяет значение ширины колонок на значение по умолчанию.
Еще...	Открывает всплывающее диалоговое окно для выбора видимых колонок.
Выбрать все ячейки	Выделяет всю таблицу.
Отменить выбор	Отменяет выбор ячеек.

Кроме того, контекстное меню заголовка содержит проверяемые элементы меню, которые позволяют отображать/скрывать отдельные колонки.

8.10.6 Сортировка и фильтрация

Сортировка и фильтрация затрагивают лишь визуальное представление данных, они не изменяют и не реорганизуют данные в таблице.

Сортировка рядов

Вы можете отсортировать ряды редактируемой или просматриваемой Таблицы Данных. Чтобы включить сортировку, кликните мышью по заголовку колонки. Щелчок мышью по заголовку, уже использованному для сортировки, изменит порядок на обратный.

Для разбиения таблицы на несколько столбцов, кликните по ним согласно желаемому порядку разбиения, удерживая клавишу Ctrl.

СОРТИРОВКА СПИСКА ОБЪЕКТОВ

Если список объектов используется как источник данных 384 для редактора, правила сортировки обновляют сортировку всего списка объектов, а не только записей, отображенных в данный момент, поэтому пересортировка может полностью изменить видимые в данный момент записи.

Фильтрация рядов

Существует два способа фильтрации данных в Редакторе Таблиц Данных:

ИСПОЛЬЗОВАНИЕ ПОЛЯ ФИЛЬТРАЦИИ ТЕКСТА

Введите текст в текстовое поле фильтра, расположенное на инструментальной панели Редактора Таблиц Данных (оно отмечено иконкой увеличительное стекло). Содержимое таблицы будет отфильтровано так, чтобы отображать лишь те ряды, которые соответствуют критерию фильтра.

Контекстное меню панели фильтра содержит следующие операции:

Выбор колонки	Позволяет выполнять поиск во всех колонках или только в выбранных.
Чувствительный к регистру	Задает поиск, чувствительный к регистру.
Нечувствительный к регистру	Задает поиск, нечувствительный к регистру.
Использовать групповые символы	Включает/отключает режим поиска групповых символов. Групповые символы могут включать в себя такие символы как * или ?, которые соответствуют любому (-ым) символу(-ам).
Использовать регулярные выражения	Включает/отключает режим поиска регулярные выражений 2142.
Искать сначала	Текст фильтра должен появиться в начале текста ячейки.
Точное совпадение	Текст фильтра должен точно соответствовать тексту ячейки.
Искать везде	Текст фильтра может появиться в любом месте текста ячейки.

ИСПОЛЬЗОВАНИЕ БЫСТРОГО ФИЛЬТРА

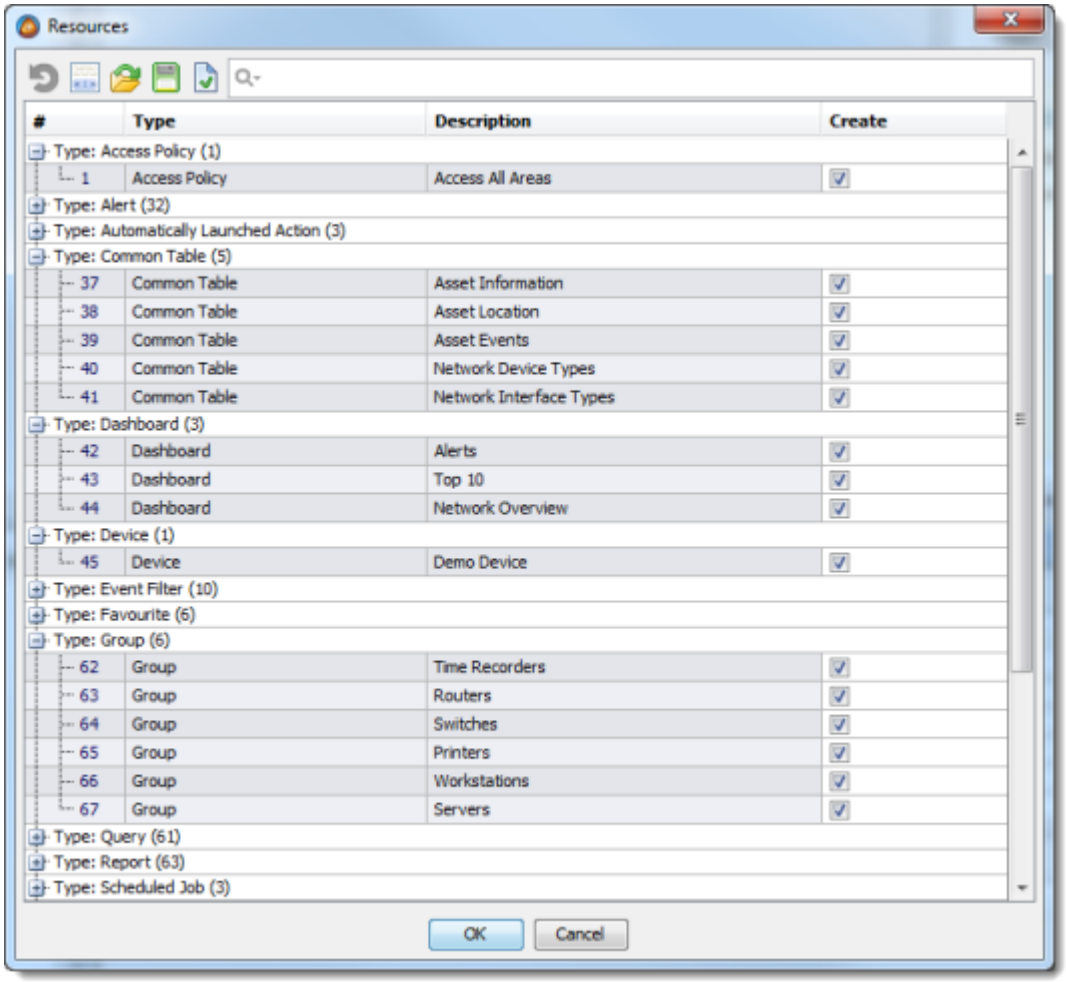
Наведите курсор мыши на ячейку с заголовком столбца. В этой ячейке появится иконка Быстрый Фильтр. Кликните по ней и выберите значения поля, которые нужно отобразить. Записи с другими значениями поля будут отфильтрованы.

ФИЛЬТРАЦИЯ СПИСКА ОБЪЕКТОВ

Если список объектов используется как источник данных 384 для редактора, правила фильтрации обновляют сортировку всего списка объектов, а не только записей, отображенных в данный момент, поэтому новые правила сортировки могут полностью изменить видимые в данный момент записи.

8.10.7 Группировка рядов

Контекстное меню заголовка 392 позволяет группировать ряды по значениям выбранного ряда (-ов). Сгруппированная таблица выглядит следующим образом:



Группирование оказывается эффективным при выводе на экран длинных таблиц.

8.10.8 Импорт/экспорт данных

Редактор Таблиц Данных имеет внедренную поддержку для экспорта/импорта данных для ряда распространенных форматов файлов. Ниже представлен список поддерживаемых форматов:

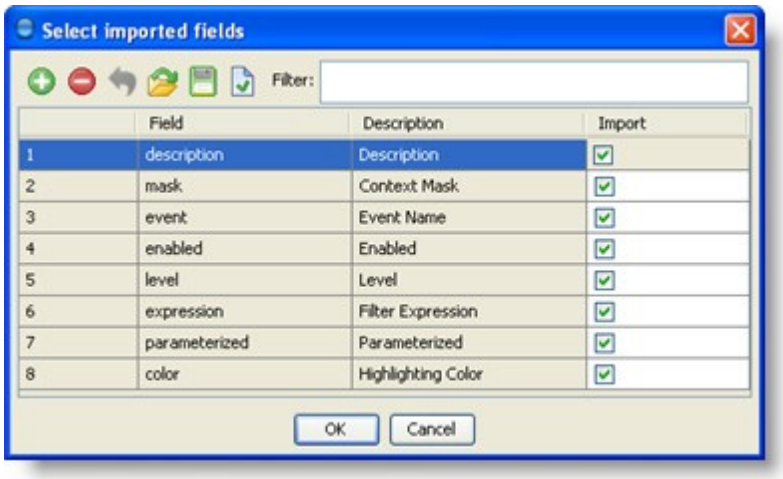
Расширение	Описание	Поддержка экспорта	Поддержка импорта	Примечания
TBL	Свой формат AtomMind	ДА	ДА	Таблица Данных кодируется по стандарту Кодирования таблиц данных 2123 и записывается в файл.
XML	Расширяемый язык разметки	ДА	ДА	Таблица Данных кодируется по стандарту Кодирования Таблиц Данных в XML 2130 и записывается в файл.
HTML	Язык разметки гипертекста	ДА	НЕТ	См. пример ниже.
CSV	Значения, разделенные символами	ДА	ДА	Пользователь может выбрать ряд настроек для кодирования/декодирования с расширением CSV до начала экспорта/импорта.
XLS	Microsoft Excel	ДА	НЕТ	

Во время импортирования данные, прочитанные из файла, не вводятся в исходном виде в Редактор Таблиц Данных. Они сливаются с данными, содержащимися в настоящий момент в Редакторе Таблиц Данных, при выполнении операции Умная Копия Таблицы Данных.

Процедура импорта свойств не может просто переписать значения, содержащиеся в редакторе, на прочтенные из файла Excel потому, что могут иметь разный формат[49]. В любом случае, система старается импортировать как можно большее количество данных.

Значения полей, предназначенных только для чтения, не импортируются. Поэтому перед импортом вам понадобится удалить записи с полями, предназначенными только для чтения, которые должны быть модифицированы.

Во время импорта данных можно выбрать, какие поля будут импортироваться. Можно импортировать лишь те поля, которые находятся в импортируемой Таблице Данных или редактируемой в настоящий момент Таблице данных, и их можно записать в редактируемую таблицу:



Если Вы выполняете импорт/экспорт в/из файл(-а) CSV, система предложит Вам выбрать опции кодирования/декодирования CSV[216].

Примеры

Таблица исходных данных:

	Description	Context Mask	Event Name	Enabled	Level	Filter Expression	Parameterized	Highlighting Color
1	Administration Events	administration	Information (info)	☑	Info		☐	<Not set>
2	AuthKey Events	authkey	Information (info)	☑	Info		☐	<Not set>
3	User Events	users.*	Information (info)	☑	Info		☐	<Not set>
4	User Login Events	users	User login (login)	☑	Info		☐	<Not set>
5	User Logout Events	users	User logout (logout)	☑	Info		☐	<Not set>
6	Device Servers Events	users.*.deviceservers	Information (info)	☑	Info		☐	<Not set>
7	Device Server Events	users.*.deviceservers.*	Information (info)	☑	Info		☐	<Not set>
8	Device Informational Ev...	users.*.deviceservers.*.devices.*	Information (info)	☑	Info		☐	<Not set>
9	Device Internal Events	users.*.deviceservers.*.devices.*	Device event (event)	☑	Info		☐	<Not set>
10	Alert Notifications	users.*.alerts.*	Alert (alert)	☑	Info		☐	<Not set>
11	Alert Events	users.*.alerts.*	Information (info)	☑	Info		☐	<Not set>
12	Scheduler Events	users.*.jobs.*	Information (info)	☑	Info		☐	<Not set>

Аналогичная таблица, экспортированная в формат CSV (Запись заголовка содержит описания поля. Скриншоты сделаны из текстового редактора):

data.csv

```
Description;Context Mask;Event Name;Enabled;Level;Filter Expression;Parameterized;Highlighting Color
Administration Events;administration;Information (info);1;Info;;0;NULL
AuthKey Events;authkey;Information (info);1;Info;;0;NULL
User Events;users.*;Information (info);1;Info;;0;NULL
User Login Events;users;User login (login);1;Info;;0;NULL
User Logout Events;users;User logout (logout);1;Info;;0;NULL
Device Servers Events;users.*.deviceservers;Information (info);1;Info;;0;NULL
Device Server Events;users.*.deviceservers.*;Information (info);1;Info;;0;NULL
Device Informational Events;users.*.deviceservers.*.devices.*;Information (info);1;Info;;0;NULL
Device Internal Events;users.*.deviceservers.*.devices.*;Device event (event);1;Info;;0;NULL
Alert Notifications;users.*.alerts.*;Alert (alert);1;Info;;0;NULL
Alert Events;users.*.alerts.*;Information (info);1;Info;;0;NULL
Scheduler Events;users.*.jobs.*;Information (info);1;Info;;0;NULL
```

Аналогичная таблица, экспортированная в формат HTML (как показано в веб-браузере):

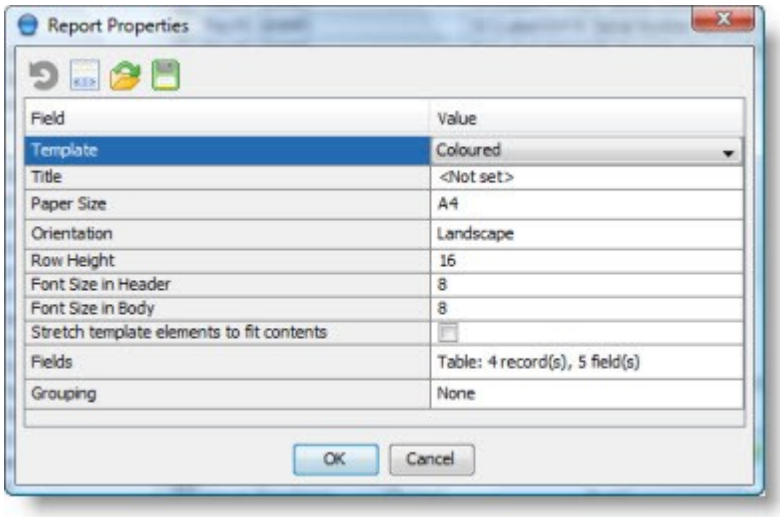
Description	Context Mask	Event Name	Enabled	Level	Filter Expression	Parameterized	Highlighting Color
Administration Events	administration	info	true	Info		false	Null
AuthKey Events	authkey	info	true	Info		false	Null
User Events	users.*	info	true	Info		false	Null
User Login Events	users	login	true	Info		false	Null
User Logout Events	users	logout	true	Info		false	Null
Device Servers Events	users.*.deviceservers	info	true	Info		false	Null
Device Server Events	users.*.deviceservers.*	info	true	Info		false	Null
Device Informational Events	users.*.deviceservers.*.devices.*	info	true	Info		false	Null
Device Internal Events	users.*.deviceservers.*.devices.*	event	true	Info		false	Null
Alert Notifications	users.*.alerts.*	alert	true	Info		false	Null
Alert Events	users.*.alerts.*	info	true	Info		false	Null
Scheduler Events	users.*.jobs.*	info	true	Info		false	Null

Та же самая таблица, экспортированная в формат XLS (как показано в Microsoft Excel):

	A	B	C	D	E	F	G	H
1	description	mask	event	enabled	level	expression	parameterized	color
2	*Description*	*Context Mask*	*Event Name*	*Enabled*	*Level*	*Filter Expression*	*Parameterized*	*Highlighting Color*
3	Administration Events	administration	Information (info)	1	Info		0	NULL
4	AuthKey Events	authkey	Information (info)	1	Info		0	NULL
5	User Events	users.*	Information (info)	1	Info		0	NULL
6	User Login Events	users	User login (login)	1	Info		0	NULL
7	User Logout Events	users	User logout (logout)	1	Info		0	NULL
8	Device Servers Events	users.*.deviceservers	Information (info)	1	Info		0	NULL
9	Device Server Events	users.*.deviceservers.*	Information (info)	1	Info		0	NULL
10	Device Informational Events	users.*.deviceservers.*.device:	Information (info)	1	Info		0	NULL
11	Device Internal Events	users.*.deviceservers.*.device:	Device event (event)	1	Info		0	NULL
12	Alert Notifications	users.*.alerts.*	Alert (alert)	1	Info		0	NULL
13	Alert Events	users.*.alerts.*	Information (info)	1	Info		0	NULL
14	Scheduler Events	users.*.jobs.*	Information (info)	1	Info		0	NULL
15								

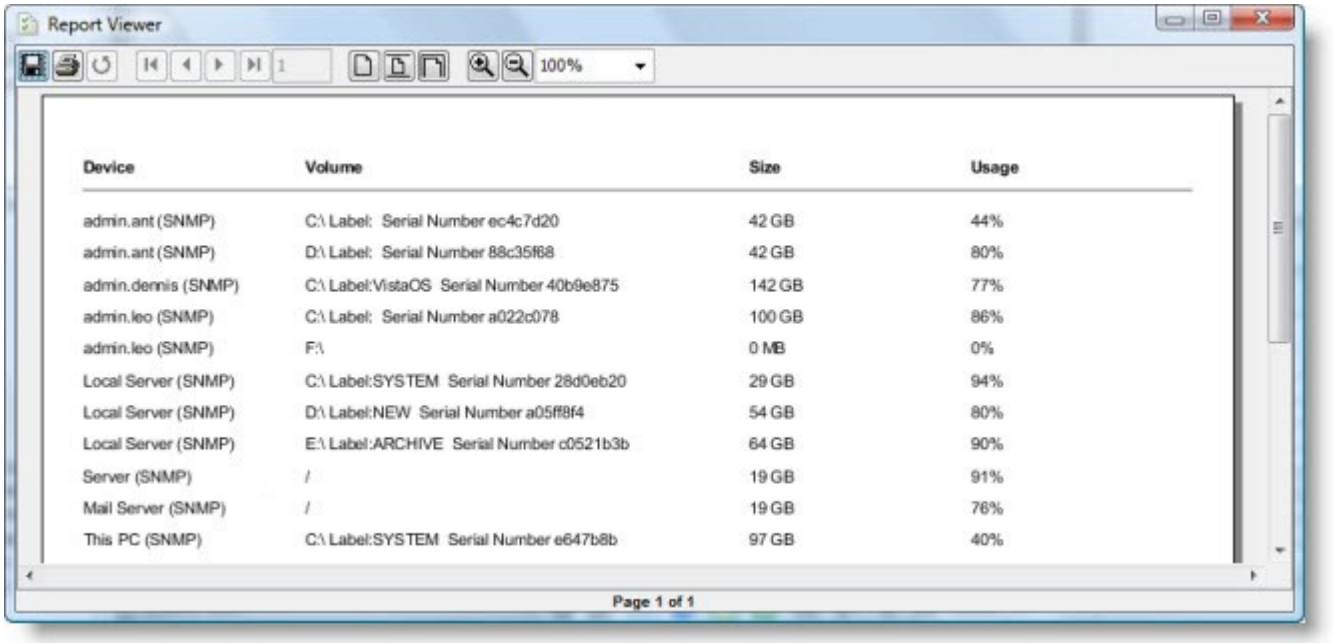
8.10.9 Создание отчета

Когда редактор Таблиц Данных находится в Режиме Множественных записей, можно создать отчет о просмотренных или отредактированных данных. Кликните мышью по кнопке **Создать Отчет**, расположенной на панели инструментов и выберите опции 930 отчета:



Если Вы кликните **OK**, AtomMind Client создает 930 конструктор отчета, заполняет его данными из редактора и открывает созданный отчет в Просмотрщике отчетов 415. Теперь отчет можно распечатать или экпортировать в PDF, HTML, CSV и пр.

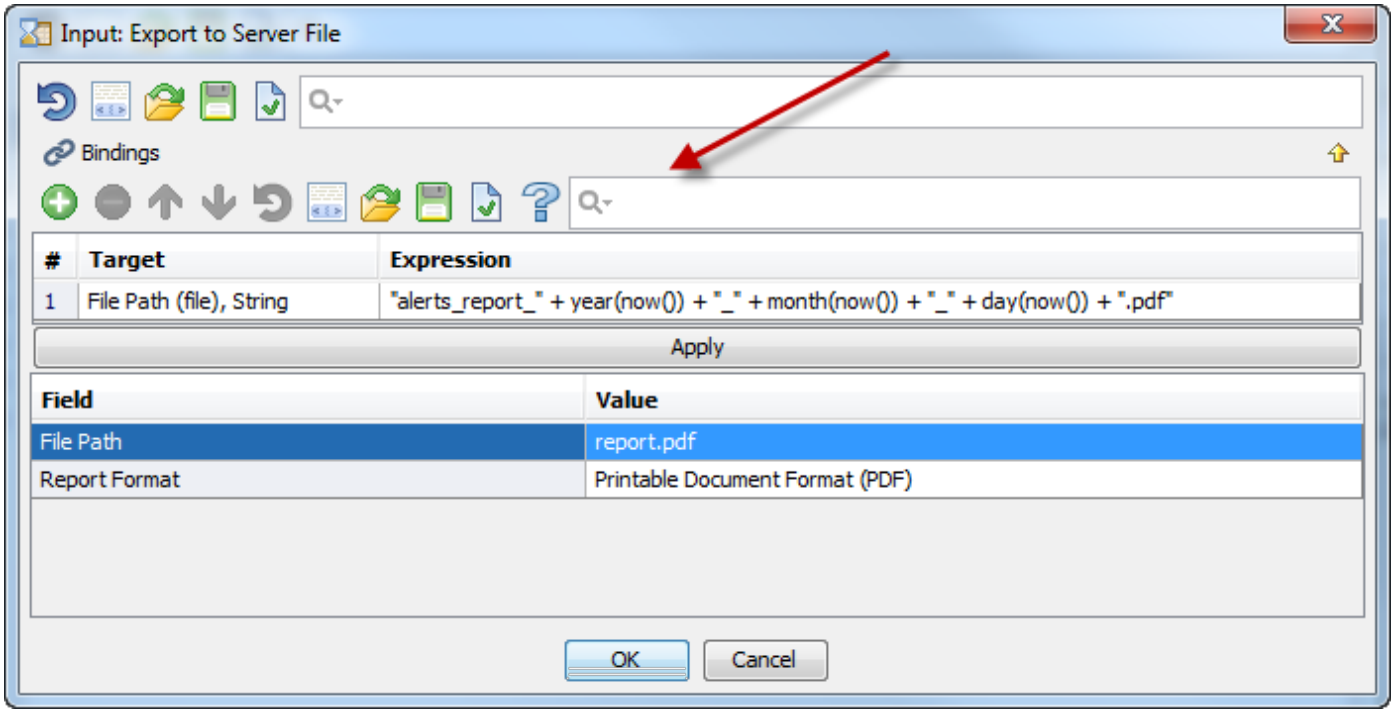
Отчет выглядит следующим образом:



8.10.10 Редактирование привязок

В некоторых случаях Редактор Таблиц Данных позволяет добавлять, редактировать или убирать привязки 741 редактируемой таблицы. Например, этот режим включется, когда таблица используется как параметр ввода для автономного действия 99 (т.е. действие, выполняемое при возникновении тревоги 779 или при помощи планировщика 823).

Если включено редактирование привязок, верхний раздел Редактора Таблиц Данных будет содержать раскрывающуюся часть привязок:



Привязки редактируются во вложенном Редакторе Таблиц Данных с двумя полями:

- **Цель**. Определяет поле (и опционально номер строки), куда записывается выражение.
- **Выражение**. Определяет выражение 112 привязки, которое рассчитывается при обработке привязок таблицы.

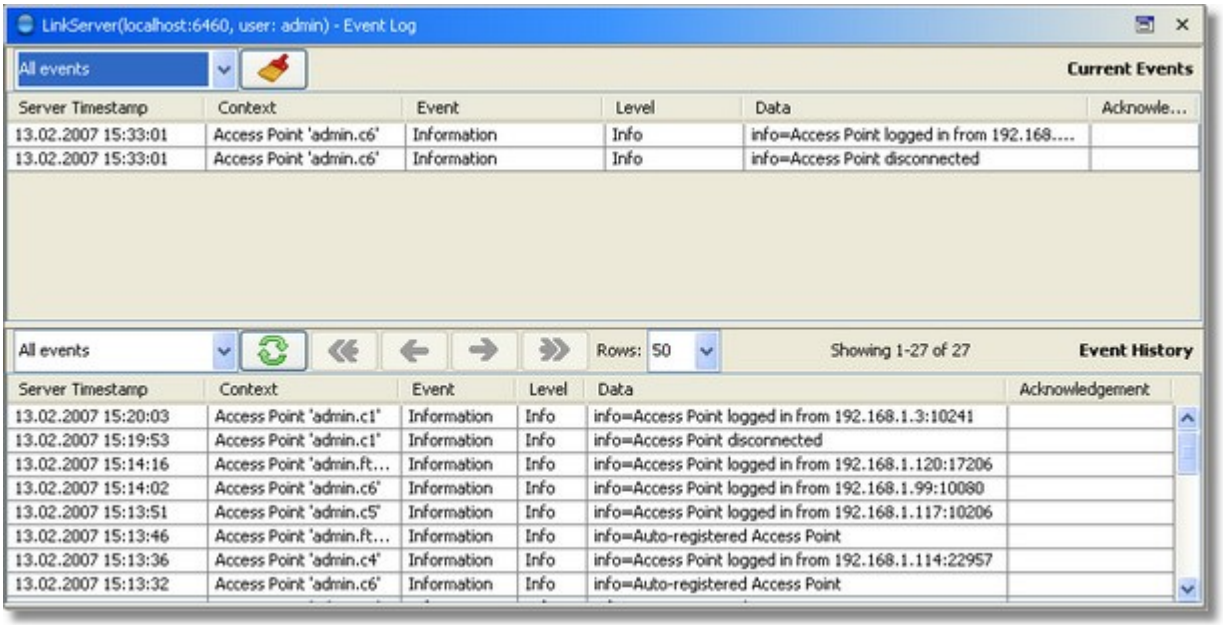
Кнопка **Применить** дает команду Редактору Таблиц Данных сохранить новые/измененные привязки в "начальной" таблице и пересчитать табличные данные в соответствии с новыми привязками.

8.11 Журнал событий

Компонент *Журнал событий* предназначен для просмотра и управления событиями 73 сервера в режиме реального времени. Он также предоставляет доступ к истории событий 73 и позволяет подтверждать 73 события.

Чтобы позволить использование Фильтров событий 762, AtomMind Client создает один журнал событий для каждого соединения с сервером 363, когда оно активно (установлено). Компонент журнала событий автоматически закрывается, когда разрывается соединение с сервером.

Окно журнала событий состоит из двух основных частей: *Текущие события* и *История событий:*



У обеих частей имеется панель инструментов с часто используемыми функциями. С левой стороны обеих панелей инструментов находится раскрывающийся список, который позволяет выбрать Фильтр событий. В начале работы, выбран фильтр <Пустой фильтр>, это означает, что события не отображаются.

В некоторых случаях компонент Журнал регистрации событий показывает заранее определенный набор типов событий. В этом случае список Фильтров Событий и соответствующих кнопок ("Возобновить Фильтр", "Настроить Фильтр") не доступны.

Размер колонок таблиц событий можно изменить, а сами колонки реорганизовать. Чтобы изменить размер колонки, кликните по ее полю и перетащите мышкой влево или вправо. Чтобы изменить порядок колонок, перетащите заголовок колонки в новое место. Размер колонки и порядок можно изменять лишь только у текущего фильтра. Он хранится на рабочем пространстве и используется вновь после перезапуска AtomMind Client.

Также можно сортировать события. Чтобы включить сортировку, кликните по заголовку колонки. Повторный щелчок мышью изменит порядок сортировки для этой колонки, в порядке возрастания или убывания.

Двойной щелчок мышью по разделительной линии между заголовками двух колонок автоматически скорректирует ширину колонки слева (т.е. размер колонки будет изменен с тем, чтобы она вмещала в себя содержащиеся данные).

Обратите внимание, что сортировка **истории событий** выполняется на сервере, поэтому для ее выполнения может потребоваться значительное количество времени.

Двойной щелчок мышью по любому событию открывает таблицу данных 49, относящуюся к Редактору таблицы данных 382.

Журнал регистрации событий использует настройки отображения **Даты**, **Времени** и **Временных поясов**, заданных в Свойствах учетной записи пользователя 481, для правильного конвертирования и отображения временных меток.

Все значения даты/времени отображаются в текущем для пользователя временном поясе.

Отображение работы фильтров журнала событий с большим количеством данных может быть причиной высокого использования памяти AtomMind Client.

Индикация уровня событий

Столбец уровня события использует набор иконок для индикации уровня 75 событий:

Icon	Event Level
	Нет (уровень события не определен)
	Уведомление
	Инфорация
	Предупреждение
	Ошибка
	Критическое событие

8.11.1 Текущие события

Таблица текущих событий отображает события, происходящие на сервере в режиме реального времени. События фильтруются согласно выбранному Фильтру событий 762 или обычным правилам фильтрования, заданным сервером во время выполнения процедуры Показать журнал событий 96. Чтобы минимизировать объем передаваемых данных, фильтрация производится на сервере.

Количество событий, которые могут быть показаны, находится в пределах до 1000. Если количество событий превышает этот лимит, самое старое событие будет удалено, в то время как новое будет добавлено.

Панель инструментов таблицы текущих событий

В раскрывающемся списке можно выбрать Фильтр Событий 762, который будет использоваться для фильтрации событий, которые будут использованы в журнале регистрации событий.

Реактивировать текущий фильтр. Эта операция позволяет заново ввести параметры фильтра.

Настроить текущий фильтр. Редактирует настройки выбранного фильтра.

Включить/Отключить вертикальную автопрокрутку. Если включено, журнал будет автоматически пролистываться и отображать самые последние события.

Включить/Отключить горизонтальную прокрутку. Если отключено, ширина всех колонок автоматически подстраивается под ширину окна Журнала событий. Если включено, у Журнала событий будет горизонтальная прокрутка, а ширина всех колонок будет выстроена так, чтобы вмещать все содержащиеся в них данные. По умолчанию эта опция отключена.

Просмотреть статистику уровней. Показывает количество событий в каждом уровне 73 в окне Текущих Событий.

Экспорт событий. Экспортирует 402 в файл события, отображаемые в текущий момент в окне Текущих событий.

Очистить список событий. Удаление таблицы с текущими событиями не удаляет эти события с сервера и не предотвращает появление в таблице новых событий.

8.11.2 История событий

История событий отображает события, которые произошли в прошлом и сохранены на сервере. События фильтруются согласно выбранному Фильтру событий 762 или пользовательским правилам фильтрации, заданными сервером во время выполнения процедуры Показать журнал событий 96.

Не все события сохраняются в истории. Таким образом, событие, которое появилось в разделе Текущие События, не обязательно появится в Истории Событий.

Панель инструментов Истории Событий

Фильтр событий 762, выбранный в раскрывающемся списке, используется для фильтрации событий, которые будут отражены.

Редактировать текущий фильтр. Эта операция позволяет заново ввести параметры фильтрации.

Настроить текущий фильтр. Позволяет редактировать настройки используемого в настоящий момент фильтра.

Включить/отключить горизонтальную прокрутку. Если отключено, ширина всех колонок автоматически подстраивается под ширину окна Журнала событий. Если включено, у Журнала событий будет горизонтальная прокрутка, а ширина всех колонок будет выстроена так, чтобы вмещать все содержащиеся в них данные. По умолчанию эта опция отключена.

Просмотреть статистику уровней. Показывает количество событий каждого уровня 73 в загруженной на настоящий момент истории событий.

Экспорт. Экспортирует 402 события, отображенные в настоящий момент в окне Истории событий, в файл.

Будут экспортироваться лишь те события, которые в настоящий момент выбраны. Чтобы экспортировать большее количество событий, следует увеличить значение в комбинированном списке событий (см ниже).

Обновить. Обновляет историю событий. Эта операция заставляет сервер перегрузить историю, используя выбранный в настоящий момент фильтр. То, какие события отражены, может поменяться после этой операции..

Первый. Перелистывает список событий до первой страницы.

Предыдущий. Загружает предыдущую страницу списка истории.

Следующий. Загружает следующую страницу списка истории.

Последний. Перелистывает историю до последней страницы.

Комбинированный список **событий**, располагающийся по центру панели инструментов, определяет, сколько событий должно отображаться за один раз.

Общее количество событий, относящееся к фильтру и ряду отображаемых событий, указано в правой части панели инструментов.

8.11.3 Контекстное меню

Контекстное меню открывается щелчком правой кнопки мыши по одному из событий Журнала событий. Оно включает в себя следующие элементы:

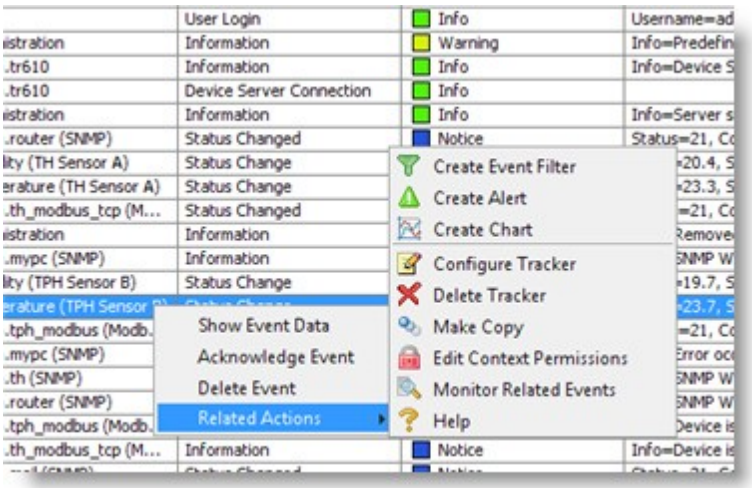
- **Показать данные события**. Открывает Таблицу данных 49, связанную с событием, в Редакторе таблиц данных 382.
- **Подтвердить событие**. Позволяет пользователю установить подтверждение события. Эта операция доступна только для постоянных событий.
- **Просмотреть подтверждения**. Позволяет просмотреть подтверждения в Редакторе таблиц данных 382.
- **Просмотреть обогащения**. Показывает обогащения 76 событий в редакторе таблицы данных 382. Эта операция доступна только для постоянных событий.
- **Удалить событие**. Позволяет удалить событие из истории событий. Эта операция доступна только для постоянных событий.

Компоненты меню событий

Контекстное меню событий также содержит подменю Связанные действия со списком действий 87, относящихся к контексту 41, где событие происходило или же действиям, относящимся непосредственно к событию 102.

Например, для события, относящегося к Device, могут быть отображены такие операции, как "перегрузить Device", "настроить Device" и др.

Пример меню контекста события:



8.11.4 Подтверждение событий

Функция подтверждения 73 события доступна лишь для постоянных событий. Когда активировано **подтверждение элемента меню событий**, система предлагает пользователю ввести текст подтверждения:

Когда подтверждение установлено, время, автор и текст этого подтвержения появятся в колонке подтверждений в таблице событий:

8.11.5 Экспорт событий

У журнала событий есть поддержка экспорта событий в распространенные форматы файлов. Во время экспорта событий в первую очередь подготавливается Таблица данных 49, содержащая данные событий. Эта таблица содержит следующие поля:

- Время события
- Контекст события
- Имя события
- Уровень события
- Данные события (содержит таблицу данных, закодированных в виде строки)
- Одно дополнительное поле для каждого поля в заявленном формате события

Ниже представлен список поддерживаемых форматов экспорта:

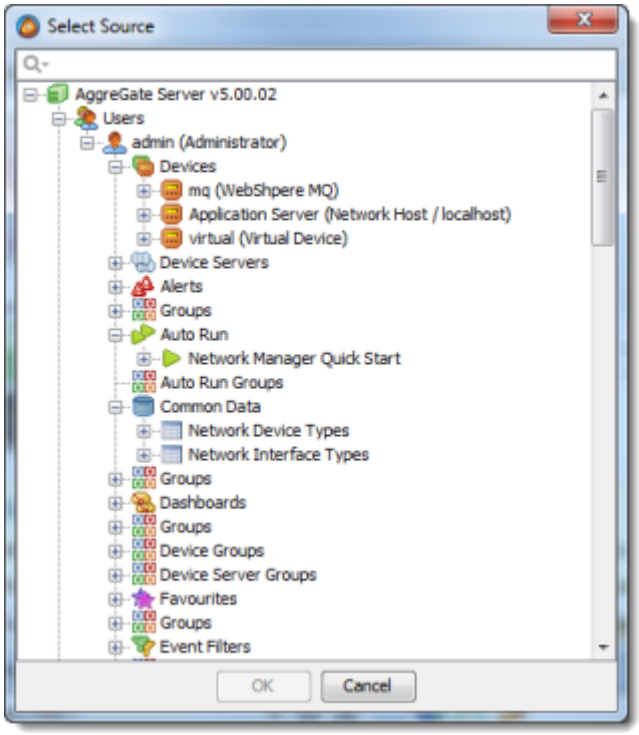
Расширение	Описание	Примечание
TBL	Собственный формат AtomMind	Таблица данных кодируется согласно стандарту Кодирования таблиц данных 2123 и записывается в файл.
XML	Расширяемый язык разметки	Таблица данных кодируется согласно стандарту XML кодирования таблиц данных 2130 и записывается в файл.
HTML	Язык разметки гипертекста	
CSV	Значения разделенные символами	Пользователь может определить ряд CSV кодирующих/декодирующих настроек до начала операции импорта/экспорта.
XLS	Microsoft Excel	

8.12 Селектор объектов

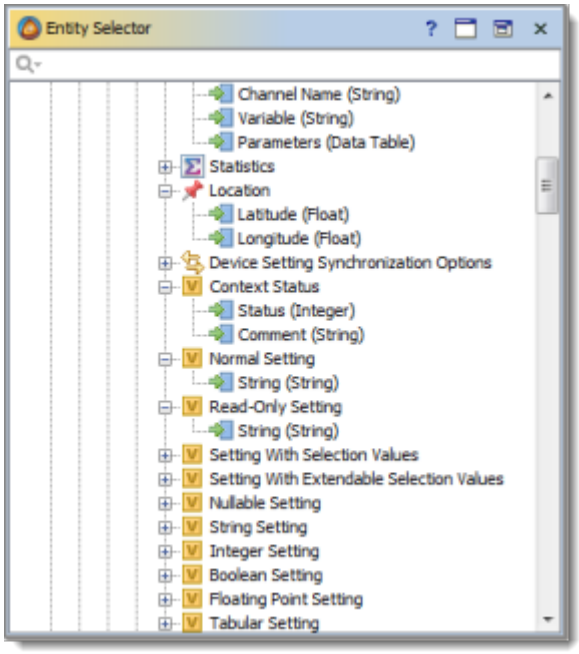
Компонент *Селектор Объектов* используется для выбора контекстов 41, переменных 61, функций 70, событий 73 AtomMind Server и даже соответствующих им **полей**.

Кликните мышью здесь 45, чтобы увидеть разницу между видимым контекстным деревом в Селекторе Объектов и Контекстным деревом на сервере.

Содержимое можно отфильтровать так, чтобы для выбора отображался лишь определенный тип объектов. Вот так выглядит Селектор Объектов, используемый для выбора одного контекста:



Таким образом он выглядит в редакторе виджетов[423], когда разрешен выбор всех типов объектов:



Использование Селектора объектов

1. Селектор объектов используется процедурой Выбрать объекты[94] в GUI-интерфейсе, когда необходимо выбрать различные контексты и их элементы.

2. В Редакторе виджетов[423] Селектор объектов можно использовать для перетаскивания объектов на различные компоненты (виджеты), расположенные в главном окне.

3. Двойной щелчок мышью по узлу дерева Селектора Объектов в Построителе выражений[404] добавляет новую Ссылку[117] в редактируемое Выражение[112].

4. Кроме того, Селектор объектов используется для выбора контекстов и контекстных масок во время редактирования полей пути/маски контекста в Редакторе Таблиц Данных[382].

Иконки в Селекторе Объектов

Если специальные иконки определены для различных контекстов и их частей, они (эти иконки) отображаются в дереве Селектора объектов. Во всех остальных случаях, используются следующие иконки:

	"Закрытый" контекст (для свернутых в текущий момент узлов контекста). Эта иконка отображается, если контекст не предлагает пользовательскую иконку.
	"Открытый" контекст (для развернутых контекстных узлов). Эта иконка отображается, если контекст не предлагает пользовательскую иконку.
	Группа переменных, функций или событий.
	Переменная. Эта иконка отображается, если определение переменной не предлагает пользовательскую иконку.
	Функция. Эта иконка отображается, если определение функции не предлагает пользовательскую иконку.
	Событие. Эта иконка отображается, если определение события не предлагает пользовательскую иконку.
	Действие. Эта иконка отображается, если определение действия не предлагает пользовательскую иконку.
	Поле ввода (для ввода значений для переменных, событий или функций). Отображается описание и тип поля.
	Поля вывода (используется функциями). Отображается описание и тип поля.

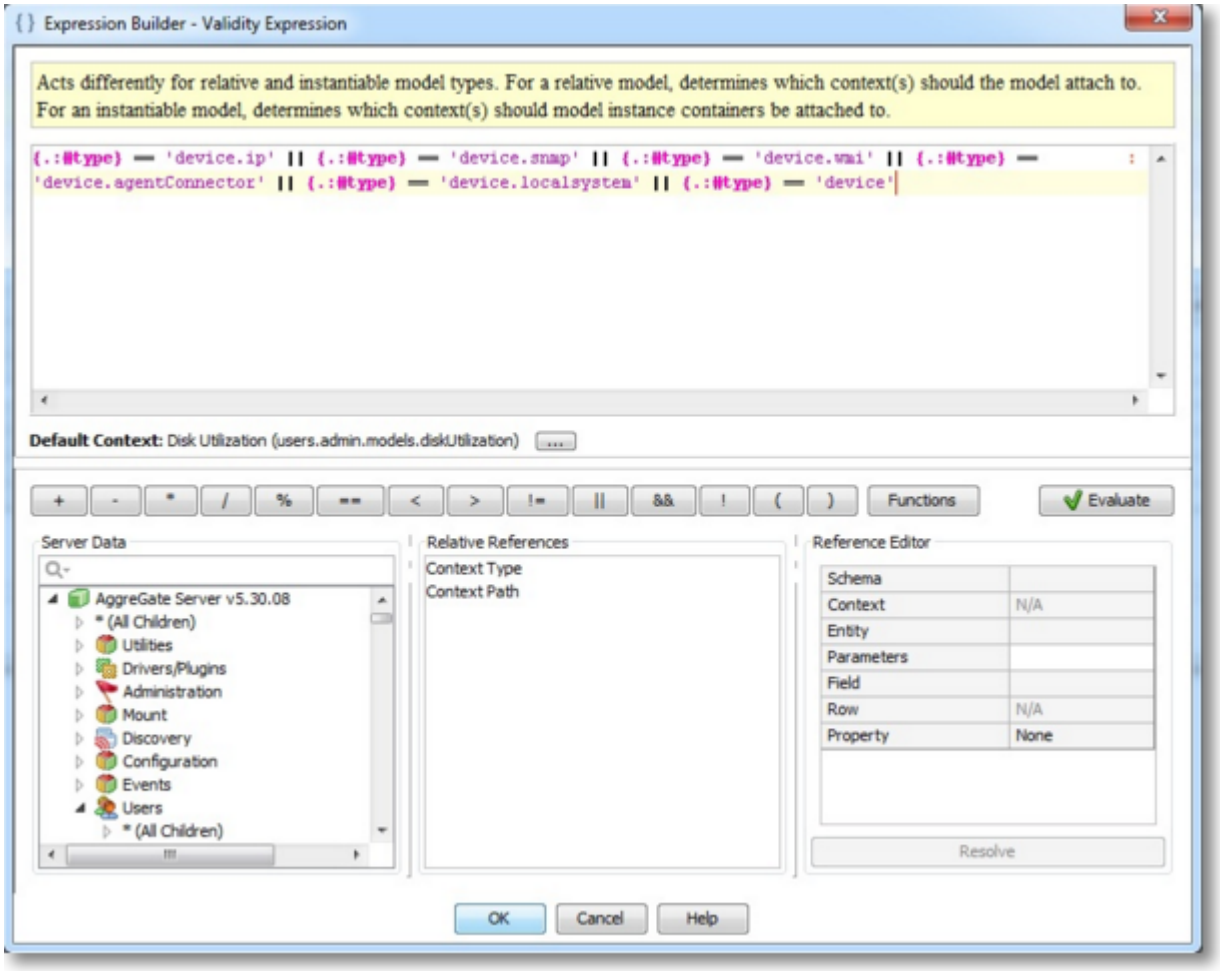
Поиск/Фильтрация узлов

Селектор Объектов имеет наверху поле фильтра. Если что-то вводится в это поле, отображаются только те узлы, чьи названия содержат введенный текст.

8.13 Редактор выражений

Компонент Редактор Выражений был создан, чтобы упростить написание выражений в Языке выражений 112. Этот компонент используется для редактирования текста выражений в Редакторе таблиц данных 382.

Окно Редактора Выражений выглядит следующим образом:



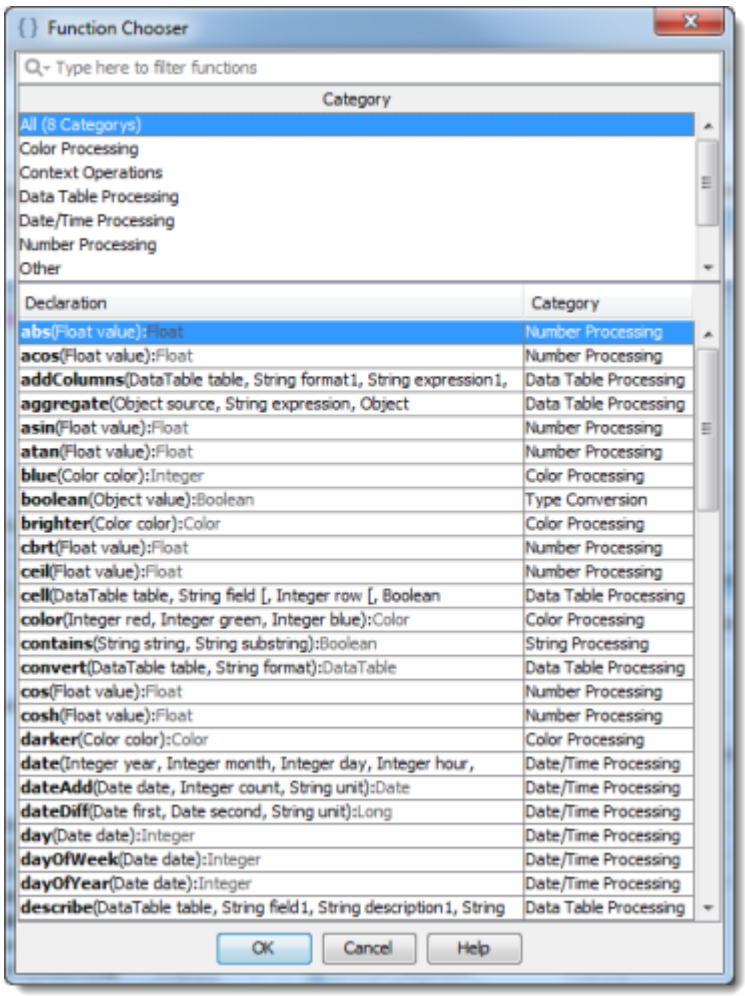
Окно состоит из следующих частей:

- **Область текста справки** сверху отображает суммарную информацию о выражении. Данная панель невидима, если не предоставлен текст справки для выражения.
- **Область текста выражения**, расположенная сверху, используется непосредственно для редактирования текста выражения.
- **Блок информации** отображает имена и описания контекста 41 по умолчанию и таблицы данных 49 по умолчанию. Кнопка в конце строки "Контекст по умолчанию" предоставляет возможность определять случайный контекст по умолчанию.
- **Кнопочная панель** с двумя группами кнопок. Кнопки, расположенные слева, используются для вставки различных операторов и других элементов Языка Выражений в место курсора в области выражения. Кнопки слева используются для проверки синтаксиса выражения (кнопка **Вычислить**), а также для подсчета результата выражений в текущем окружении (кнопка **Выполнить**). Кроме того, есть еще и комбинированная кнопка для быстрой вставки функций 124.
- Область **Данных Сервера**, содержащая компонент Селектор объектов 402. Двойной щелчок по любому из объектов вставляет ссылку 117 на объект в место расположения курсора в области выражения.
- Область **компонентов**, содержащая еще один Селектор Объектов, позволяет вставлять ссылки в свойства компонентов виджета. Этот выбор доступен во время редактирования выражений, связанных с виджетом 1296.
- Список **относительных ссылок** (не отображаемый на выше приведенном скриншоте) показывает доступные в текущей среде ссылки. Если доступных ссылок нет, эта часть не видна. Двойной щелчок мышью по ссылке вставляет ее текстовое представление в место расположения курсора в области выражения.

Относительные ссылки - это ссылки с относительной **контекстной** частью или не имеющие спецификации **контекста/объекта**. См. раздел Ссылки 117.

- **Редактор Ссылок**, используемый для редактирования ссылок, выбран в настоящий момент в области выражения. Любую часть ссылки можно изменить независимо. Когда изменения внесены, текст выражения обновляется. Кнопка **Разрешить** позволяет подсчитать текущее значение, указанное редактируемой ссылкой.

- Кнопка **Вычислить** (можно также вызвать нажатием F5 или Ctrl+Enter) позволяет посчитать выход выражения. Изображение кнопки меняется с зеленой галочки (✓) на красную (✓), если синтаксис всего выражения (или его выбранной части, если такая есть) неправильный.
- Кнопка **Функции**, которая активизирует Выбор Функций, позволяет вставлять функцию[124]:



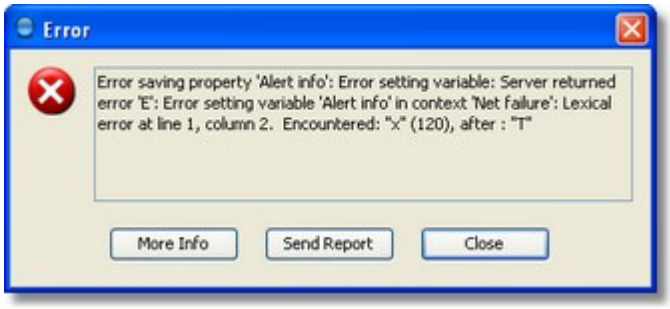
Вычисление выражений в редакторе

Редактор выражений позволяет проверить правильность синтаксиса практически любого выражения. Во многих случаях также возможно рассчитать результат, который произведет выражение. Это можно сделать, поскольку Редактор выражений знает о среде вычисления[114] данного выражения. Однако среда вычисления не всегда полная, и в некоторых случаях невозможно правильно вычислить выражение из Редактора Выражений. Вот несколько примеров:

- При редактировании выражения фильтра событий[762] нет определенного события, предоставляющего данные для выражения.
- То же самое относится к выражению триггер события[787] тревоги[779] - нет такого события, с которым его можно протестировать, и как следствие нет "реальной" таблицы данных по умолчанию в среде вычисления выражения.

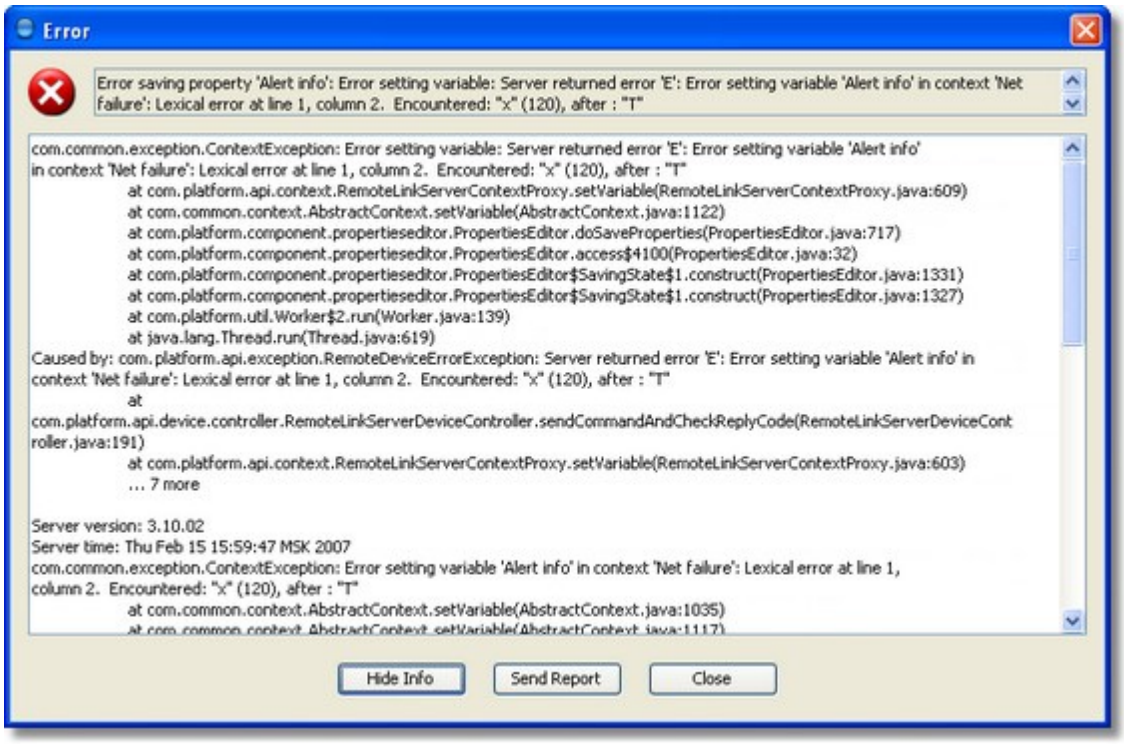
8.14 Диалоговое окно сообщения об ошибке

Диалоговое окно сообщения об ошибке обычно выглядит следующим образом:



У него есть три кнопки:

Кнопка **More Info** (дополнительная информация) позволяет получить подробную инфмормацию о сообщении об ошибке. Опытные пользователи используют её для поиска и устранения неисправности.

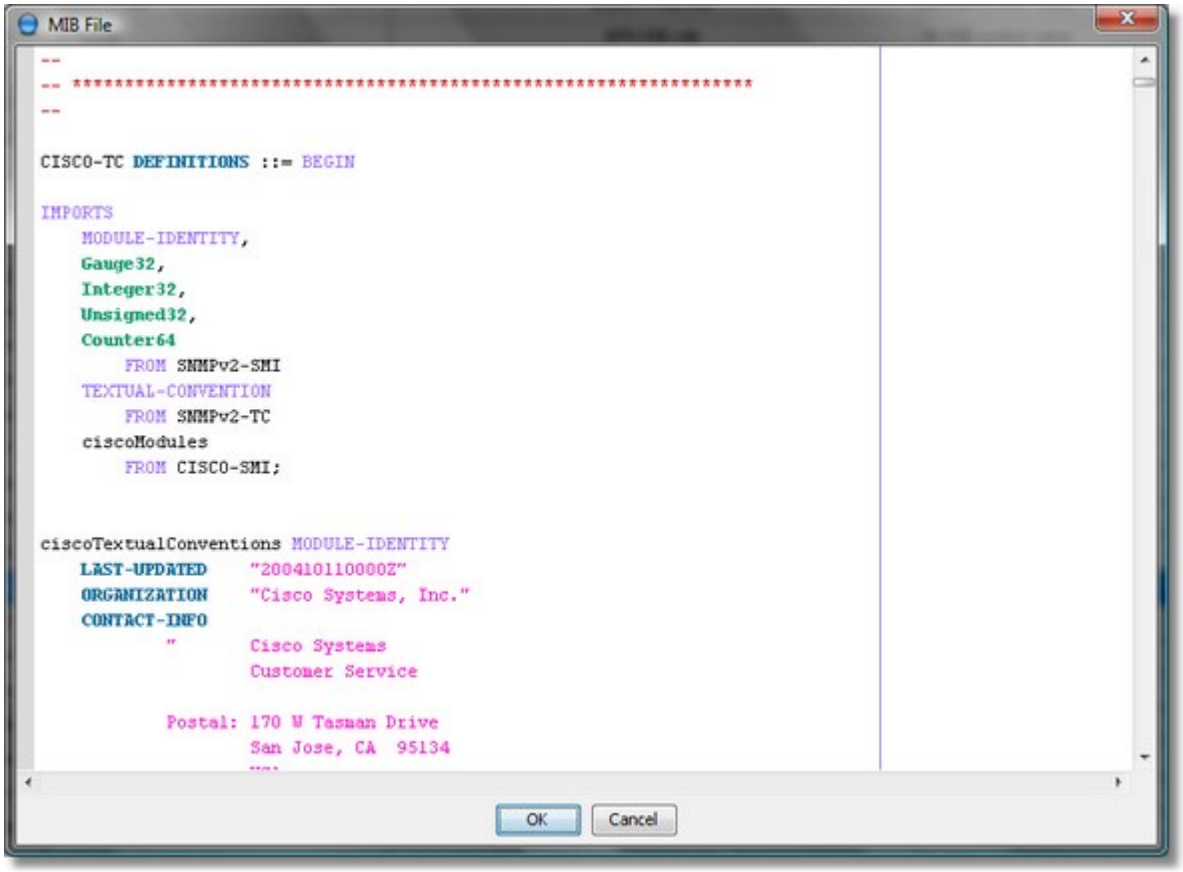


Кнопка **Send Report** (отправить отчет) отправляет информацию об ошибке на ТВЭЛ. Email содержит текст, отображённый в момент, когда Вы кликнули мышью по кнопке **More Info**. Никакая иная информация не передается. Компьютер, на котором установлен AtomMind Client, должен иметь доступ к Internet для того, чтобы отчет можно было отправить. Отчет об ошибке отправляется по протоколу HTTP на сервер ТВЭЛ.

Кнопка **Close** (закрыть) просто закрывает диалоговое окно. Если включена регистрация данных 422, Вы позже можете обнаружить сообщение об ошибке (во всех подробностях) в журнале регистрации данных.

8.15 Текстовый редактор

Компонент Текстовый Редактор позволяет редактировать текстовые данные различного формата (на языке Java, документы XML, файлы SNMP MIB и пр.). Текстовый редактор поддерживает различные правила подсветки синтаксиса для различных типов текстовых документов.



"Горячие" кнопки

Ниже приведен список "горячих" кнопок, поддерживаемых компонентом текстовый редактор.

Комбинация клавиш	Описание
Ctrl-A	Выбрать все
Ctrl-D	Удалить строку
Ctrl-I	Сместить строки вправо
Ctrl-J	Объединить строки
Ctrl-C	Скопировать
Ctrl-X	Вырезать
Ctrl-V	Вставить
Ctrl-Z	Отменить
Ctrl-E, Ctrl-Z	Восстановить
Ctrl-Shift-Backspace	Удалить начало строки
Ctrl-Shift-Delete	Удалить конец строки
Ctrl-Left	Перейти к предыдущему слову
Ctrl-Right	Перейти к следующему слову
Ctrl-Up	Перейти к предыдущему параграфу

Ctrl-Down	Перейти к следующему параграфу
Ctrl-Shift-Left	Расширить выделение на одно слово назад
Ctrl-Shift-Right	Расширить выделение на одно слово вперед
Shift-Left	Расширить выделение на один символ назад
Shift-Right	Расширить выделение на один символ вперед
Shift-Home	Расширить выделение до начала строки
Shift-End	Расширить выделение до конца строки
Shift-Up	Расширить выделение на одну строку назад
Shift-Down	Расширить выделение на одну строку вперед
Shift-Page Up	Расширить выделение на одну страницу назад
Shift-Page Down	Расширить выделение на одну страницу вперед
Ctrl-Shift-Up	Расширить выделение на один параграф назад
Ctrl-Shift-Down	Расширить выделение на один параграф вперед
Ctrl-Shift-Home	Расширить выделение до начала документа
Ctrl-Shift-End	Расширить выделение до конца документа
Ctrl-Backspace	Удалить символ перед курсором
Ctrl-Delete	Удалить символ после курсора
Ctrl-Home	Перейти к началу документа
Ctrl-End	Перейти к концу документа
Ctrl-[	Выбрать кодовый блок
Ctrl-]	Перейти к соотвествующей скобке
Ctrl-E, Ctrl-[	Перейти к предыдущей скобке
Ctrl-E, Ctrl-]	Перейти к следующей скобке
Ctrl-'	Прокрутить на одну строку вверх
Ctrl-/	Прокрутить на одну строку вниз
Alt-'	Прокрутить на одну страницу вверх
Alt-/	Прокрутить на одну страницу вниз
Ctrl-E, Ctrl-A	Добавить выбранный текст в буфер обмена и оставить в редакторе
Ctrl-E, Ctrl-U	Добавить выбранный текст в буфер обмена и удалить из редактора
Ctrl-E, Ctrl-J	Прокрутить до текущей строки
Ctrl-E, Ctrl-N	Поместить курсор по центру экрана

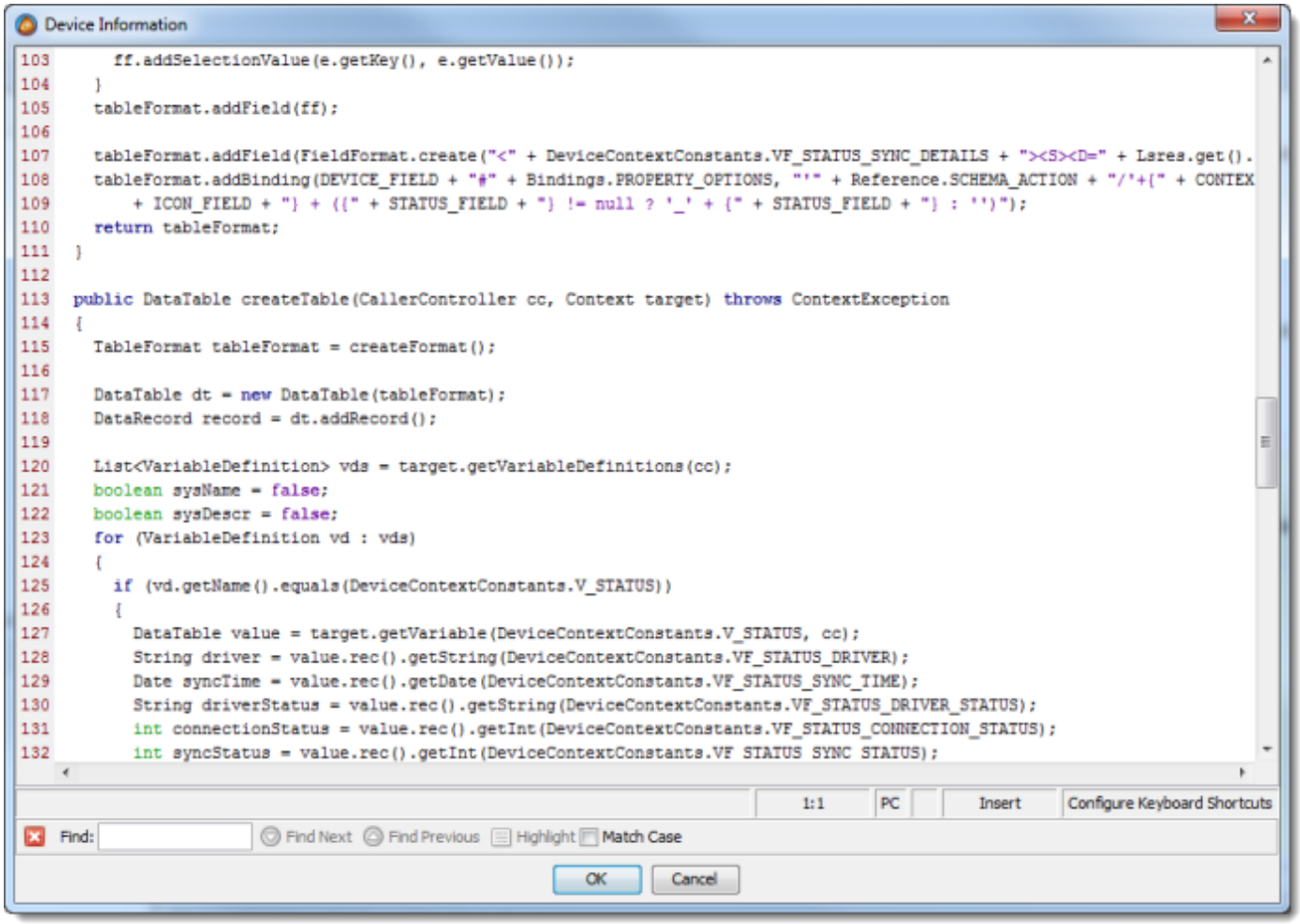
8.16 Редактор кода

Компонент **Редактор кода** позволяет редактировать код, написанный на одном из поддерживаемых языков (Java, ST 1994). Используется для создания скриптов AtomMind Server 879, скриптов виджетов 1308, программ для Управление процессами.

Редактор кода поддерживает:

- выделение синтаксиса редактируемого языка
- поиск и замену
- соответствие скобок
- сворачивание кода

- автоматический отступ
- всплывающие подсказки
- проверку синтаксиса
- автоматическое импортирование необходимых классов



Сочетания клавиш

Приведенная ниже таблица представляет комбинации клавиш по умолчанию доступные в Редакторе кода. Для их редактирования нажмите на метку **Конфигурировать комбинации клавиш** в строке состояния Редактора кода.

Имя	Комбинация клавиш	Описание
ОСНОВНОЕ РЕДАКТИРОВАНИЕ		
Стереть слева	BACK_SPACE	Удалить предыдущий символ в положении вставки.
Удалить до начала слова	Control BACK_SPACE	Удалить предыдущие символы до несловесного символа.
Удалить	DELETE	Удалить следующий символ в положении вставки.
Удалить до конца слова	Control DELETE	Удалить следующие символы до несловесного символа.
Удалить текущую строку	Control Y	Удалить текущую строку.

Вставить перенос строки	ENTER	Вставить перенос строки в положении вставки.
Разбить строку	Control ENTER	Вставить перенос строки в положении вставки и сохранить текущее положение вставки.
Начать новую строку	Shift ENTER	Начать новую строку рядом со сторокой вставки и поместить вставку в начале новой строки.
Выделение отступа	TAB	Сделать отступ на строке вставки, если нет выделения, и на всех выбранных строках, если есть выделение.
Выделение на уровень вниз	Shift TAB	Сделать отступ на нижнем уровне от вставки, если нет выделения, и на всех выбранных строках, если есть выделение.
Объединить строки	Control Shift J	Объединить следующую строку со строкой вставки, если нет выделения, и объединить все выбранные строки, если есть выделение.
Переключить Вставку/Перезапись	INSERT	Переключить статус перезаписи/вставки.
Переключить на прямоугольное выделение	Control BACK_SLASH Alt Shift INSERT	Переключить обычное выделение на прямоугольное выделение.
Переключить регистр	Control Shift U	Переключить регистр выделения.
Контекстная подсказка	Control SPACE	Вызвать контекстную подсказку.
Редактор выражений 404	Control E	Вызвать редактор выражений 404.
ИЗМЕНЕНИЕ ПОЛОЖЕНИЯ ВСТАВКИ И ВЫБОР		
Выделить все	Control A	Выделить весь текст в редакторе.
Переместить вставку на начало строки	HOME	Переместить вставку на начало текущей строки.
Переместить вставку на конец строки	END	Переместить вставку на конец текущей строки.
Выделить до начала строки	Shift HOME	Выделить с текущего положения вставки до самого начала строки.
Выделить до конца строки	Shift END	Выделить с текущего положения вставки до самого конца строки.
Переместить вставку на начало документа	Control HOME	Переместить вставку на начало редактора кода.
Переместить вставку на конец документа	Control END	Переместить вставку на конец редактора кода.
Выделить до начала документа	Control Shift HOME	Выделить с текущего положения вставки до самого начала документа.
Выделить до конца документа	Control Shift END	Выделить с текущего положения вставки до самого конца строки.

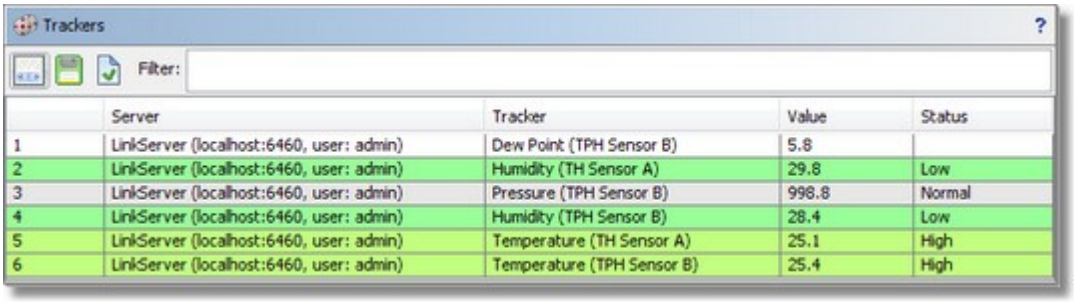
Страница вверх	PAGE_UP	Переместить вставку на страницу вверх.
Страница вниз	PAGE_DOWN	Переместить вставку на страницу вниз.
Выделить до предыдущей страницы	Shift PAGE_UP	Выделить с текущего положения вставки на одну страницу вверх.
Выделить до следующей страницы	Shift PAGE_DOWN	Выделить с текущего положения вставки на одну страницу вниз.
Переместить вставку к предыдущему символу	LEFT	Переместить вставку на один символ влево.
Переместить вставку к следующему символу	RIGHT	Переместить вставку на один символ вправо.
Выделить предыдущую вставку	Shift LEFT	Выделить предыдущий символ текущего положения вставки.
Выделить следующую вставку	Shift RIGHT	Выделить текущий символ текущего положения вставки.
Переместить вставку к предыдущему слову	Control LEFT	Переместить вставку на предыдущее слово – первый символ пробела до текущего положения вставки.
Переместить вставку к следующему слову	Control RIGHT	Переместить вставку на следующее слово – первый символ пробела до текущего положения вставки.
Выделить предыдущее слово	Control Shift LEFT	Выделить с текущего положения вставки до первого символа пробела перед ней.
Выделить следующее слово	Control Shift RIGHT	Выделить с текущего положения вставки до первого символа пробела после нее.
Переместить вставку на предыдущую строку	UP	Переместить вставку на одну строку вверх.
Переместить вставку на следующую строку	DOWN	Переместить вставку на одну строку вниз.
Выделить предыдущую строку	Shift UP	Выделить с места положения вставки на одну строку вверх.
Выделить следующую строку	Shift DOWN	Выделить с места положения вставки на одну строку вниз.
Перейти на строку	Control G	Вызвать диалог, чтобы помочь пользователю ввести индекс строки и прокрутить на нее.
Выделить слово во вставке	Control W	Выделить текущее слово.
Выделить соответствующие строки	Control B	Выделить текущий блок, который начинается и заканчивается двумя соответствующими скобками.

Дублировать выделение	Control D	Дублировать выделение. Если выделения нет, строка вставки будет продублирована.
Комментарии строки	Control SLASH	Поставить строковый комментарий для строки или нескольких строк.
Комментарии блока	Control Shift SLASH	Использовать блочный комментарий для строки или нескольких строк.
ОТМЕНИТЬ/ВЕРНУТЬ ПОСЛЕДНЕЕ ДЕЙСТВИЕ		
Отменить	Control Z	Отменить последнюю операцию редатирования.
Вернуть	Control Shift Z	Вернуть последнюю невыполненну операцию редактирования.
ОПЕРАЦИИ С БУФЕРОМ ИЗОБРАЖЕНИЯ		
Вырезать в буфера изображения	Control X Shift DELETE	Вырезать выбранный текст. Если ничего не выбрано, вырезать текущую строку.
Копировать в буфер изображения	Control C Control INSERT	Копировать выбранный текст. Если ничего не выбрано, копировать текущую строку.
Вставить из буфера изображения	Control V Shift INSERT	Вставить содержимое буфера изображения в текущее положение вставки.
Вставить буфер изображения с диалогом	Control Shift V Control Shift INSERT	Вызвать диалог, чтобы пользователь мог выбрать одно из предыдущих содержимых буфера изображения и вставить его.
НАЙТИ И ЗАМЕНИТЬ		
Найти	Control F	Вызвать диалог, чтобы пользователь мог ввести искомый текст.
Найти следующий экземпляр	F3	Найти следующий экземпляр искомого текста.
Найти предыдущий экземпляр	Shift F3	Найти предыдущий экземпляр искомого текста.
Заменить	Control R	Вызвать диалог, чтобы пользователь мог ввести текст на замену другого текста.
Быстрый поиск	Alt F3	Использовать функцию поиска для выхода на всплывающую подсказку, где пользователь может ввести текст без использования диалога.
ОПЕРАЦИИ СВЕРТКИ		
Выделить свертки	Control PERIOD	Создать свертку, которая содержит выбранный текст.
Развернуть свертки	Control EQUALS	Развенуть текущие свертки.
Свернуть свертки	Control MINUS	Свернуть текущие свертки.
Развернуть все	Control Shift EQUALS	Развернуть все свертки.

Свернуть все	Control Shift MINUS	Свернуть все свертки.

8.17 Датчики

В окне датчиков отображен список датчиков 2181. Он включает в себя датчики, полученные от всех соединений сервера 363, настроенных в AtomMind Client. Цвет датчика указывает на его состояние.



Компонент Датчики основан на Редакторе таблиц данных 382 и, таким образом, имеет аналогичные панель инструментов, контекст, контекстное меню и другие функциональные возможности.

Контекстное меню

Контекстное меню таблицы датчиков имеет два дополнительных элемента:

- **Настроить**. Приводит в исполнение действие Настроить датчик 105.
- **Удалить**. Приводит в исполнение действие Удалить датчик 106.

Операция перетаскивания мышью

Выделите любой узел Системного дерева 370 и перетащите его в таблицу Датчиков, чтобы запустить действие Создать датчик 1595. Это создаст новый датчик, который отслеживает выбранную переменную из контекста, соответствующего перемещенному узлу.

8.18 Избранное

Окно Избранное отображает список Избранного 2186. Оно сочетает избранное, полученное из всех соединений сервера 363, настроенных в AtomMind Client. Щелчок мышью по Избранному запускает соответствующее действие сервера.

Таблица включает Избранное даже с тех серверов, с которыми у AtomMind Client нет прямого соединения в настоящий момент. С момента первого подключения к новому серверу AtomMind Client получает список Избранного и хранит его с тех пор.

Компонент Избранное основан на Редакторе Таблиц Данных 382 и, таким образом, имеет аналогичные панель инструментов, контекстное меню и прочие функциональные возможности.

Контекстное меню

Контекстное меню таблицы Избранное имеет два дополнительных элемента:

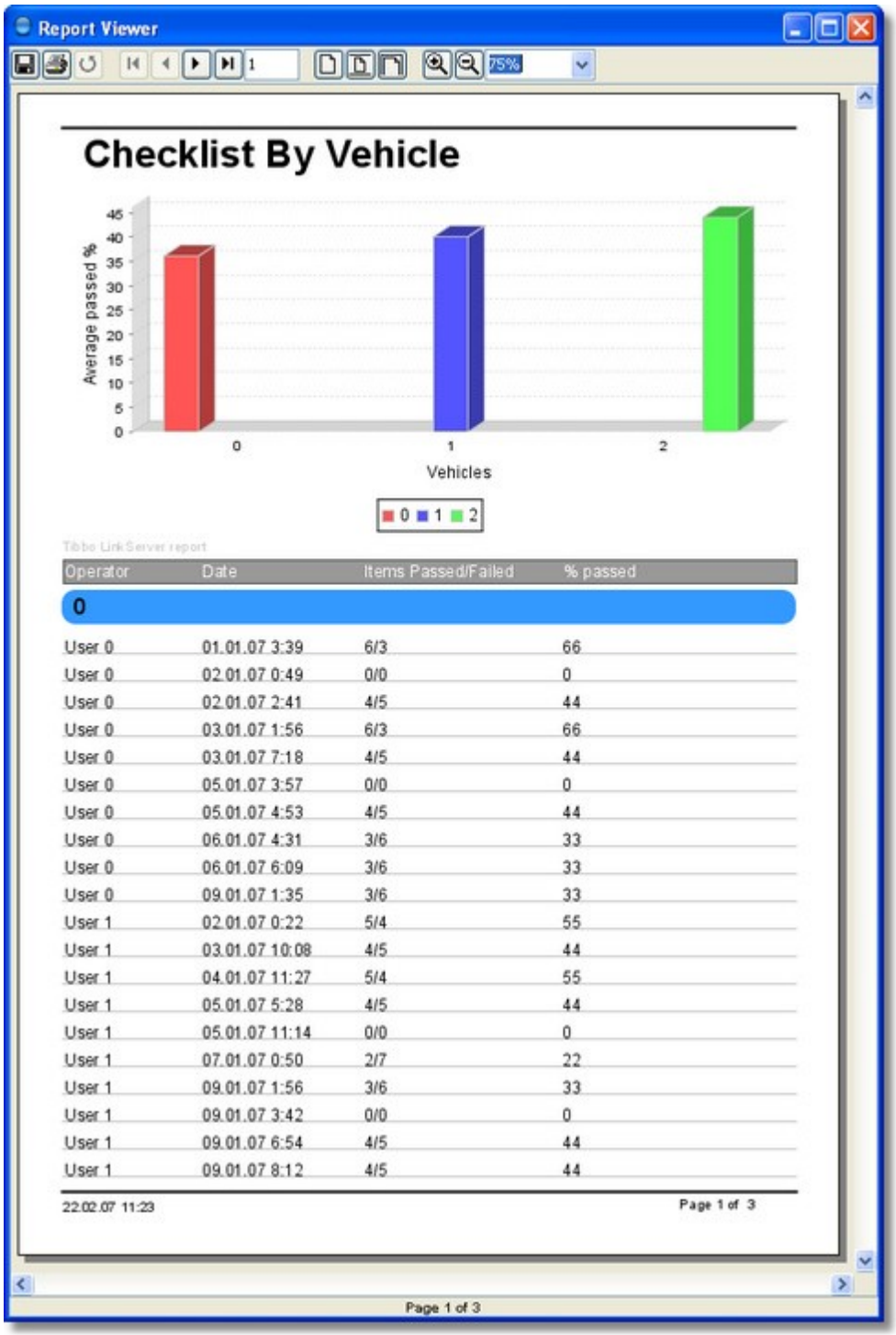
- **Настроить**. Приводит в исполнение действие Настроить избранное 105.
- **Удалить**. Приводит в исполнение действие Удалить избранное 106.

Операция перетаскивания мышью

Выделите любой узел Системного дерева 370 и перетащите его в таблицу Избранное для того, чтобы включить действие Добавить в избранное 1526. Таким образом, создается новый компонент в Избранном, который запускает выбранное действие из контекста, соответствующего перетаскиваемому узлу.

8.19 Просмотрщик отчетов

Просмотр отчетов используется для отображения, печати и сохранения отчетов 928, сгенерированных AtomMind Server. Просмотр Отчетов выглядит следующим образом:



Главное окно Просмотра Отчетов состоит из панели инструментов и самого отчета. Панель инструментов предлагает выполнить следующие операции:

Сохранить отчет. Экспортировать отчет в файл.

Напечатать отчет.

Перезагрузить отчет.

Первая страница. Перейти к первой странице.

Предыдущая страница. Перейти к предыдущей странице.

Следующая страница. Перейти к следующей странице.

Последняя страница. Перейти к следующей странице.

2 **Перейти к странице**. Перейти к определенной странице.

Нормальный размер. Вернуть стандартный масштаб просмотра.

Вместить страницу. Изменить масштаб до размера, соответствующего целой странице в окне.

Изменить ширину. Изменить масштаб ширины до размера, соответствующего ширине окна.

Увеличить масштаб. Увеличить коэффициент маштабирования.

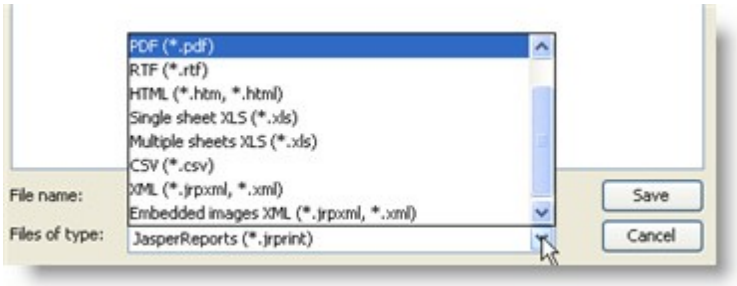
Уменьшить масштаб. Уменьшить коэффициент маштабирования.

75% **Коэффициент масштабирования**. Изменить коэффициент маштабирования.

Просмот отчетов может выводить отчеты в печать, а также экспортировать их в следующие форматы файлов:

- PDF (Adobe Acrobat)
- RTF (Rich Text Format)
- XLS (Microsoft Excel, один или несколько листов)
- HTML (язык HTML)
- CSV (Переменные, разделяемые запятой; можно импортировать в различные приложения)
- XML (Язык XML)

Чтобы экспортировать отчет, кликните мышью по кнопке **Сохранить**. Диалог Сохранения Файла выглядит следующим образом:



8.20 Редактор отчетов

Редактор отчетов - это часть AtomMind Client 359, используемая для редактирования Шаблонов отчетов 930. Он используется для изменения формы отчета, изменения его компонентов (заголовка, краткого содержания, верхнего/нижнего колонтитула страницы и деталей), а также редактирования выражений, используемых для подсчета значений данных, отображенных в отчете. Нет необходимости создавать новые шаблоны с нуля, поскольку система генерирует шаблоны по умолчанию на основе просматриваемых данных. В большинстве случаев вам просто нужно провести их тонкую настройку и сделать собственными.

Редактор отчетов работает в режиме **что видишь на экране, то и получишь,** не нуждается в программировании. Он позволяет вам:

- Изменить компоновку отчета
- Изменить разделы отчета (название, краткий обзор, заголовок/сноски страницы, заголовок/сноски столбца и другое)
- Редактировать выражения, используемые для форматирования данных (временная метка без секунд и т.д.)
- Просмотреть итоговый отчет

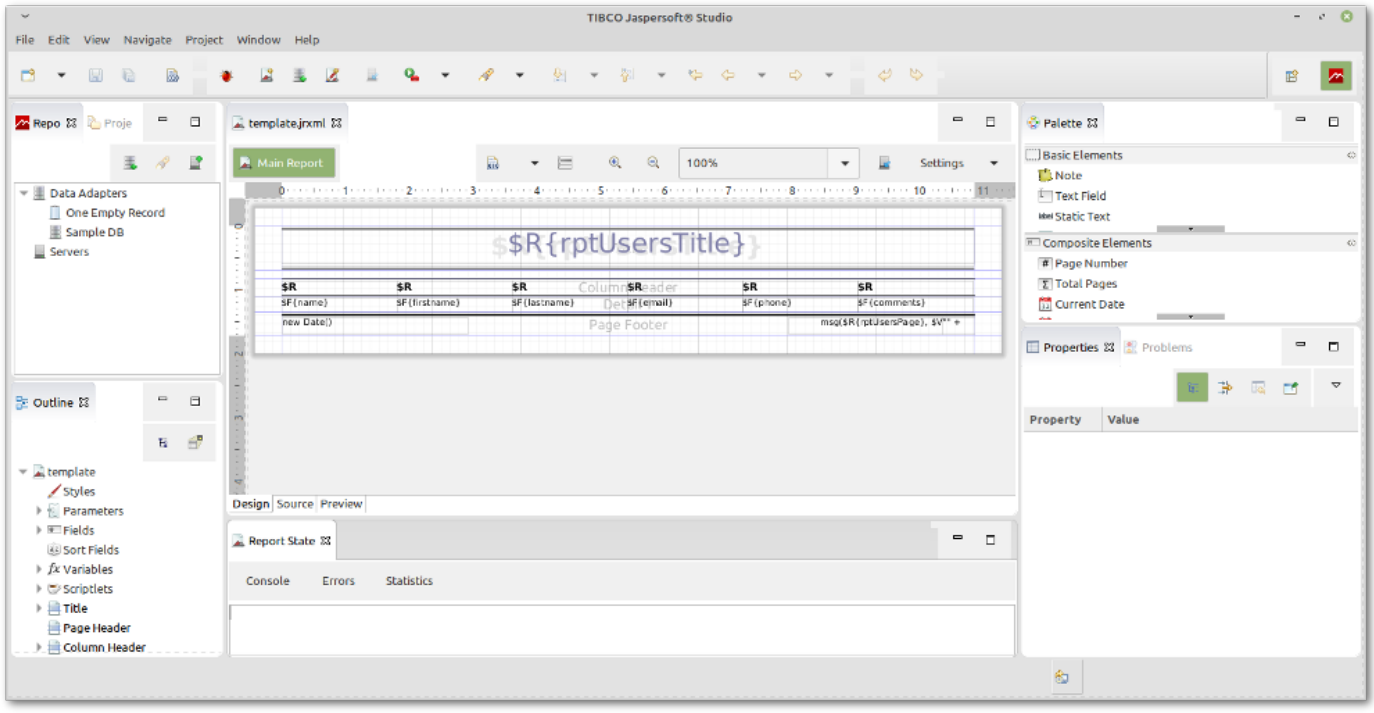
Редактор отчетов также обеспечивает:

- Набор средств для рисования прямоугольников, линий, эллипсов, текстовых полей, меток, графиков, подотчетов, штрих-кодов и других компонентов пользовательского интерфейса
- Встроенный редактор выражений с выделением синтаксиса для письменных выражений
- Поддержка Юникода и нелатинских языков (русский, китайский, японский, корейский,…)
- Просмотр структуры отчетов
- Расширенная поддержка графиков
- Операции перетаскивания
- Неограниченное количество операций отменить/вернуть действие
- Поддержка кросс-таблиц и вложенных отчетов
- Библиотека стилей

Пользовательский интерфейс редактора отчетов

Окно Редактора Отчетов состоит из нескольких основных частей:

- **Главное меню** для контроля над операциями
- **Главная панель инструментов** обеспечивает быстрый доступ к часто используемым операциям (таким как, например, просмотр отчетов).
- **Рабочее пространство**, используемое для просмотра и доступа к шаблонам отчета, просмотра/редактирования XML-источника шаблона и предпросмотра финального отчета.
- **Панель инструментов окна отчета**, позволяющая получить доступ к свойствам выбранных элементов (таких как выравнивание и шрифт элемента)
- **Панель структуры документа** содержит поля отчетов, параметры, поля (разделы) и элементы в иерархическом дереве.
- **Консоль вывода сообщений** отображает все сообщения, относящиеся к созданию отчетов, и заполняет их данными и приписывает им функциональный статус Редакторов Отчетов.
- **Панель свойств** используется для просмотра и редактирования свойств выбранного (-ых) в настоящий момент элемента (-ов).
- **Окно палитры компонентов**, позволяющая добавлять новые компоненты к отчету.
- **Панель Библиотеки Стилей** обеспечивает доступ к обычно используемым стилям элементов.



Редактирование шаблонов отчета

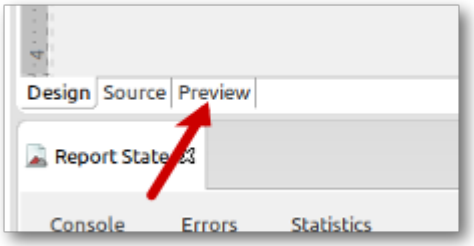
Когда запущен Редактор Отчета, он автоматически открывает шаблон отчета для редактирования. Вы можете внести любые изменения и сохранить его в любое время. Чтобы закончить редактирование, следует:

- Щелкнуть **Файл > Выход** и решите, хотите ли Вы сохранить изменения (если они были внесены) или нет:

- или щелкните мышью по иконке "Закрыть окно" в верхнем правом углу, чтобы отменить изменения и немедленно выйти из редактора отчетов.

ПРедварительный просмотр отчетов

В любое время возможно составить шаблон отчета и его просмотреть. Для предпросмотра отчета кликните на **Предпросмотр** в панели инструментов отчета:



Все отчеты по умолчанию отображаются в Просмотрщике отчетов 415. Однако в меню **Предпросмотр** можно выбрать и другое место отображения. Отчеты можно экспортировать в формат PDF, HTML или TXT.

Руководство по работе с Редактором Отчетов

AtomMind использует iReport Graphical Report Designer для редактирования шаблонов отчетов. См. документацию к iReport о том, как изменять шаблоны отчетов и использовать различные свойства подсистемы отчетов - графиков, подотчетов, выражений и скриплетов.

8.20.1 Пользовательские PDF шрифты

При использовании текстовых элементов внутри Jaspersoft Studio, пользователи могут выбрать шрифт. Хотя задача кажется простой, при использовании шрифтов возникает множество проблем. Основная проблема заключается в том, что доступные шрифты предоставляются операционной системой, и поэтому вы можете столкнуться со следующим:

- Доступный в одной ОС шрифт недоступен в другой. Тогда, в качестве альтернативы, для элемента используется шрифт по умолчанию.
- Шрифт может быть доступен в различных ОС, но немного отличаться в них.

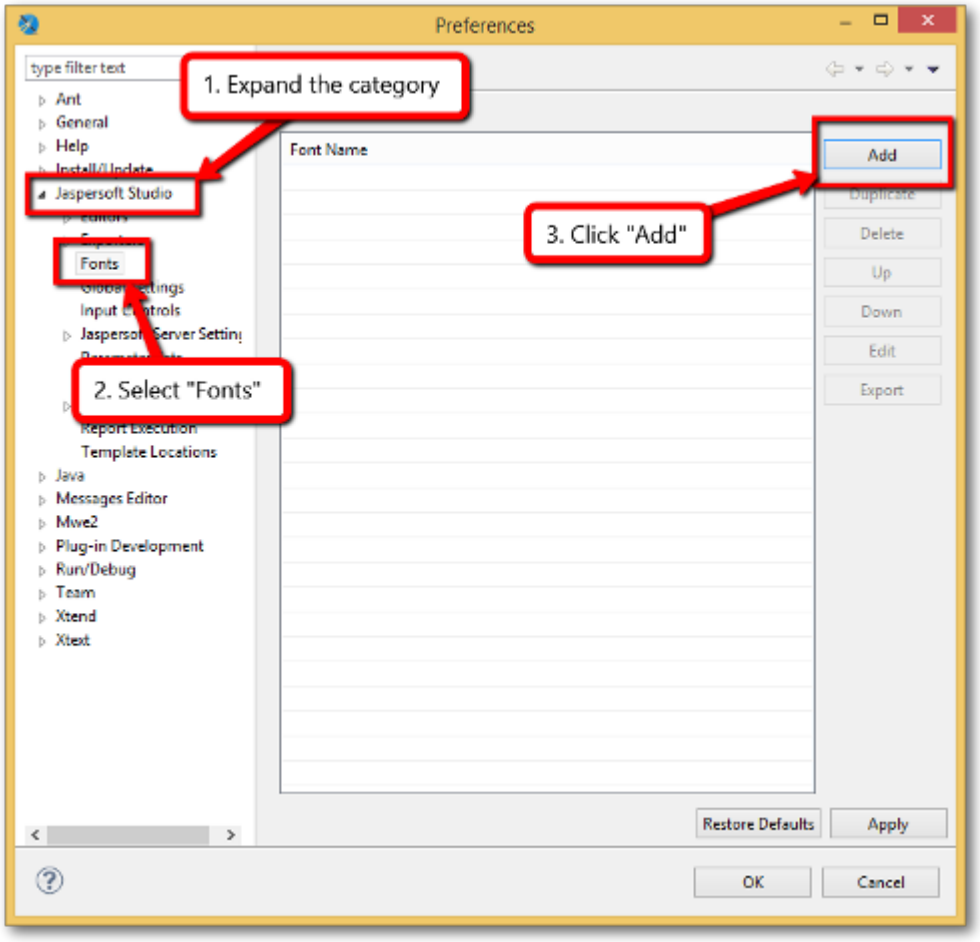
Во многих случаях это просто недопустимо. Использование шрифта, отличного от планового, может потребовать больше или меньше места для текста, а это может повлиять не только на графический вид, но и на компоновку самого отчета. Чтобы преодолеть эту трудность, можно использовать сторонние файлы со шрифтами вместо шрифтов ОС. Благодаря этому, отчет станет независимым от шрифтов ОС и в любом случае сохранит внешний вид своего расположения. У формата PDF есть свой набор шрифтов. В случае если необходимо использовать другие шрифты в документа, их нужно встроить в PDF при помощи ***Расширения Шрифтов***.

Загрузка шрифта

Во-первых, у вас должен быть шрифт, который вы хотите использовать. Jaspersoft Studio позволяет использовать широкий набор типов шрифтов, таких как TTF, SVG, WOFF и EOT. TTF - один из наиболее используемых форматов, и многие вебсайты собирают шрифты с различными лицензиями. В этом руководстве мы будем использовать шрифт Carnivalee Freakshow Font, который можно также найти в приложении в данной странице. Скачайте и сохраните TTF файл. Если вы скачаете его с вебсайта по ссылке, будет необходимо извлечь содержимое архивированного файла.

создание расширения шрифтов

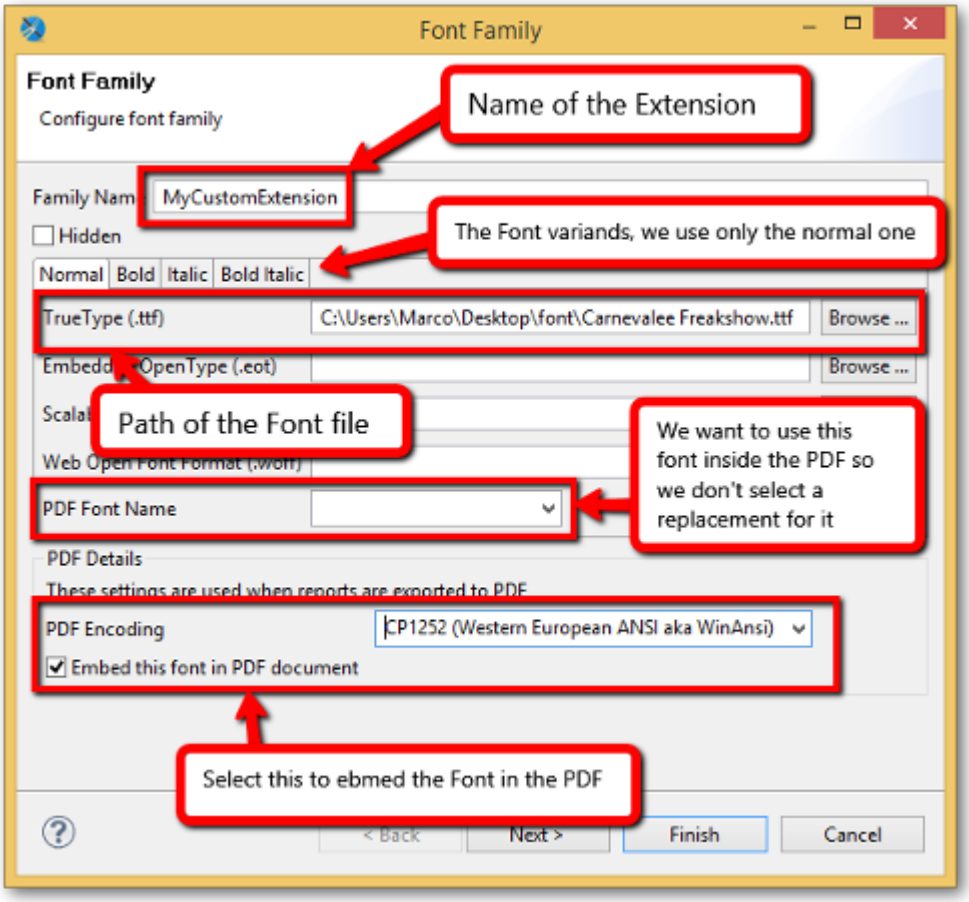
Откройте Jaspersoft Studio, перейдите к ***Window->Preferences.*** В диалоговом окне раскройте категорию ***Jaspersoft Studio*** слева, выберите ***Font*** и нажмите ***Add.***



Теперь вам необходимо задать уникальное имя для расширения шрифтов и указать путь к расширению. Так как в руководстве мы используем TTF, мы будем использовать поле True Type.

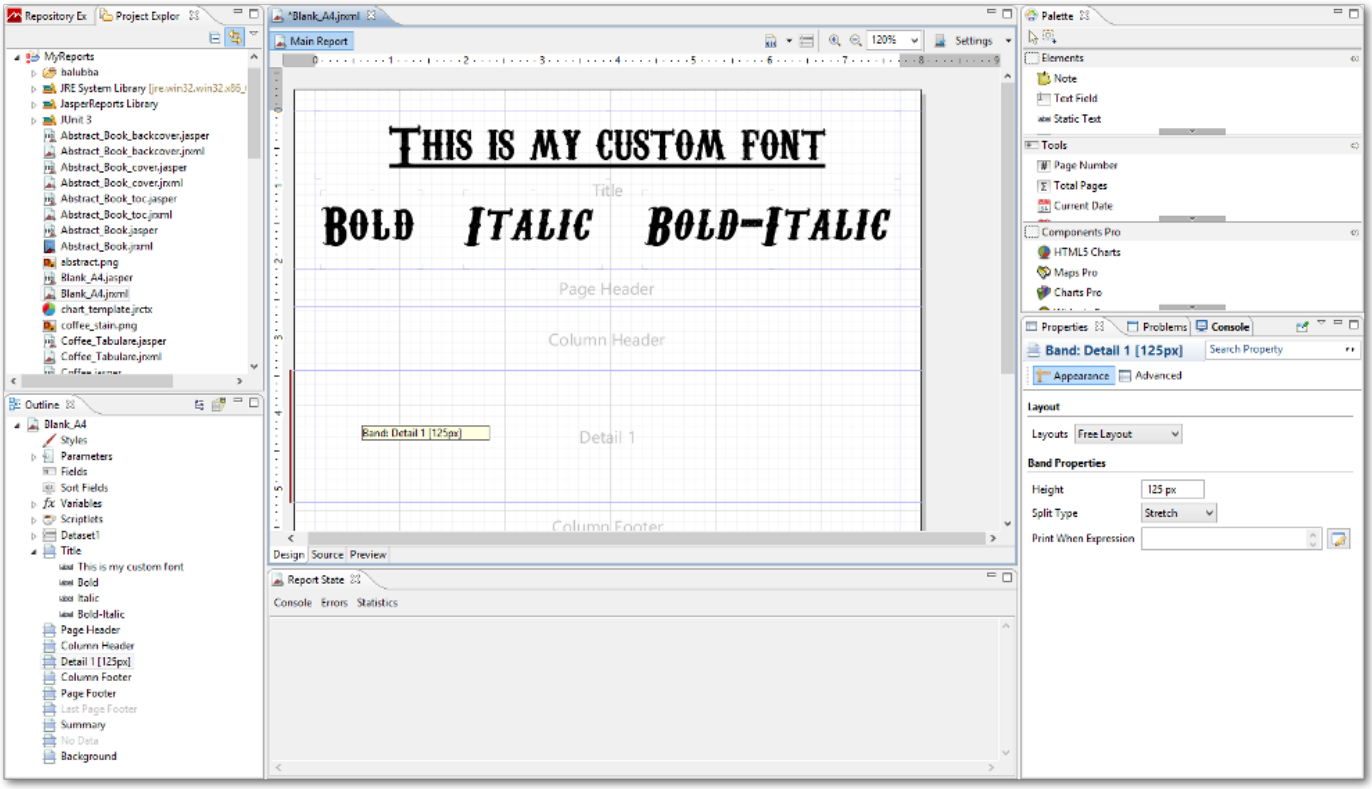
Вы можете также указать вариант шрифта для жирного, наклонного и полужирного текстовых стилей. Если вы их не укажете, вы все равно сможете использовать шрифт с одним из этих стилей. Обычный шрифт будет изменен для получения стилизованного варианта. На ваше усмотрение, вы можете задать разные шрифты для каждого из вариантов. В данном руководстве мы используем только обычный шрифт.

Вы можете также выбрать опцию замены данного шрифта при экспорте в PDF. Мы оставим это поле незаполненным, так как после экспорта в PDF мы хотим использовать тот же шрифт. По этой причине нам надо выбрать ***"Embed this font in the PDF document"*** и ***сопоставимую кодировку***, например, CP1251. Обратите внимание, что встраивание шрифта в PDF-экспорт увеличит размер получившегося PDF-файла.



Нажмите ***"Next"***. При выполнении следующих шагов вы можете предоставить дополнительную информацию другим экспортерам и ограничить использование данного шрифта только определенным количеством узлов. Нам это не надо, поэтому просто нажимаем ***"Finish",*** чтобы закрыть диалог и ***"Ok"*** для закрытия диалога предпочтений.

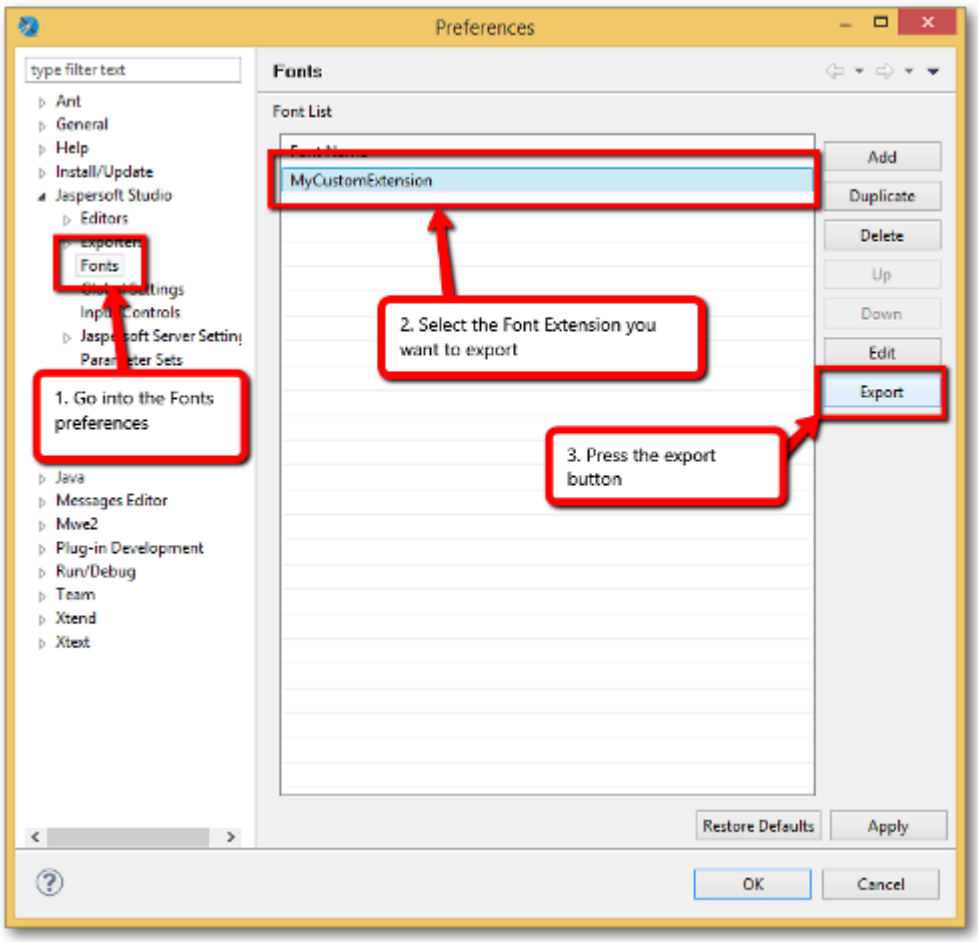
Если вы теперь откроете отчет, то среди доступных шрифтов увидите имя, совпадающее с именем вашего расширения шрифтов, ***"MyCustomExtension"*** (Если вы его не видите, попробуйте закрыть и снова открыть отчет). Выберите шрифт, чтобы применить его к элементу. Вы можете поменять стиль на жирный, курсив или полужирный, даже если вы это не указывали при создании расширения шрифтов.



Экспорт расширения шрифтов

Среди проблем, которые решает расширение шрифтов, - необходимость иметь один и тот же шрифт на каждой ОС, чтобы текст выглядел везде одинаково. Но следуя тому, что описано в руководстве, нам нужно скопировать TTF файл на каждую ОС, и на каждой системе воспроизвести расширение шрифтов с точно таким же именем. В противном случае, созданный в одной ОС отчет не найдет шрифта в другой ОС. Если бы мы использовали несколько TTF файлов для стилей текста, нам бы пришлось копировать и эти файлы и более глубоко настраивать Расширение. Это несложно, но может занять какое-то время, а повторение процедуры много раз может привести к ошибкам, например, к опечатке в имени расширения. Для упрощения задачи есть возможность экспортировать наше расширение шрифтов как JAR-файл и использовать его в любых проектах.

Сначала откройте диалог ***Preferences*** и перейдите к ***Fonts***, чтобы увидеть все ваши расширения шрифтов. Выберите ранее созданное расширение ***"MyCustomExtension"*** и нажмите ***"Export"***.



Появится диалог сохранения. Задайте имя целевого файла и сохраните его. В этом JAR-файле будет все необходимое для использования расширения на других ОС, вам лишь нужно открыть общий доступ к этому единственному файлу, чтобы перенести расширение шрифта на другую машину. В приложении к этому руководству вы можете найти файл, сгенерированный таким способом.

опубликовать шрифт на сервере

Когда у вас будет файл font.jar, перетащите его в вашу директорию ***AtomMind/lib*** и перезапустите сервер.

8.21 Ведение журнала

Аналогично *AtomMind Server, AtomMind Client* использует библиотеку журналирования **Apache Log4J** для записи вывода его внутренних событий. Он достаточно гибкий, позволяет иметь множество уровней и ресурсов для журналирования информации наряду с большим количеством направлений для записи. Запись вывода может перенаправляться на:

- Консоль
- Текстовые файлы
- Файлы XML
- Записи событий для *Windows*
- Системный журнал *UNIX*
- Базу данных
- Удаленный сетевой сервер
- Сообщения E-mail
- Службу *Java Message (JMS)*
- И многие другие.

Журналирование в AtomMind Client настраивается и управляется таким же образом, как и в AtomMind Server. Это может означать, что вы скорее всего используете настройку по умолчанию. Если вы хотите внести изменения, вам необходимо напрямую редактировать файл с настройками ведения журнала AtomMind Client, который находится в директории инсталляции и имеет имя `logging-client.xml`. Структура этого файла кратко описана в разделе файл настройки ведения журнала 167.

Дополнительная информация о журналировании содержится в следующих разделах документации:

- Настройки журналирования 166 AtomMind Server
- Файл настроек журналирования 167
- Уровни журналирования 171

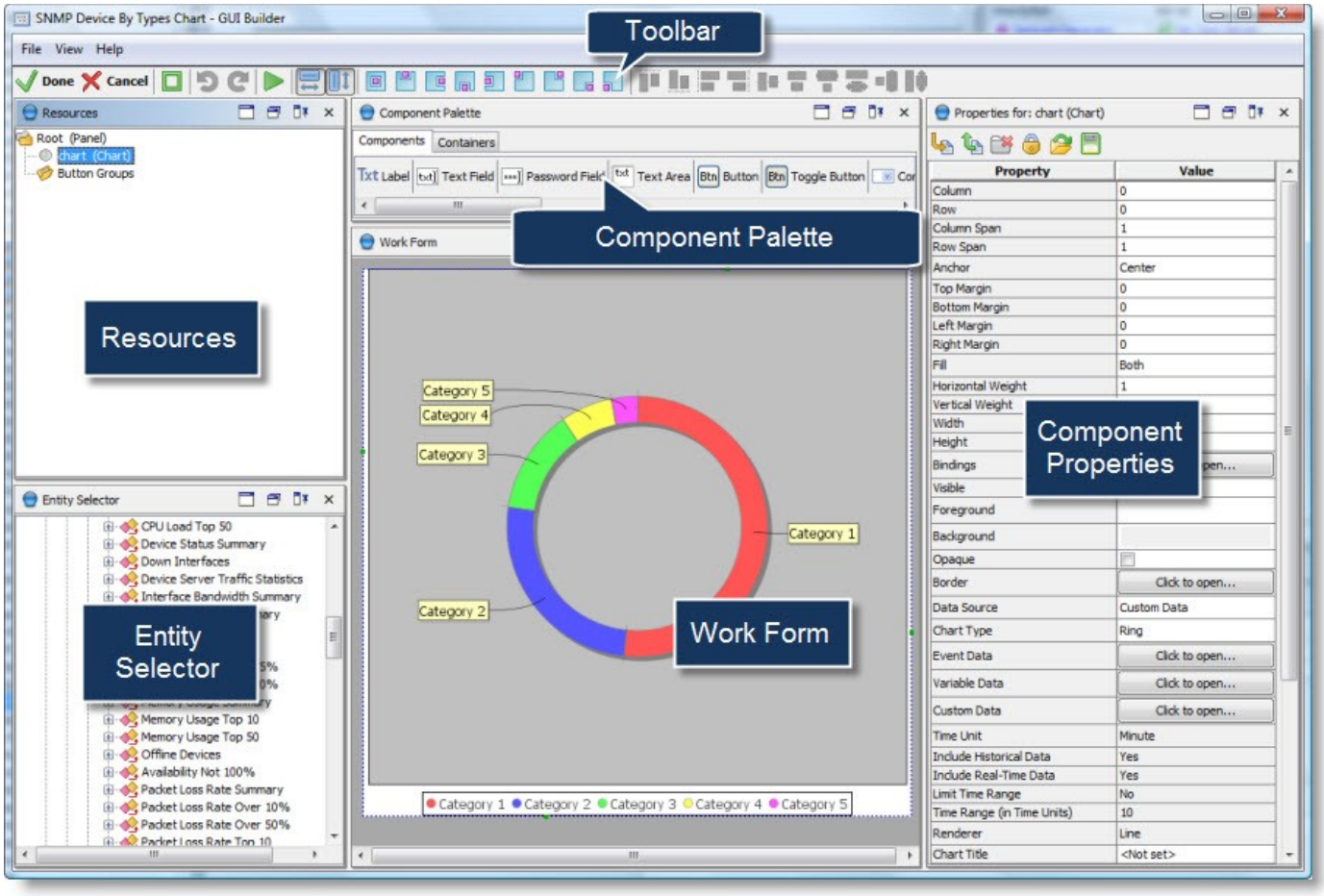
Это описание может показаться сжатым, но лишь потому, что журналирование -- это сложная тема, которая используется лишь при необходимости решить неполадки в системе. Если начать подробно описывать процедуру ведения журнала, можно написать еще одно руководство для Log4J. Мы предпочли воспользоваться сторонним инструментом журналирования ввиду его надежности, гибкости и наличию хорошей документации.

8.22 Редактор виджетов

Редактор виджетов - это часть AtomMind Client 359, которая используется для редактирования шаблонов 946 виджетов 943 или конструирования виджетов с нуля.

Главное окно

AtomMind GUI Builder всегда открывается в отдельном окне:



В главном окне располагаются несколько первичных компонентов:

- Главное Меню 424
- Панель инструментов 424
- Рабочая форма 426

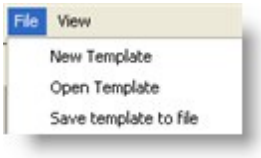
- Окно ресурсов 428
- Селектор объектов 430
- Палитра компонентов 430
- Окно свойств компонентов 431

Большинство компонентов AtomMind GUI Builder представляют собой перемещаемые панели, которые можно свернуть/развернуть, удалить, поместить в закладки, изменить их размер и пр. См. Настройка плавающих панелей 366.

В главе Редактор свойств 377 содержится информация о том, как пользоваться Редактором Свойств. Свойства компонентов перечисляются и описываются на этой странице 955.

8.22.1 Главное меню

У AtomMind GUI Builder есть два меню: Файл и Просмотр.



МЕНЮ ФАЙЛ

- **Новый шаблон**. Очищает шаблон редактируемого в настоящий момент виджета.
- **Открыть шаблон**. Загружает шаблон нового виджета из файла. Редактируемый в текущий момент шаблон будет утрачен.
- **Сохранить шаблон в файл**. Сохраняет редактируемый в настоящий момент шаблон в файл.

МЕНЮ ВИД

В этом меню можно включать и отключать элементы. Они используются для того, чтобы скрыть или отобразить разные части AtomMind GUI Builder.

- **Окно ресурсов**. Переключает Окно ресурсов 428.
- **Окно палитры компонентов**. Переключает Палитру компонентов 430.
- **Окно редактора свойств**. Переключает Окно свойств компонентов 431.
- **Окно селектора объектов**. Переключает Селектор объектов 430.

МЕНЮ СПРАВКА

- **Открыть справку по редактору виджетов**. Открывает в браузере документацию о конструкторе виджетов.
- **Открыть справку по виджетам**. Открывает в браузере документацию о виджетах.

8.22.2 Панель инструментов

Предоставляет доступ к часто используемым операциям:

	**Готово**. Завершает редактирование шаблонов виджета и сохраняет изменения в AtomMind Server.
	**Отмена**. Отменяет редактирование и отбрасывает изменения.
	**Переключить декорации**. Включает или отключает оформление ячеек разметки панели. См. оформление ячеек 427.
	**Сохранить шаблон.** Сохраняет редактируемый шаблон на сервере обновлением поля *шаблон* в свойствах виджета 947. Редактирование не прерывается.
	**Отменить**. Отменяет предыдущую операцию. Для этой функции есть горячие кнопки **Ctrl-Z**.
	**Восстановить**. Повторяет предыдущую отмененную операцию. Для этой функции есть горячие кнопки **Ctrl-Y**.

	**Запустить виджет**. Запускает редактируемый виджет в режиме тестирования. У этой функции есть горячая кнопка: **F5**.
	**Заливка по горизонтали**. Переключает горизонтальную заливку 952 выбранного компонента. Действие доступно для компонентов, расположенных в контейнере с Табличной разметкой 950.
	**Масштаб.** Увеличивает и уменьшает масштаб корневой панели. функция связана с прокруткой колеса мыши.
	**Заливка по вертикали**. Переключает вертикальную заливку 952 используемого компонента. Действие доступно для компонентов, расположенных в контейнере с Табличной разметкой 950.
	**Поместить привязку 951 выбранного компонента в центр**. Действие доступно для компонентов, расположенных в контейнере с Табличной разметкой 950.
	**Установить точку привязки 951: Север**. Действие доступно для компонентов, расположенных в контейнере с Табличной разметкой 950.
	**Установить точку привязки 951: Восток**. Действие доступно для компонентов, расположенных в контейнере с Табличной разметкой 950.
	**Установить точку привязки 951: Юг**. Действие доступно для компонентов, расположенных в контейнере с Табличной разметкой 950.
	**Установить точку привязки 951: Запад**. Действие доступно для компонентов, расположенных в контейнере с Табличной разметкой 950.
	**Установить точку привязки 951: Северо-Запад**. Действие доступно для компонентов, расположенных в контейнере с Табличной разметкой 950.
	**Установить точку привязки 951: Северо-Восток**. Действие доступно для компонентов, расположенных в контейнере с Табличной разметкой 950.
	**Установить точку привязки 951: Юго-Восток**. Действие доступно для компонентов, расположенных в контейнере с Табличной разметкой 950.
	**Установить точку привязки 951: Юго-Запад**. Действие доступно для компонентов, расположенных в контейнере с Табличной разметкой 950.
	**Выровнять выбранные компоненты по верхнему краю**. Действие доступно для компонентов, расположенных в контейнере с Абсолютной разметкой 954.
	**Выровнять выбранные компоненты по нижнему краю**. Действие доступно для компонентов, расположенных в контейнере с Абсолютной разметкой 954.
	**Выровнять выбранные компоненты по левому краю**. Действие доступно для компонентов, расположенных в контейнере с Абсолютной разметкой 954.
	**Выровнять выбранные компоненты по правому краю**. Действие доступно для компонентов, расположенных в контейнере с Абсолютной разметкой 954.
	**Выровнять выбранные компоненты по горизонтальной оси относительно центра**. Действие доступно для компонентов, расположенных в контейнере с Абсолютной разметкой 954 .
	**Выровнять выбранные компоненты по вертикальной оси относительно центра**. Действие доступно для компонентов, расположенных в контейнере с Абсолютной разметкой 954 .
	**Установить ширину выбранных компонентов, как равную их минимальной ширине.**
	**Установить ширину выбранных компонентов, как равную их максимальной ширине.**
	**Установить высоту выбранных компонентов, как равную их минимальной высоте.**
	**Установить высоту выбранных компонентов, ка равную их максимальной высоте.**
	**Распределить по центрам горизонтальной оси**. Выровнять компоненты таким образом, что расстояния между их центрами вдоль горизонтальной оси становятся равны. Левые и правые компоненты не перемещаются. Эта кнопка доступна при выборе трех или более компонентов.
	**Распределить по краям горизонтальной оси**. Выровнять компоненты таким образом, что расстояния между краями соседних компонентов вдоль горизонтальной оси становятся равны. Левые и правые компоненты не перемещаются.

	Эта кнопка доступна при выборе три или более компонентов.
	**Распределить по центрам вертикальной оси**. Выровнять компоненты таким образом, что расстояния между их центрами вдоль вертикальной оси становятся равны. Самые верхние и нижний компоненты не перемещаются. Эта кнопка доступна при выборе трех или более компонентов.
	**Распределить по краям вертикальной оси**. Выровнять компоненты таким образом, что расстояния между краями соседних компонентов вдоль вертикальной оси становятся равны. Левые и правые компоненты не перемещаются. Эта кнопка доступна при выборе три или более компонентов.

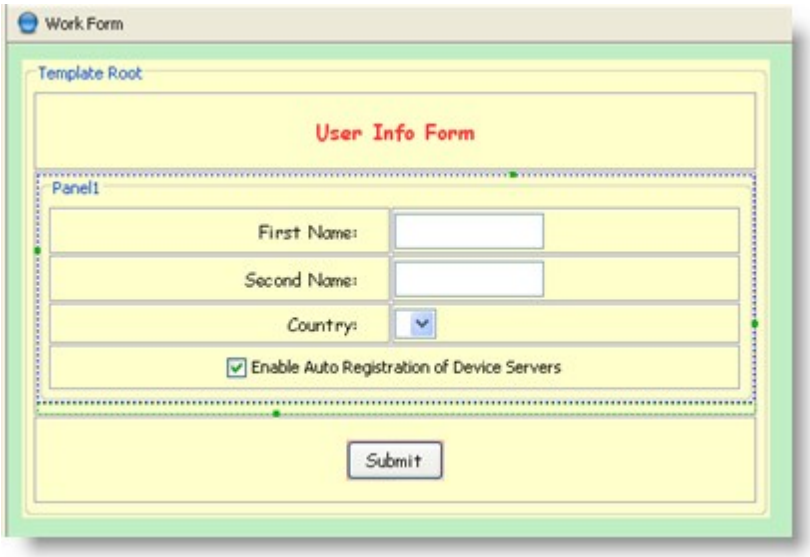
8.22.3 Рабочая форма

Рабочая форма используется для предварительного просмотра разметки 950 виджета во время его составления и редактирования. Это основная часть окна AtomMind GUI Builder

Рабочая форма всегда отображает главный контейнер виджета - корневую панель. Если он содержит дочерние компоненты или контейнеры, вся иерархия отображается в рабочей форме. Другими словами, когда оформление ячеек 427 отключено, виджет в рабочей форме напоминает то, как он будет выглядеть при запущенном AtomMind Client 359.

Вы можете поместить новые компоненты в рабочую форму, перемещать компоненты в текущем контейнере или в другие контейнеры, изменять размер компонентов, а также изменять их поля 952, создавать новые привязки 439 и пр. См. редактирование разметки виджета 432.

Вот, как выглядит форма во время редактирования шаблона виджета 946:

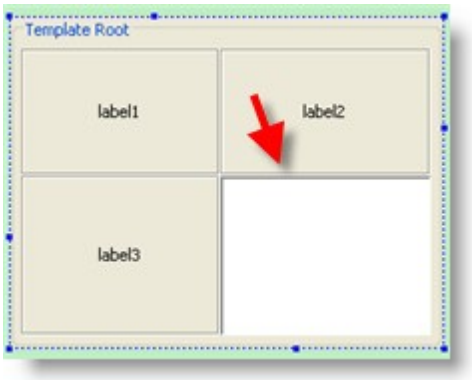


Компоненты в AtomMind GUI Builder выглядят аналогично тому, как они будут выглядеть, когда виджет запущен в AtomMind Client, но с некоторыми отличиями:

1. Игнорируется большая часть попыток ввода пользователя (например, кнопки визуально "не нажимаются", когда пользователь кликает по ним мышью)

2. Если кнопка **переключения оформления** (см. ниже) нажата, у всех контейнеров видно обрамление, отображающее их заголовки:

3. Пустые ячейки в табличной разметке 950 представлены в виде белых прямоугольников с обрамлением.



Обрамление этих прямоугольников используется для изменения размера GUI-компонентов и определения их границ. См. редактирование разметки виджета 432.

Оформление ячеек

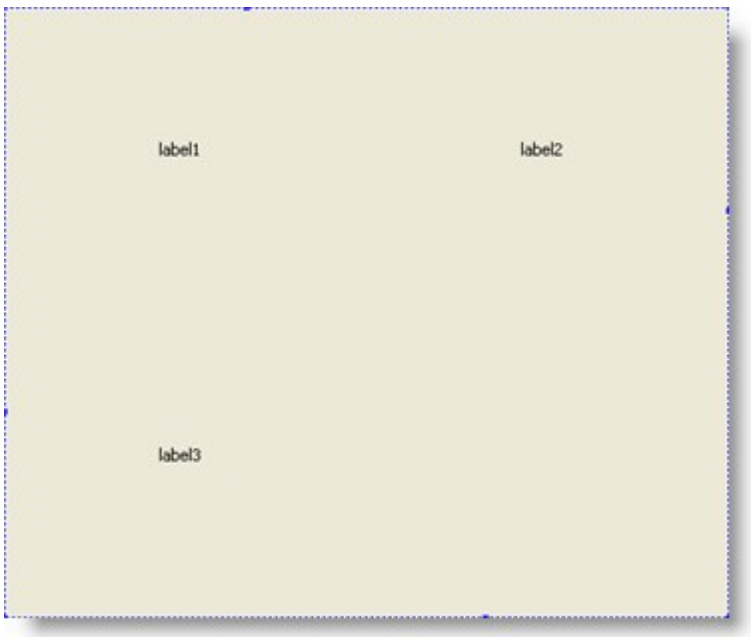
Когда ячейка табличной разметки 950 отображена в рабочей форме, у нее могут быть дополнительные *оформления*. Эти оформления облегчают редактирование разметки, но виджет при этом выглядит не так, как при обычном режиме выполнения 949. Оформление переключает кнопка, расположенная в панели инструментов 424 AtomMind GUI Builder, "**Переключить Оформление".**

Дополнительные визуальные элементы отображаются, когда включены оформления:

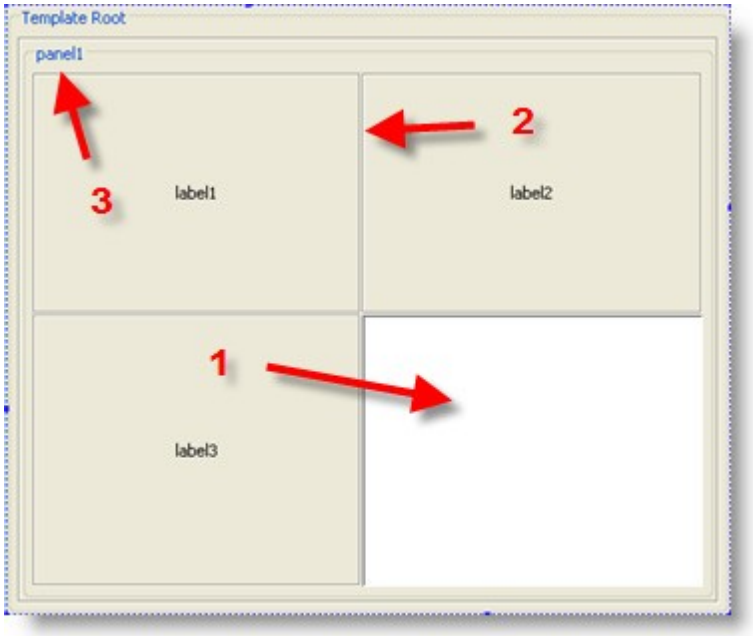
1. Пустые ячейки Табличной Разметки отображены как белые прямоугольники.
2. У каждой ячейки Табличной Разметки есть обрамление, которое помогает находить разрозненные и соединенные ячейки.
3. У контейнеров 1041 обрамление отображает их названия. Если у контейнера есть множество панелей, каждая панель также содержит обрамление заголовка с названием.

Панель с табличной разметкой (с/без оформлений) выглядит следующим образом:

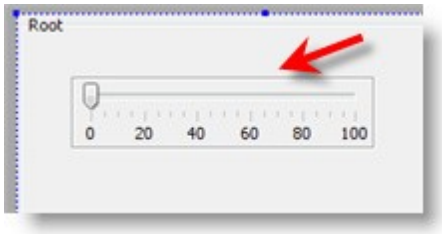
Без оформлений:



С оформлениями:



В абсолютной разметке 954 каждый компонент оформлен рамкой, которая помогает просматривать ее поля:



Контекстное меню

Щелчок правой кнопкой мыши по компоненту рабочей формы вызывает то же самое контекстное меню, которое появляется внутри Окна Ресурсов. Описание элементов меню можно найти здесь 429.

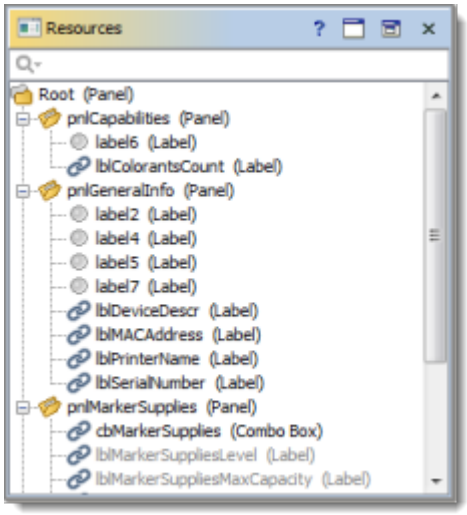
Панель графических примитивов

С правой стороны рабочей формы есть специальная панель инструментов, позволяющая рисовать графические примитивы 1269. Эта панель инструментов становится активной, как только выбирается контейнер 1041 с абсолютной разметкой 954.

См. Рисование графических примитивов 439 для получения информации о процессе создания графических примитивов.

8.22.4 Окно ресурсов

Окно ресурсов содержит дерево компонентов виджета и другие объекты (т.е. Группы кнопок 968). Корневая панель 1042 формирует корневой узел дерева. Обычные компоненты и контейнеры отображаются как узлы с разными иконками. Компоненты, которые содержит контейнер или панель (т.е. в случае Многоуровневой панели 1049) отображаются как их дочерние узлы.



Дерево ресурсов можно использовать для выбора компонентов и панелей в мультипанельных контейнерах. Когда выбран компонент или панель в окне ресурсов, он также выбирается и в рабочей форме. Его свойства отображаются в Окне свойств компонента 431.

Невидимые компоненты (у которых свойство **Видимый** установлено в FALSE) отмечены серым цветом.

Иконки узлов

	Корневая панель 1042 виджета
	Компонент Контейнер 1041.
	Стандартный 955 компонент.
	Стандартный компонент, у которого есть хотя бы одна привязка 1295 (входящая или исходящая).

Контекстное меню узла дерева ресурсов

У контекстного меню для ресурса имеются следующие операции:

- **Редактировать привязки**. Открывает привязки 1295, относящиеся к редактированию данного компонента.
- **Переименовать**. Это способ изменить имя компонента, контейнера или панели.
- **Удалить**. Навсегда удаляет выбранный узел (контейнер, компонент или панель). Контекстное меню для узла может также содержать операции, предназначенные именно для данного компонента. Например, контекстное меню для Панели с вкладками содержит элемент **Добавить Панель**. У узла группы кнопок 1040 тоже есть операция **добавить Группу Кнопок**.
- **Создать копию**. Делает копию компонента.

Если выбранный узел -- контейнер 1041, в его контекстном меню также есть подменю **Создать**, которое используется для создания нового компонента-виджета в в этом контейнере.

Аналогичное меню появляется, если щелкнуть правой кнопкой мыши по компоненту в Рабочей форме 426.

Переименование компонентов

Имя компонента используется для обращения к нему через привязки 1295. Компонент можно переименовать через элемент меню **Переименовать**. Переименование выглядит следующим образом:

Если у переименованного компонента есть привязки, появится всплывающее окно, в котором можно будет их отладить согласно измененному имени компонента.

Компоненты поиска/фильтра

Используйте поле фильтра в верхнем окне ресурсов для отображения только тех компонентов, чье имя/тип содержат указанную строку.

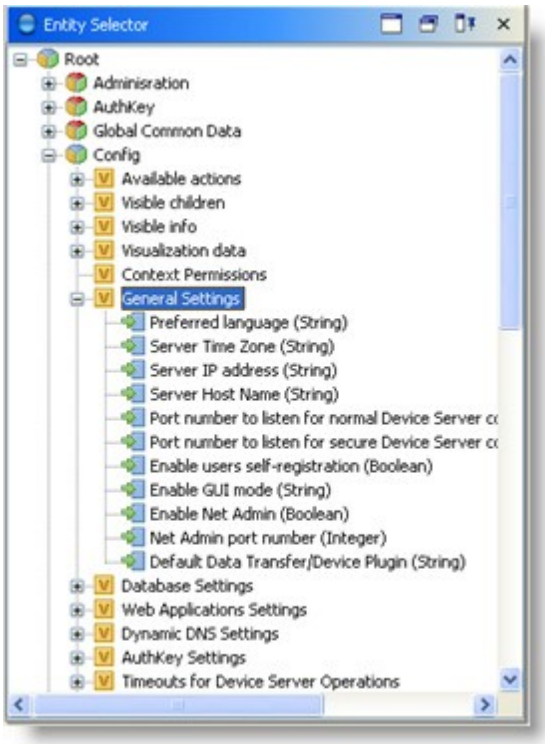
Операции по перетаскиванию мышью

Дерево ресурсов поддерживает следующие операции по перетаскиванию:

- Если компонент перетаскивается из палитры компонентов 430 в узел контейнера с табличной разметкой 950, новый компонент помещается в первую свободную ячейку таблицы компонента. Если в контейнере нет свободных ячеек, в этом случае создается новая ячейка.
- Если компонент перетаскивается из палитры компонентов 430 в узел контейнера с абсолютной разметкой 954, новый компонент помещается в положение X=0, Y=0.
- Если компонент перетаскивается из рабочей формы 426 в другой компонент рабочей формы или в дерево ресурсов, они меняются местами.

8.22.5 Селектор объектов

Окно селектора объектов 402 позволяет осуществлять поиск по контекстному 41 дереву AtomMind Server и видеть все доступные переменные 61 и функции 70 с их полями. Контекст по умолчанию 946, заданные во время редактирования виджета, а также все его дочерние узлы отмечены **жирным** шрифтом.



Селектор объектов используется в большей степени для создания новых привязок 1295. См. создание привязок 439

8.22.6 Палитра компонентов

Палитра компонентов используется для добавления новых компонентов в разметку виджета путем перетаскивания их мышью в рабочую форму 426 или в контейнер в окне ресурсов 428.

У палитры несколько вкладок:

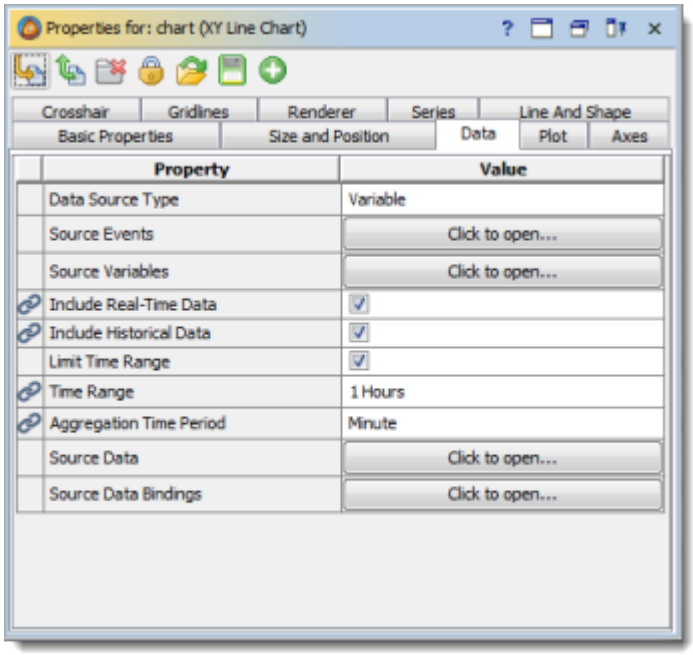
- **Компоненты**. Содержит все обычные компоненты 955.
- **Контейнеры**. Содержит все контейнеры 1041.
- **Графики категорий.** Содержит все графики категорий 1067.

- **XY Графики**. Содержит все графики XY 1099.
- **Прочие графики**. Содержит все другие графики 1149.

В разделе редактирование разметки виджета 432 содержится дополнительная информация о том, как использовать палитру компонентов, когда необходимо добавить новые компоненты в разметку виджета.

8.22.7 Свойства компонентов

Это окно содержит редактор свойств 377, который используется для изменения свойств выбранного компонента.



Обратите внимание, что свойства, отмеченные иконкой 🔗, имеют соответствующие привязки 943. Доступ к этим привязкам осуществляется через контекстное меню свойств (см ниже)

При изменении свойств, их сохранение выполняется немедленно. Как правило, нет необходимости кликать по кнопке **Сохранить** в панели инструментов Редактора Свойств.

Если новое значение свойства не может сохраниться автоматически (из-за того, что было введено неправильное значение, например, отрицательная ширина), статус этого значения будет отмечен в Редакторе Свойств как "Изменен". Измененные свойства выделяются темно-серым цветом, и Вы должны исправить их, а затем кликнуть по кнопке **Сохранить**, расположенной в панели инструментов. Если сохранение, выполненное таким очевидным образом, не выполнено, появляется Диалоговое окно сообщения об ошибке 407, в котором описана проблема.

Контекстное меню

Контекстное меню редактора свойств компонента позволяет управлять привязками 1295 и специальными свойствами 1277 компонента. У контекстного меню есть следующие элементы:

- **Привязать свойство**. Создает новую привязку, которая изменит текущее свойство, т.е. указатель 1295 на данное свойство. Система подсказывает пользователю выбрать сначала другие свойства привязки 1295.
- **Редактировать привязки**. Открывает диалоговое окно, позволяющее настраивать все привязки, относящиеся к данному свойству, т.е. имеет это свойство как цель 1295, активатор 1296, или появляется внутри выражения привязки 1296.

- **Просмотреть информацию о свойстве**. Показывает подробную информацию 61 о свойстве, включая его формат 50.
- **Добавить специальной свойство**. Позволяет создавать новые специальные свойства для текущего компонента.
- **Редактировать специальное свойство**. Открывает свойство. Открывает параметры выбранного специального свойства для редактирования.
- **Удалить специальное свойство**. Удаляет выбранное специальное свойство.

8.22.8 Редактирование разметки виджета

Виджет может содержать множество вложенных контейнеров, у каждого из которых своя компоновка 950. Типы компоновки могут быть совмещены, т.е. корневая панель с абсолютной компоновкой может иметь дочернюю вложеную панель 1045, чьи вкладки используют сетчатую компоновку, а одна из ячеек вкладки может иметь вложенную панель 1045 с абсолютной компоновкой, и т.д.

Компоновка каждого контейнера редактируется отдельно. Однако возможно перемещать компоненты между контейнерами.

Редактирование компоновки включает в себя:

- Добавление к виджету новых компонентов путем перетаскивания их из палитры 430 и размещения на рабочей форме 426
- Перемещение компонентов из одного контейнера в другой
- Перемещение компонентов между двумя местоположениями в одном контейнере
- Изменение размеров компонентов
- Копирование компонентов
- Редактирование размера и положения 950 компонента (поля, точки привязки и т.д.)
- Рисование графических примитивов 1268

Прежде чем ознакомиться с данным разделом, Вы должны прочитать и усвоить информацию о разметке виджета 950.

Результата от операций, описанных в этом разделе, можно добиться редактированием вручную визуальных ограничений 950 компонентов виджета.

Копию компонента можно сделать путем перемещения его мышкой в другое место, при этом должна быть нажата кнопка Alt.

Увеличить или уменьшить размер всего виджета можно, нажав **Ctrl и одновремнно вращая колесико мыши**.

8.22.8.1 Редактирование сетки

Чтобы добавить новые компоненты из палитры компонентов 430 или перемесить и изменить компоненты, которые уже расположены в рабочей форме 426, можно их перетащить мышью.

Компонент можно перетащить из...

1. палитры компонентов 430 (чтобы создать новый компонент выбранного типа)
2. рабочей формы 426,
3. окна ресурсов 428.

Компонент можно перетащить в...

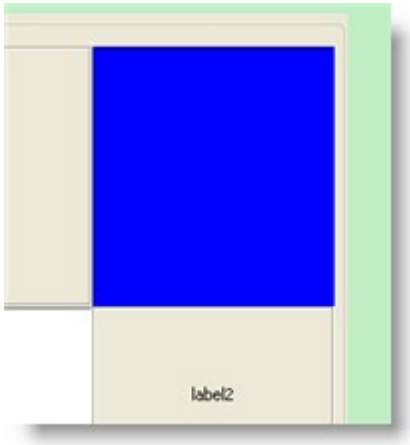
1. В пустую ячейку в табличной разметке,
2. На существующий компонент,

3. На зону сброса, что приводит к автоматическому созданию новой строки и/или колонки в табличной разметке панели или
4. На зону сброса, приводящую к скрытому созданию новой Панели 1045.

Отдельные компоненты и поддиректории компонентов событий можно перетаскивать мышью между разными виджетами, при этом каждый из них редактируется в отдельном экземпляре AtomMind GUI Builder.

1. ПЕРЕМЕЩЕНИЕ В ПУСТУЮ ЯЧЕЙКУ

Когда компонент перемещается в рабочую форму, а мышь зависает над пустой ячейкой (у такой ячейки белый фон, если включено оформление 427), ячейка выделяется голубым цветом:



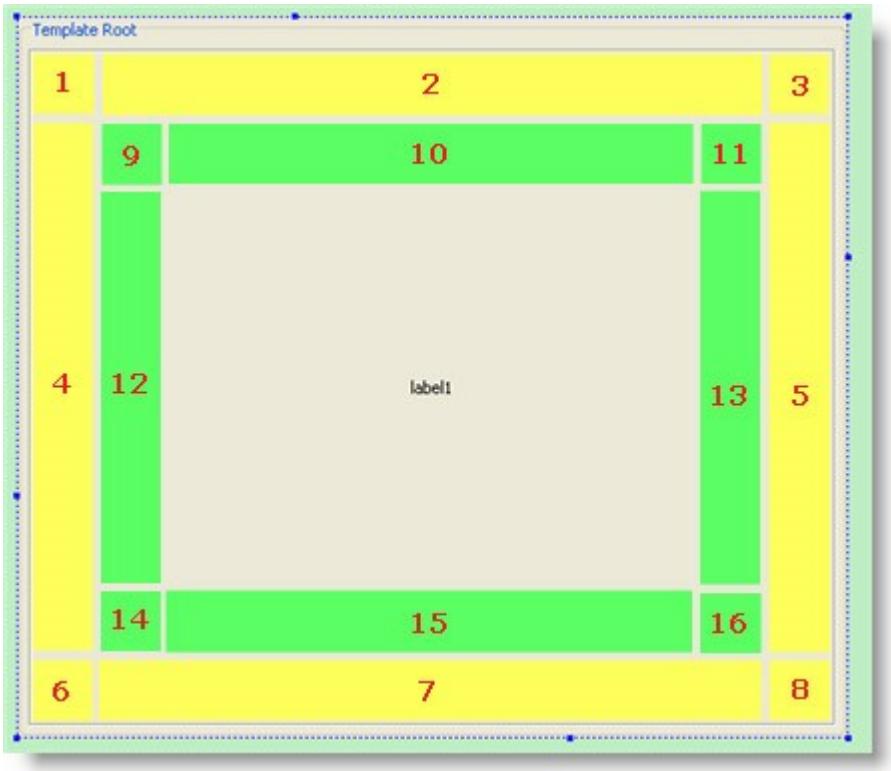
Компонент, перемещенный сюда, займет пустую ячейку

2. ПЕРЕТАСКИВАНИЕ НА СУЩЕСТВУЮЩИЙ КОМПОНЕНТ

Можно перетащить существующий компонент из его местоположения в форме на другой компонент. Два компонента обменяются местами, даже если они располагались в разных контейнерах. Однако это правило не действительно для новых компонентов, перетаскиваемых из палитры.

2 И 3. ПЕРЕТАСКИВАНИЕ НА СПЕЦИАЛЬНОЕ МЕСТОПОЛОЖЕНИЕ

Упомянутые ранее третий и четвертый методы работают одинаково. Перетаскиваемый компонент помещается в ячейку, которую уже занимает другой компонент, но не поверх другого компонента. У каждой ячейки, занятой компонентом (**label1** в приведенной ниже таблице), есть 16 возможных мест для размещения.

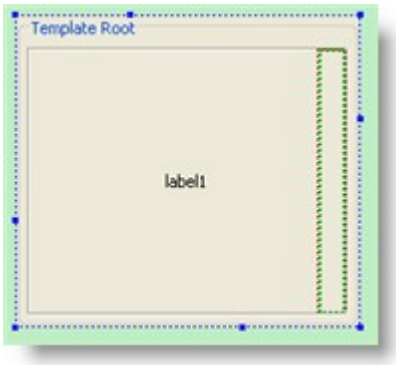


Если компонент перемещается поверх одного из первых восьми мест (выделено на картинке желтым), он помещается в новый ряд и/или колонку табличной разметки. Подробно это описывается для метода №3, указанного ниже.

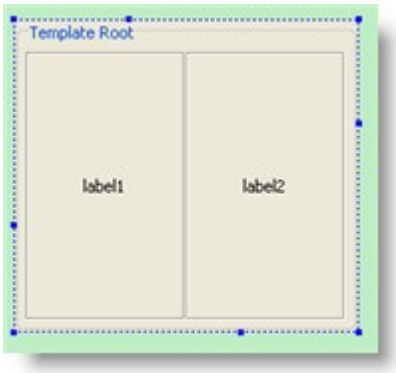
Если компонент перемещается поверх месторасположений 9-16 (выделено зеленым, описано выше), создается новый контейнер панели[1045] в занятой ячейке **label1**. У этой панели теперь будет два компонента: **label1** и перемещенный компонент. Более подробное описание приводится ниже, в описании для метода №4.

3. ПЕРЕТАСКИВАНИЕ НА ЗОНУ 1-8

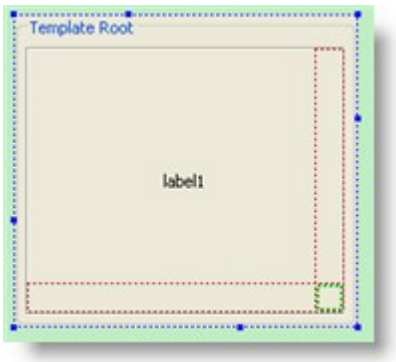
Если компонент перетаскивается на зону 1-8 (изображено на рисунке выше), он переместится в табличную разметку контейнера **label1** (в нашем случае это корневая панель). Новый ряд и/или колонка будут созданы в табличной разметке для размещения перемещенного компонента. Когда мышь зависает над месторасположениями 1-8, отображается "фантомные" рамки, что позволяет увидеть, где будет располагаться новый ряд и/или колонка. Если мышь зависает над зоной 5, это будет выглядеть следующим образом:



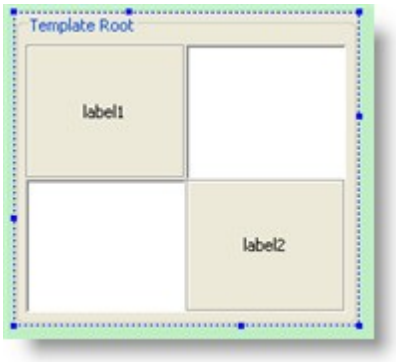
Если теперь отпустить новый компонент, справа от **label1** будет создана новая колонка:



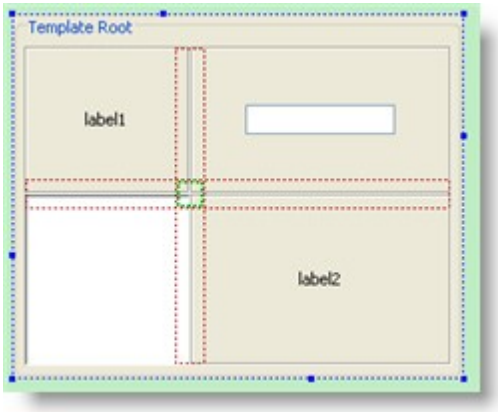
Если мышь остановится над зоной номер 8 , там появятся "фантомные" рамки:



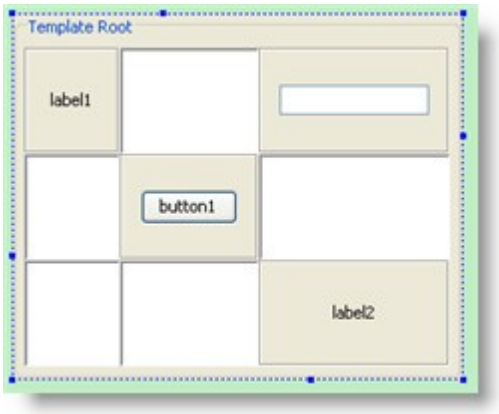
В случае, описанном выше, новый компонент будет размещен на новый ряд или колонку, созданные внизу и справа от **label1**. Однако при этом появятся две дополнительные пустые ячейки:



В усложненной разметке "фантомные" рамки выглядят следующим образом:

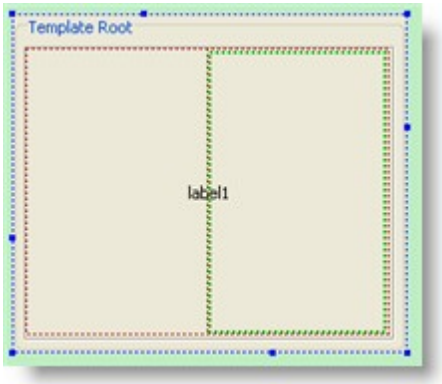


Так выглядит разметка, когда компонент перемещен:

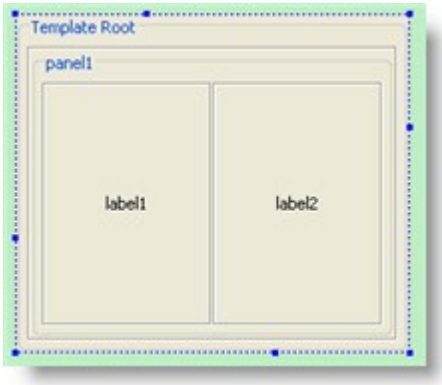


4. ПЕРЕМЕЩЕНИЕ В ЗОНУ 9-16

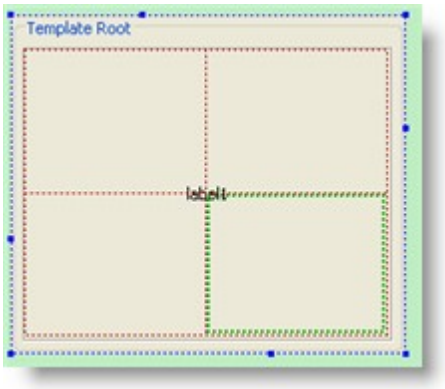
Перемещение в эту зону приводит к созданию в ячейке новой панели 1045. Эта панель будет содержать два компонента: **label1** и перемещенный компонент. Их взаимное положение в новой панели будет зависеть от того, куда был перемещен новый компонент. "Фантомные" рамки также появляются, когда мышь задерживается над этими зонами во время операции перетаскивания. "Фантомная" рамка для зоны 13 выглядят следующим образом:



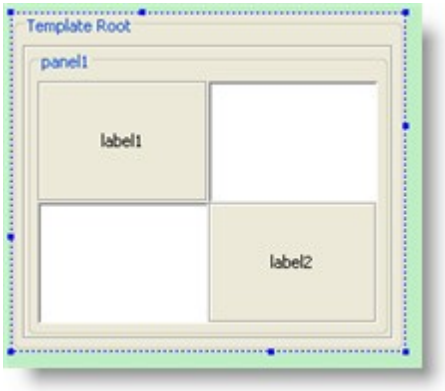
Если компонент перемещается в эту зону, будет создана новая панель, содержащая две ячейки в одном ряду:



В зоне 16 "фантомные" рамки показывают, что новая панель будет содержать четыре ячейки. **label1** будет размещен в ряду 1, колонке 1, в то время как перемещаемый компонент окажется в ряду 2 колонке 2:

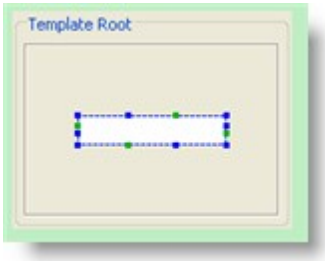


Новая панель выглядит следующим образом:

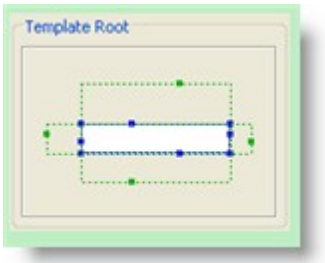


МАСШТАБИРОВАНИЕ КОМПОНЕНТОВ И РЕДАКТИРОВАНИЕ ИХ ПОЛЕЙ

Когда компонент выбран в рабочей форме, у него появляется голубая окантовка с синими и зелеными точками, которая позволяет изменять размер полей Табличной Разметки:



Чтобы изменить размер компонента, следует перетащить мышью синие точки. Чтобы изменить поля, нужно тащить зеленые точки. Если значение полей равно нулю, они изображены вокруг компонента, как прорисованные зеленым пунктиром четырехугольники:



Если удерживать кнопку **Shift** во время масштабирования компонента, это приведет к тому, что его пропорции сохранятся.

Если удерживать кнопку **Alt** key во время масштабирования, это приведет к тому, что размер компонента будет изменен относительно его центра, а не верхней левой точки.

У корневой панели 1042 никогда не бывает полей (она должна занимать все доступное пространство в окне виджета).

8.22.8.2 Редактирование абсолютного позиционирования

Редактирование абсолютной разметки гораздо проще редактирования табличной разметки. Перетащите компоненты мышью, и их координаты X и Y обновятся соответствующим образом. Для наиболее точного перетаскивания используйте клавиши с изображением стрелок.

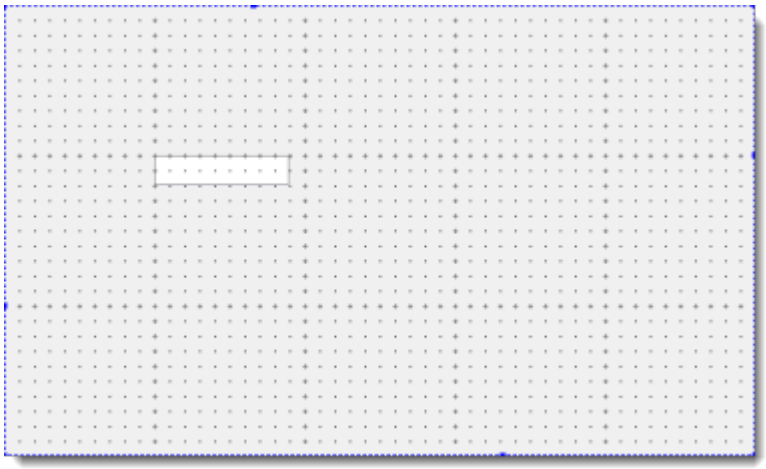
Любой выбранный компонент можно масштабировать при помощи мыши. Кроме того, поддерживается одновременное перемещение/мастабирование выбранных компонентов.

Привязывание сетки

Часто бывает удобно сохранять обычное расстояние между компонентами в пределах абсолютной компоновки. Виджет будет выглядеть лучше, если компоненты правильно выровнены относительно друг друга. И хотя панель инструментов 424 AtomMind GUI Builder предлагает множество средств для выравнивания компонентов, один из лучших способов обеспечить правильное выравнивание - это использовать **привязывание сетки**.

Не путайте привязывание сетки абсолютной компоновки с компоновкой сетки 950.

Привязывание сетки видно только если виджет редактируется в AtomMind GUI Builder. У него два шага: первый шаг, который может опционально использоваться для привязывания компонентов во время операций перетаскивания, и второй шаг, который всегда в десять раз больше первого шага.



Привязывание сетки можно контролировать отдельно в каждом контейнере, использующем абсолютную разметку. Существует три свойства компонентов 955, контролирующих привязывание сетки:

- **Показать сетку**. Включает/выключает видимость привязывания сетки.
- **Шаг сетки**. Главный шаг привязывания сетки в пикселях.
- **Пивязать к сетке**. Если включен, все углы компонента прикрепляются к узлам сетки во время операций перетаскивания 441.

Несмотря на то, что функции управления привязыванием сетки имеются в окне свойств и сохраняются в шаблоне виджета, как и любые другие свойства, они не влияют на функционирование действующего виджета.

Единственной целью привязывания сетки является облегчение размещения компонентов в AtomMind GUI Builder.

8.22.8.3 Рисование графических примитивов

Графические примитивы 1268 можно рисовать только в контейнерах 1041 с абсолютной разметкой 954. Для создания нового примитива выберите контейнер с абсолютной разметкой, затем выберите тип примитива в панели инструментов, находящейся в правой части рабочей формы 426. После этого:

- Несколько раз кликните внутри выбранного контейнера для настройки опорных точек примитива, у которого больше двух таких точек (например, ломаная линия или многоугольник). Закончите примитив двойным щелчком.
- Кликните и потяните внутри выбранного контейнера для создания примитива, у которого две опорные точки (например, прямоугольник или эллипс).
- Нарисуйте идеальный горизонтальный, вертикальный или диагональный примитив, удерживая Shift во время сдвига. Удерживайте Ctrl, чтобы повернуть ваш примитив под углом в соответствии с координатами сетки.

Как только создание примитива закончилось, он становится стандартным компонентом виджета, который, как и любые другие компоненты, может перемещаться или изменяться в размере. У него есть имя, которое может меняться и на которое можно ссылаться из привязок виджета 1295, и, как только оно выбрано, его свойства можно редактировать в окне Свойства компонентов 431.

Для переключения работы с режима создания примитивов на режим выбора примитивов, выберите иконку  в панели инструментов по созданию примитивов.

8.22.9 Управление привязками

AtomMind GUI Builder помогает создавать привязки 1295 для виджета, используя интуитивно-понятные визуальные операции. Существует несколько способов создавать привязки и управлять ими:

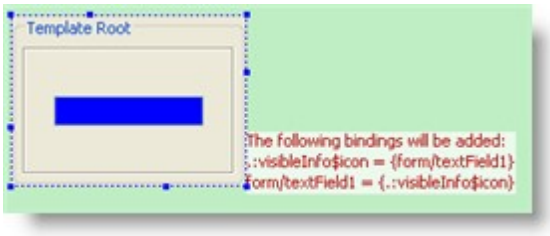
- Создание путем перетаскивания мышью
- Изменение привязок компонентов через контекстное меню компонентов
- Изменение привязок свойства через контекстное меню свойств.

Создание привязок путем перетаскивания мышью

Можно перетащить узел из Селектора объектов 430 на компонент или контейнер в рабочей форме 426 или в окне ресурсов 428.

ПРОСМОТР ПРИВЯЗОК ПЕРЕД ИХ СОЗДАНИЕМ

При перетаскивании узла с селектора объектов на рабочую форму, если мышь задерживается над компонентом формы, привязки отображаются в панели инструментов красного цвета. Левая часть привязки (до символа "=") является целью привязки 1295, в то время как правая часть представляет собой выражение 1296 привязки:



Если затем узел перетаскивается на компонент, создаются новые привязки. Только что созданные привязки отображаются во всплывающем окне, и их можно настраивать.

Одновременное редактирование всех привязок виджета

Для просмотра и редактирования всех привязок виджета в одном диалоговом окне следует:

- Открыть свойства корневой панели 1042 в окне свойства компонента 431
- Открыть свойство таблицы **Все Привязки** и редактировать его
- Кликните OK, чтобы сохранить изменения

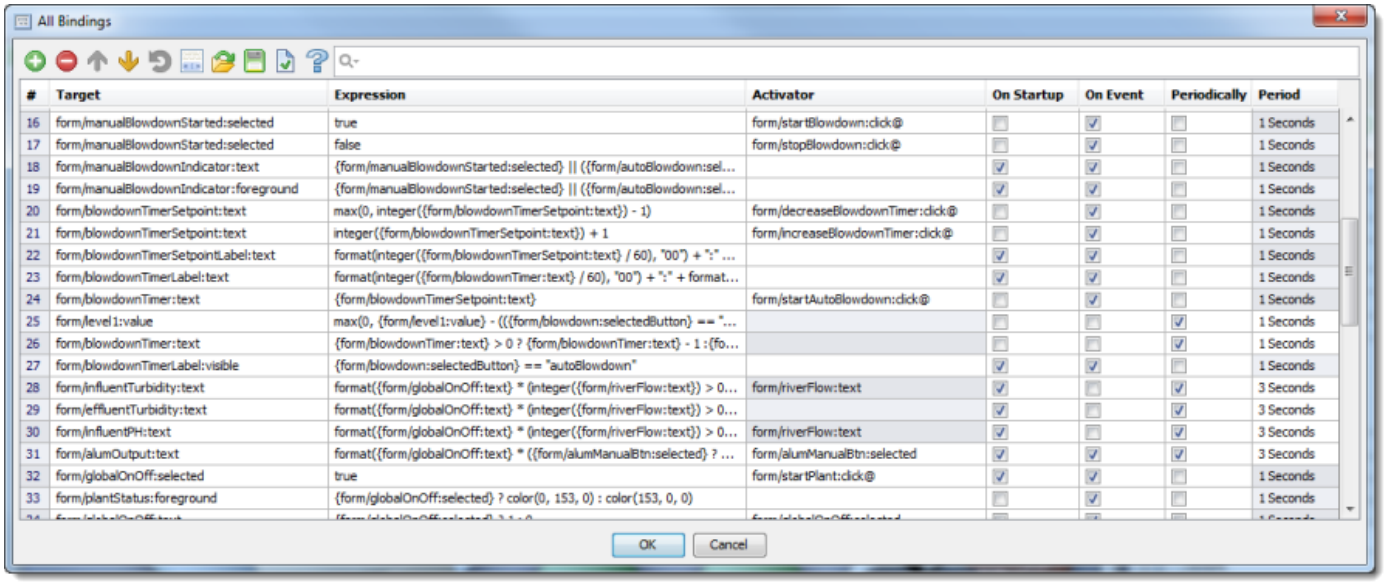


Таблица "Все Привязки" поддерживает функцию сортировки и фильтрации всего списка привязок.

Изменение привязок компонента

Чтобы получить доступ к управлению привязками, относящимися к определенному компоненту, щелкните правой кнопкой мыши в Окне Ресурсов 428 или Рабочей формы 426 и выберите **Редактировать Привязки**. После этого всплывет окно привязок компонента.

Кроме того, можно редактировать таблицу **Привязок** в окне Свойства компонента 431. Она также дает доступ ко всем привязкам, относящимся к данному компоненту.

Привязка свойства

Чтобы создать новую привязку, нацеленную 1295 на определенное свойство компонента, кликните правой кнопкой мыши по имени этого свойства в окне Свойств компонента 431 и выберите **Привязать Свойство**. Появится диалоговое окно с настройками привязки, позволяющее установить выражение привязки, активатор и опции выполнения.

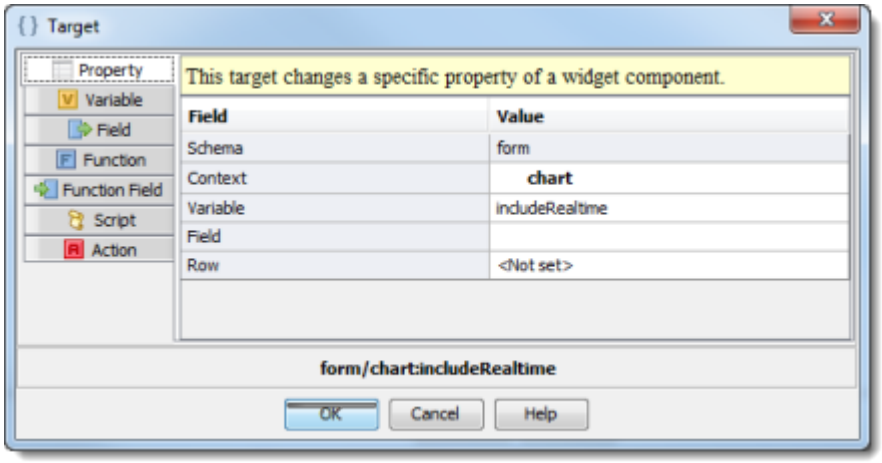
Изменение привязок свойств

Чтобы управлять всеми привязками, относящимися к свойству одного компонента, кликните правой кнопкой мыши по имени этого свойства в окне Свойства компонента 431 и выберите **Редактировать Привязки**. Появится окно с привязками свойства.

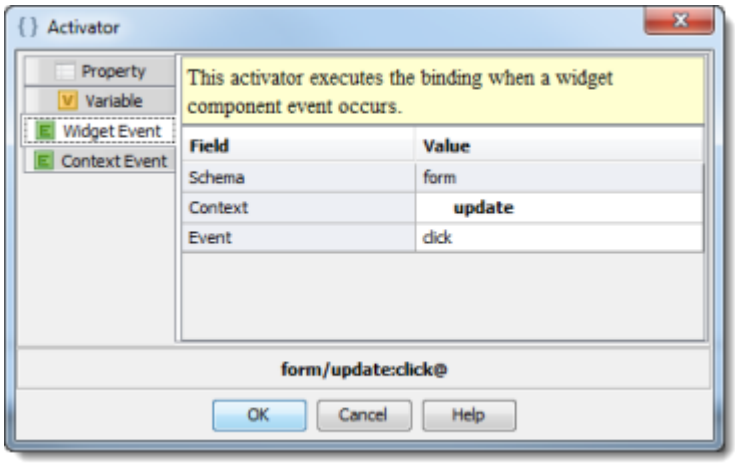
Редактирование привязки

Различные части привязки можно редактировать в разных визуальных редакторах.

- Цель 1295 привязки редактируют в **Редакторе Цели Привязки**:



- Выражение привязки редактируют в редакторе выражений
- Активатор привязки редактируют в **Редакторе Активатора Привязки**:



8.22.10 Операции перетаскивания мышью

В этом разделе говорится об операциях по перетаскиванию мышью, которые во многом упрощают привязывание свойств компонентов к элементам модели данных и друг к другу.

Операция перетаскивания мышью между Селектором Объектов и Рабочей Формой

1. ПЕРЕТАСКИВАНИЕ ПОЛЯ ПЕРЕМЕННОЙ НА КОМПОНЕНТ

В этом случае создаются две привязки:

- Привязка, которая читает поле переменной при загрузке и сбрасывает событие корневой панели, а также записывает его в свойства по умолчанию компонента.
- Привязка, которая записывает значение свойства компонента по умолчанию назад в контекст, при возникновении события отправки данных корневой панели.

	Target	Expression	Activator	On Startup	On Event	Periodically	Evaluation Period (seconds)
1	.:childInfo$description	{form/textField1}	form/templateRoot#submit	☐	☑	☐	0
2	form/textField1	{.:childInfo$description}	form/templateRoot#reset	☑	☑	☐	0

2. ПЕРЕТАСКИВАНИЕ ПОЛЯ ВВОДА ФУНКЦИИ НА КОМПОНЕНТ

Эта операция создает одну привязку. Ее целью является поле ввода функции. Ее выражение - это ссылка на свойство по умолчанию компонента виджета. Ее активатор - событие отправки корневой панели.

	Target	Expression	Activator	On Startup	On Event	Periodically	Evaluation Period (seconds)
1	.:create()$description	{form/textField1}	form/templateRoot#submit	☐	☑	☐	0

3. ПЕРЕТАСКИВАНИЕ ПОЛЯ ВЫВОДА ФУНКЦИИ НА КОМПОНЕНТ

Эта опеция также создает единичную привязку к цели, указывающей на свойство компонента. Выражение привязки - это ссылка, указывающая на поле вывода функции. Активатор[1296] этой привязки - событие отправки[1044] корневой панели.

	Target	Expression	Activator	On Startup	On Event	Periodically	Evaluation Period (seconds)
1	form/textField1	{events:getHistory()$level}	form/templateRoot#submit	☐	☑	☐	0

4. ПЕРЕТАСКИВАНИЕ ПЕРЕМЕННОЙ НА КОМПОНЕНТ РЕДАКТОР ТАБЛИЦ ДАННЫХ[979]

Эта операция создает три привязки:

- Привязка, которая читает значение переменной при загрузке виджета и записывает ее как свойство по умолчанию Редактора Таблиц Данных, т.е. той таблицы, с которой в текущий момент работает редактор.
- Привязка, которая записывает таблицу, содержащуюся в Редакторе Таблиц Данных, назад в контекст во время события отправки[1044] корневой панели.
- Привязка, которая устанавливает включенное[1275] свойство Редактора Таблиц Данных в соответствии с **Доступным_для_записи** свойством определения переменной для того, чтобы убедиться, что значение переменных **Только_для_чтения** нельзя редактировать.

	Target	Expression	Activator	On Startup	On Event	Periodically	Evaluation Period (seconds)
1	form/dataTableEditor1#enabled	{users.admin:status#writable}		☑ On Startup (onstartup)		☐	0
2	form/dataTableEditor1	{users.admin:status}		☑	☐	☐	0
3	users.admin:status	{form/dataTableEditor1}	form/templateRoot#submit	☐	☑	☐	0

5.ПЕРЕТАСКИВАНИЕ ФУНКЦИИ НА КОМПОНЕНТ КНОПКА[968]

Эта операция не создает новые привязки. Напротив, она находит все привязки виджета:

- Чьи **цели** указывают на поле ввода данной функции или
- Чьи **выражения** содержат ссылки на поля вывода данной функции и устанавливает их **активаторы** на событие действие[971] компонента кнопки. Это заставляет их работать вместе в рамках одной сессии обработки привязок[1296], которая начинается после щелчка мышью по кнопке (когда событие **действие** активно).

Если у виджета нет привязок, относящихся к полям ввода или вывода перетаскиваемой функции, то перетаскивание мышью не будет позволено.

6. ПЕРЕТАСКИВАНИЕ ПОЛЯ ПЕРЕМЕННОЙ НА ГРУППУ КНОПОК[1040]

Эта операция создает три привязки, которые позволяют использовать кнопки в группе для изменения значения поля:

	Target	Expression	Activator	On Startup	On Event	Periodically	Evaluation Period (seconds)
1	form/buttonGroup1#format	{.all_device_servers:childInfo$name#format}	form/templateRoot#reset	☑	☑	☐	0
2	.all_device_servers:childInfo$name	{form/buttonGroup1}	form/templateRoot#submit	☐	☑	☐	0
3	form/buttonGroup1	{.all_device_servers:childInfo$name}	form/templateRoot#reset	☑	☑	☐	0

Target (target)

Если у поля есть выбор значения[49], эта операция также изменяет свойство значения[974] всех кнопок-переключателей, которые принадлежат к группе доступных значений для выбора. Если у Вас есть, например, 5 значений выбора, и лишь три кнопки-переключателя, лишь три первых значения будут использованы, и система не покажет ошибку.

7. ПЕРЕТАСКИВАНИЕ ПОЛЯ ПЕРЕМЕННОЙ НА КОМБИНИРОВАННУЮ ЯЧЕЙКУ[975] ИЛИ СПИСОК[964]

Эта операция создаст те же самые привязки, что и операция **1,** описанная выше[441], (Перетаскивание поля переменной A на компонент), но в то же время она добавляет дополнительные привязки, цель которых заключается в отображении выбор значения[49], определенных форматом поля в комбинированной ячейке или списке.

8. ПЕРЕТАСКИВАНИЕ КОНТЕКСТА DEVICE НА ЛЮБОЙ ИЗ КОНТЕЙНЕРОВ

Эта операция перетаскивания создает новый Компонент устройства[1023] в целевом контейнере. У нового компонента будут ярлыки, указывающие на основной статус перетаскиваемого Device. У него будет также

Таблица с заранее определенным Статусом, настроенная на то, чтобы указывать текущий статус соединения устройства (Online, Offline или Suspended).

Кроме того создается статус привязки устройства к серверу с параметром **Статус** компонента устройства.

8.22.11 Автономная версия

В этом разделе говорится об автономной версии Widget editor и как начать ее использовать.

Автономная версия Widget editor имеет тот же функционал, что и Widget editor, и позволяет использовать все его функции в вашем веб браузере. Получить доступ к Автономной версии Widget editor можно через AtomMind Web UI.

Чтобы начать работу с Автономной версией Widget editor:

- Получите доступ к AtomMind Web UI. Более подробно см. разел Web UI 220.
- Кликните правой кнопкой мыши по любому из ваших виджетов и выберите `Edit Template`.
- Вы будете перенаправлены на страницу загрузки Автономной версии Widget editor. Система автоматически определит вашу ОС и версию вашего AtomMind Server. После этого, система даст вам ссылку на скачивание подходящего файла установки Автономной версии Widget editor.
- Скачайте и установите этот файл.

После этого вы сможете использовать Автономную версию Widget editor и получите доступ к функционалу Widget editor из вашего AtomMind Web UI.

8.23 AtomMind IDE

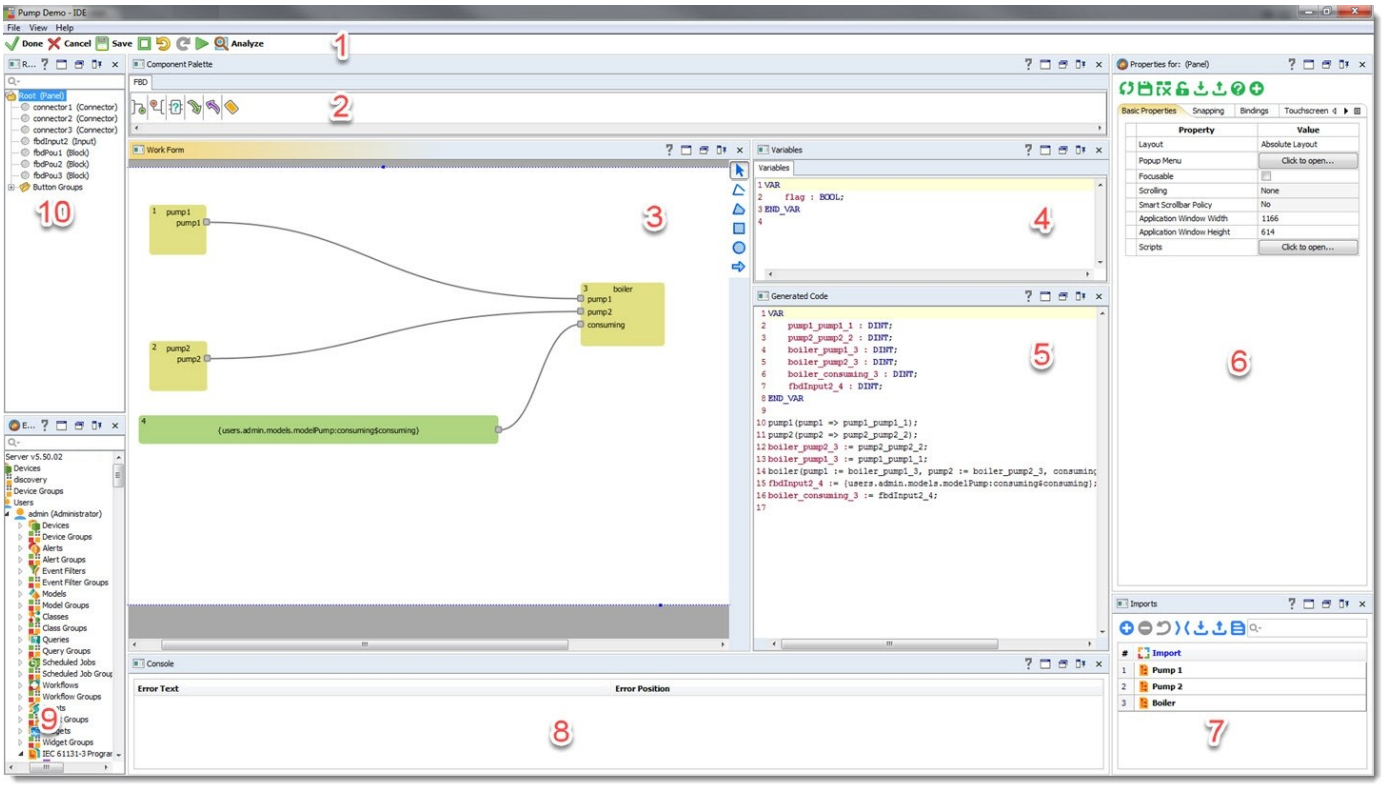
AtomMind IDE - это часть AtomMind Client 359, которая используется для редактирования программ на языках стандарта Управление процессами 1983.

Среда программирования AtomMind IDE включает набор инструментов для написания, анализа и отладки программных компонентов 444.

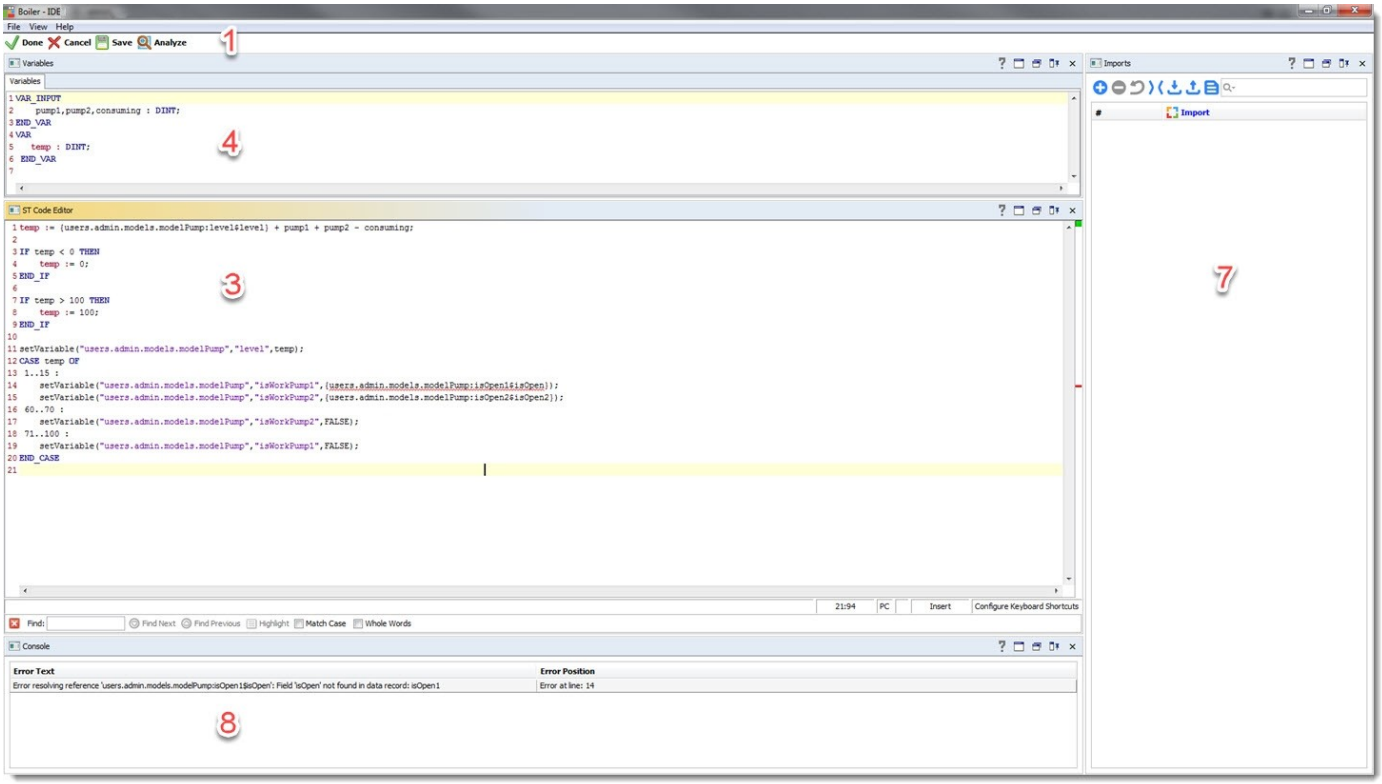
Главное окно

AtomMind IDE всегда открывается в отдельном окне. Редактор имеет два режима работы:

1 - Редактирование и разработка графических программных компонентов:



2 - Редактирование и разработка текстовых программных компонентов:



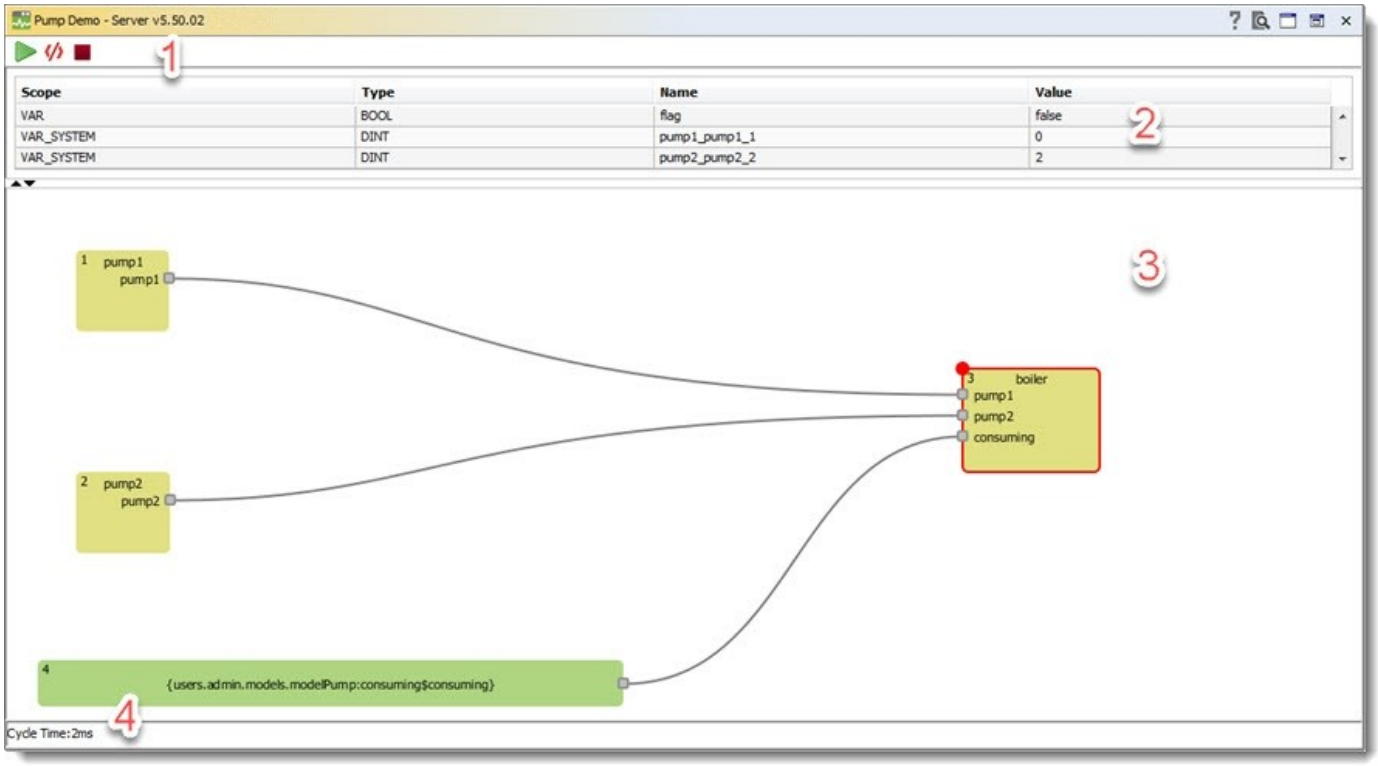
1. Панель инструментов 452
2. Палитра компонентов 450
3. Рабочая форма 448
4. Область объявления переменных 449
5. Сгенерированный код 452
6. Окно свойств компонентов 431
7. Таблица импортов 451
8. Консоль 452
9. Селектор объектов 430
10. Окно ресурсов 428

Большинство компонентов AtomMind IDE представляют собой перемещаемые панели, которые можно свернуть/развернуть, удалить, поместить в закладки, изменить их размер и пр. См. Настройка плавающих панелей 366.

8.23.1 Отладчик

Отладчик - средство для отладки программ Управление процессами 1983. Позволяет запустить программу в режиме отладки и проанализировать значения переменных, а также использованных программ, функций и функциональных блоков. Дает возможность остановки на конкретном шаге или строчке программы и прохождению программы по шагам.

В режиме отладки может быть запущена программа, которая при создании была отмечена как "Задача", и для неё задан цикл выполнения.
Программа не может быть запущена до тех пор, пока она не будет проанализирована и сохранена без ошибок.



Для текстового языка "рабочая область" №3 будет заменена на редактор кода 409, допускающий только просмотр кода программы.

Оба вида отладчика (текстовый и графический) предусматривают установку точек остановки по двойному клику на компоненте или на номере строки. После запуска программы ее выполнение продлится до ближайшей точки. После остановки в точке пользователь сможет провести отладку программы шаг за шагом. Удаление точки происходит по двойному клику по ней.

Элементы отладчика

1 КНОПКИ УПРАВЛЕНИЯ

	Запуск. Запускает циклическое выполнение программы.
	Шаг. Переход на следующий шаг выполнения программы.
	Стоп. Остановка выполнения программы.

2 ОБЛАСТЬ ВЫЧИСЛЕНИЯ ПЕРЕМЕННЫХ

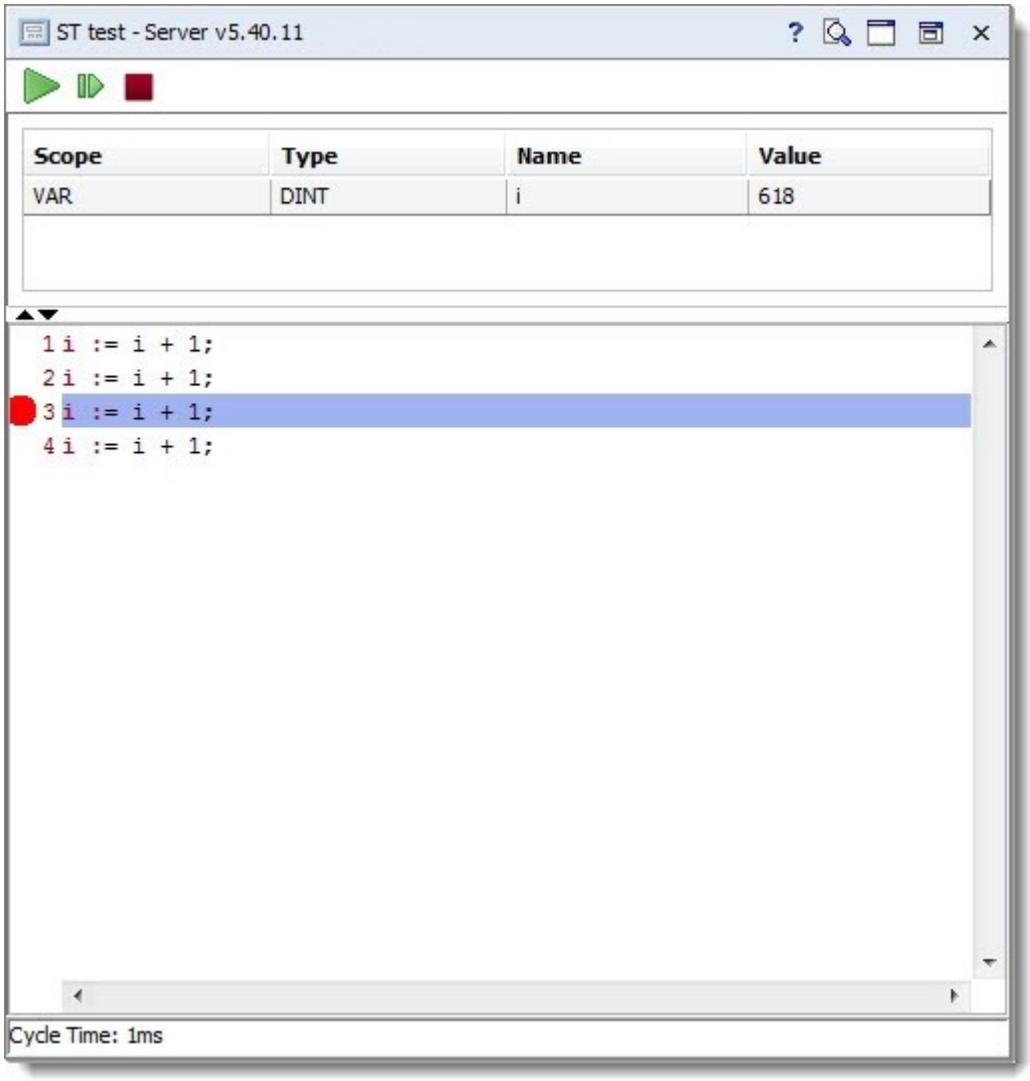
- В области переменных отражаются все переменные, используемые в программе. Здесь будут отображаться как локальные переменные, так и входные и выходные переменные функциональных блоков и функций.

- На каждом шаге программы значения переменных вычисляются и заносятся в таблицу (2).

- Точка останова останавливает выполнение программы на определенном шаге и позволяет проанализировать значения переменных на каждом шаге.

3 РАБОЧАЯ ОБЛАСТЬ

Отладчик может содержать как текстовую так и графическую рабочие формы 448. Оба случая предполагают исключительно просмотр программы.



Точки останова – это места, в которых выполнение программы будет приостанавливаться, что позволяет просмотреть значения переменных на определенном этапе работы программы. Точки останова можно задавать во всех редакторах:

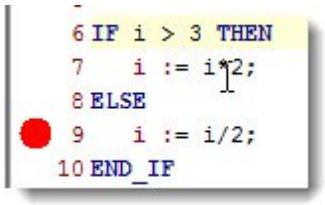
- В текстовом редакторе точка останова устанавливается на номер строки;
- В FBD - на графический элемент;
- В SFC - на шаг.

Пошаговое выполнение. Под «шагом» подразумевается:

- В ST: Выполнить следующую инструкцию;
- В FBD: Выполнить следующую цепь;
- В SFC: Продолжить действие до следующего шага.

Пошаговое выполнение позволяет проверить логическую правильность программы.

Перед добавлением точки происходит проверка места установки. Пример:



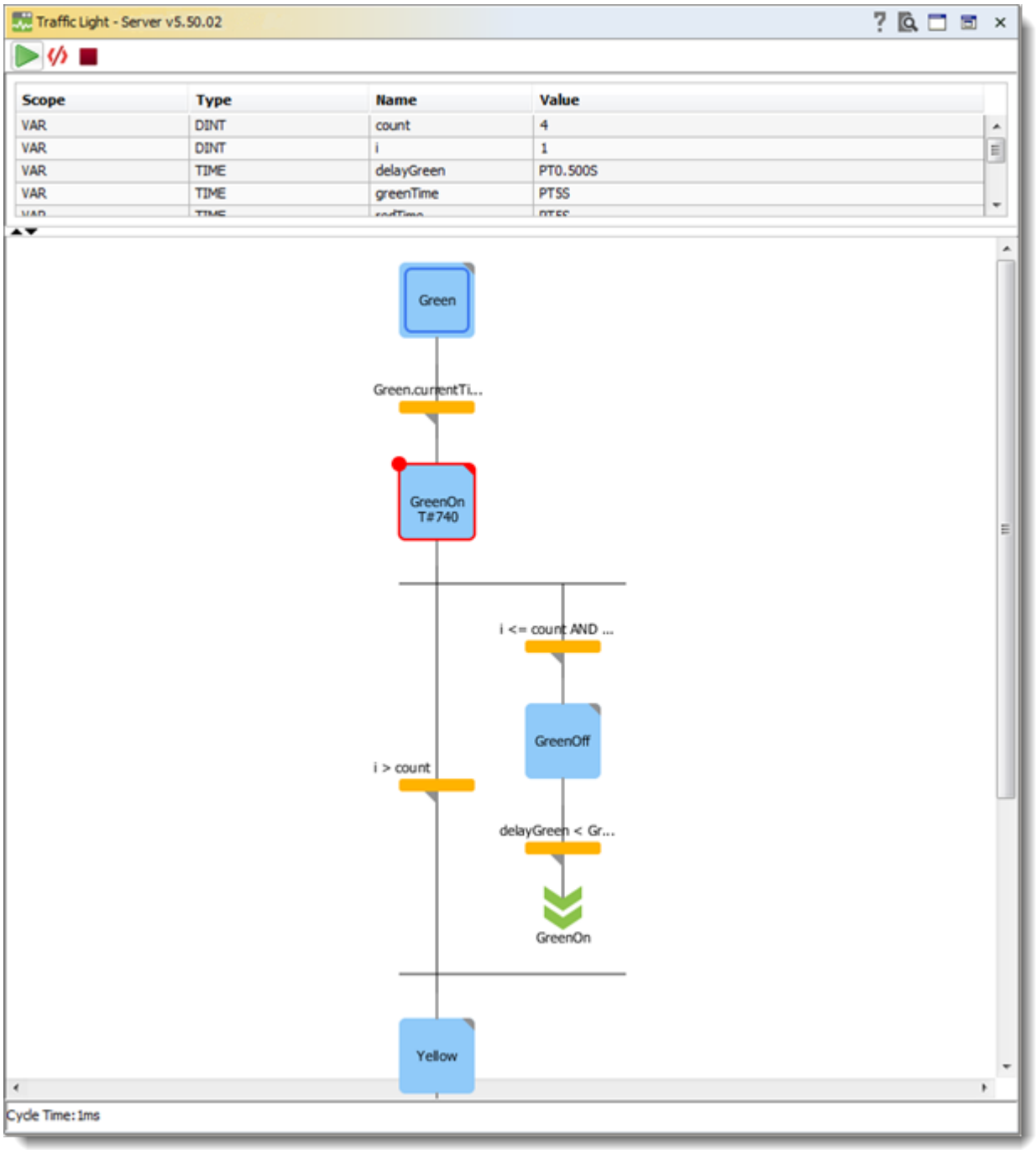
Точка не может быть установлена на строку 8, т.к. нет выражения для вычисления. При попытке установить точку на строку 8, она автоматически будет установлена на следующую по порядку строку с выражением (т.е. строка 9).

4 КОНТРОЛЬ ВРЕМЕНИ ЦИКЛА

Показывает примерное необходимое время для выполнения одной итерации программного компонента.

SFC В РЕЖИМЕ ОТЛАДКИ

В режиме отладки активные шаги изображаются синим цветом. Внутри шага под его именем выводится время активности этого шага в миллисекундах. На примере показано, что шаг активен 1 секунду и 500 миллисекунд.

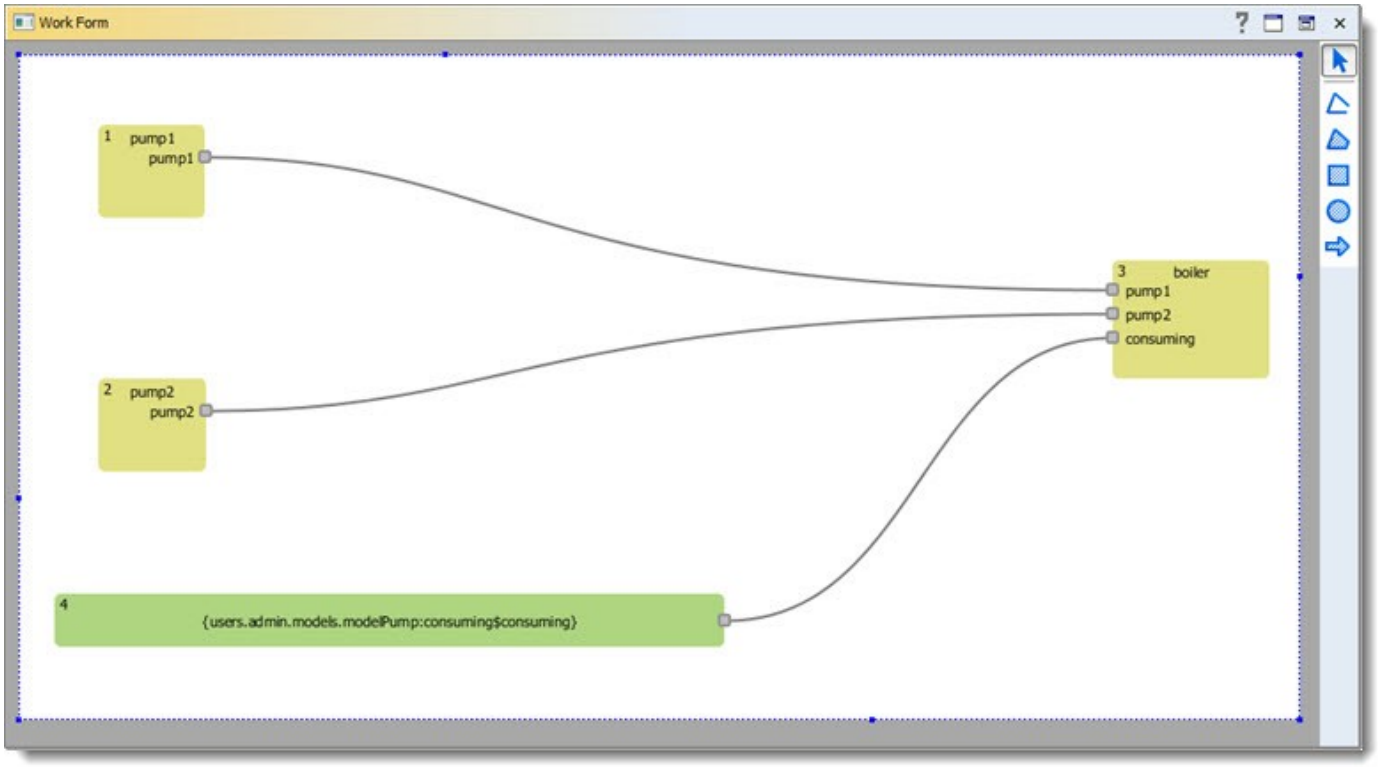


8.23.2 Рабочая форма

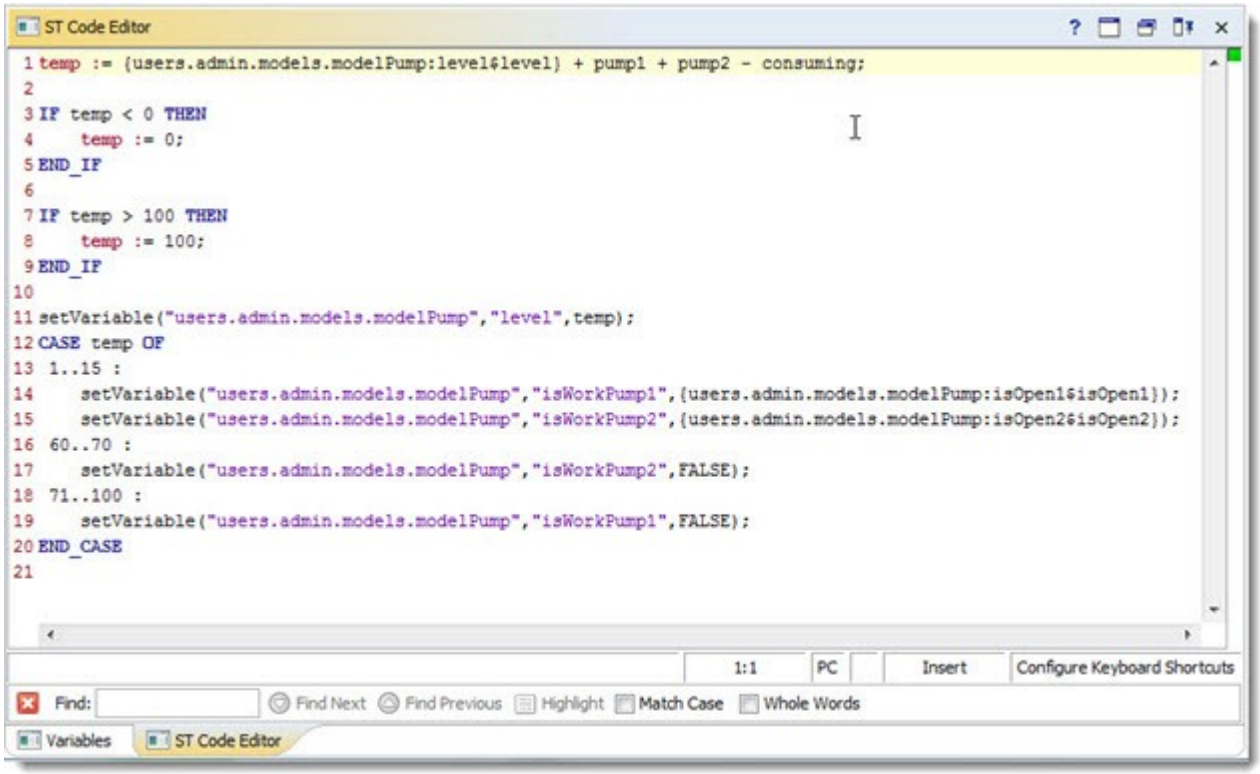
Рабочая форма - основная область для построения схем и диаграмм, а так же написания кода.

В соответствии с режимами работы AtomMind IDE, рабочая область принимает вид:

- Рабочая форма 426 служит для редактирования графических языков. Принцип работы аналогичен рабочей форме в редакторе виджетов 426.



- Редактор кода 409 служит для редактирования программ на языке ST 1994 и переменных в области переменных 449 AtomMind IDE.

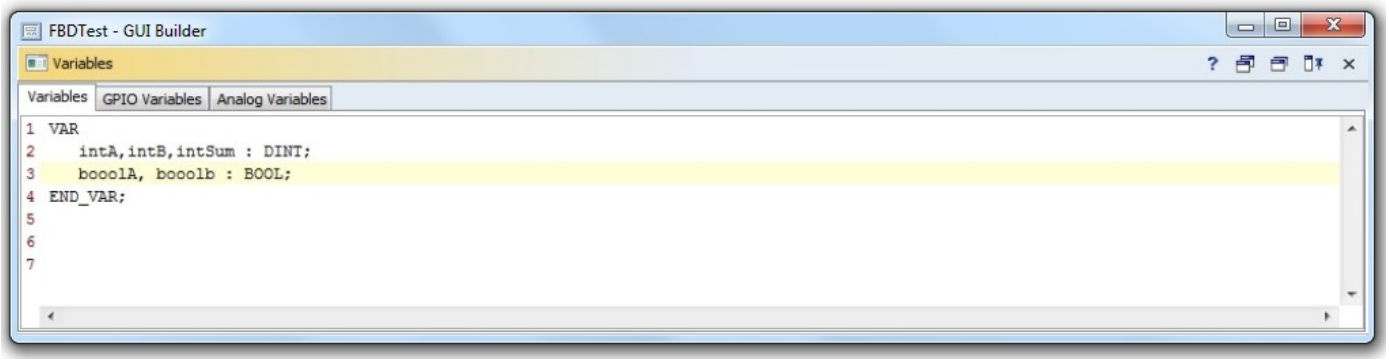


8.23.3 Область переменных

Область переменных Управление процессами является редактором кода 409 для языка ST 1994 с анализатором и встроенными подсказками.

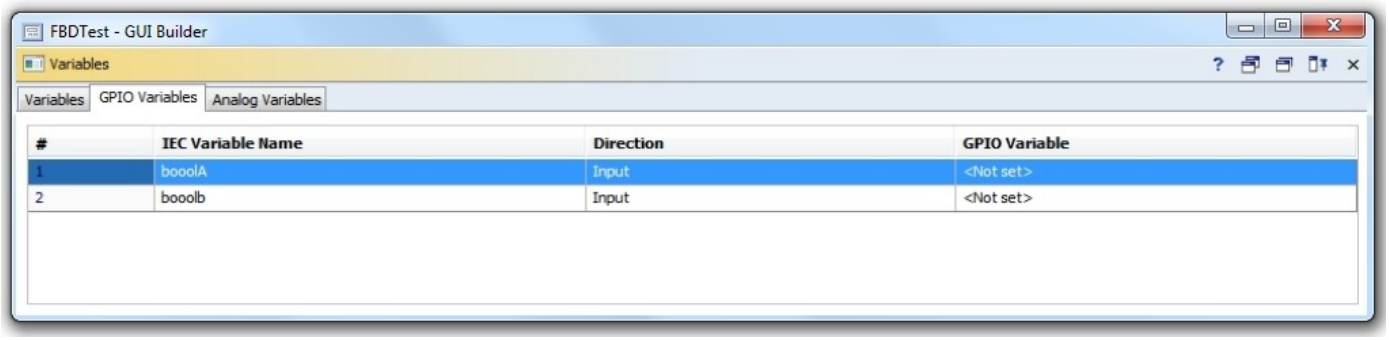
Содержит информацию о:

- переменных 1986 программного компонента.



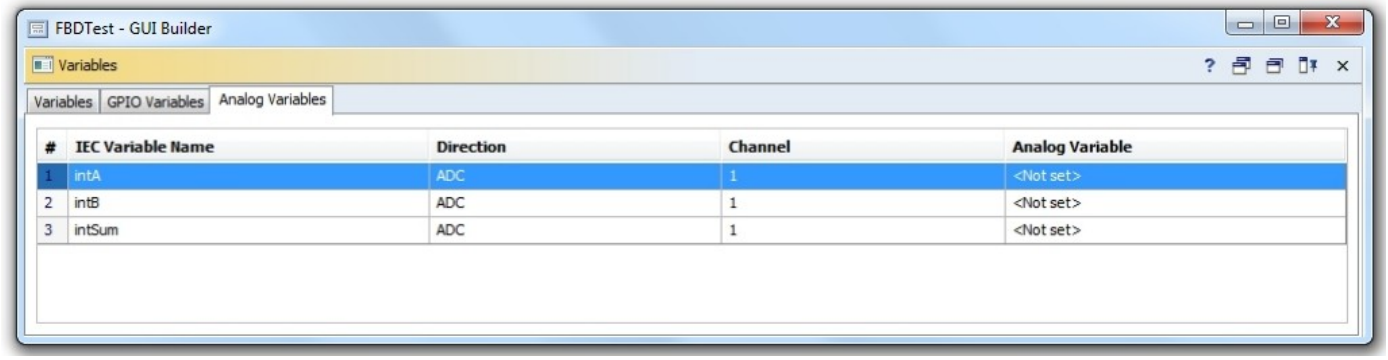
В дальнейшем они будут использоваться в теле программы для вычислений. После создания переменной и анализа программы она станет доступна пользователю и будет добавлена в список всплывающих подсказок.

- GPIO (интерфейс ввода/вывода общего назначения) переменных устройства.



Это низкоуровневый интерфейс ввода-вывода прямого управления. Через этот интерфейс устройство, подключенное в AtomMind, может слушать и отдавать команды любому внешнему устройству, такому как датчики, диоды и проч.

- ADC (аналого-цифровой преобразователь) аналоговых переменных устройства.

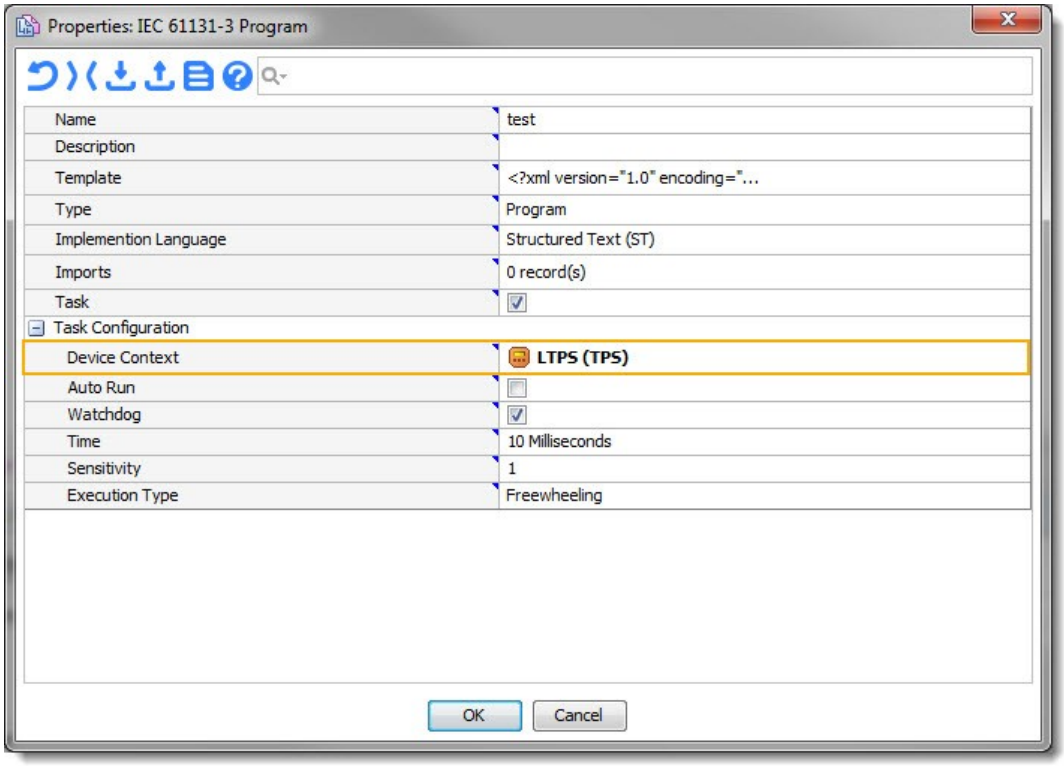


Если устройство содержит ADC, преобразующий входной аналоговый сигнал в дискретный код (цифровой сигнал), вы можете использовать переменные соответствующего диапазона.

Обычно от -10 000 до 10 000 мВ.

Использование

Переменные описываются на языке ST 1994, их значения могут быть присвоены GPIO и аналоговым переменным, а также самой программе. Переменные устройства могут быть доступны, если при настройке программы был выбран контекст устройства.



8.23.4 Палитра компонентов

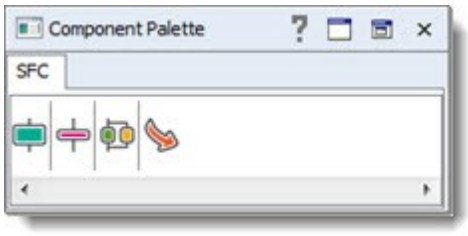
Палитра компонентов используется для добавления новых компонентов в разметку виджета путем перетаскивания их мышью в рабочую форму 448.

Программа на каждом из языков редактируется в необходимой ей схеме компоновки 950 на рабочей форме 448.

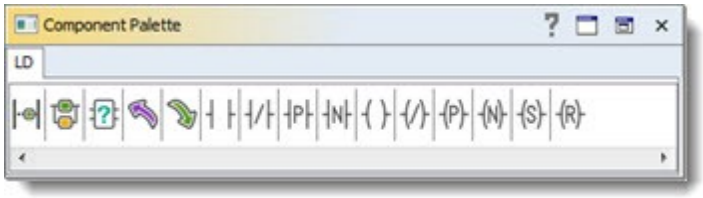
Для каждого из редактируемых языков палитра содержит различные компоненты:

- Компоненты языка FBD 457 используются в абсолютной компоновке

- Компоненты языка SFC 462 используются в табличной компоновке



- Компоненты языка LD 466 используются в табличной компоновке

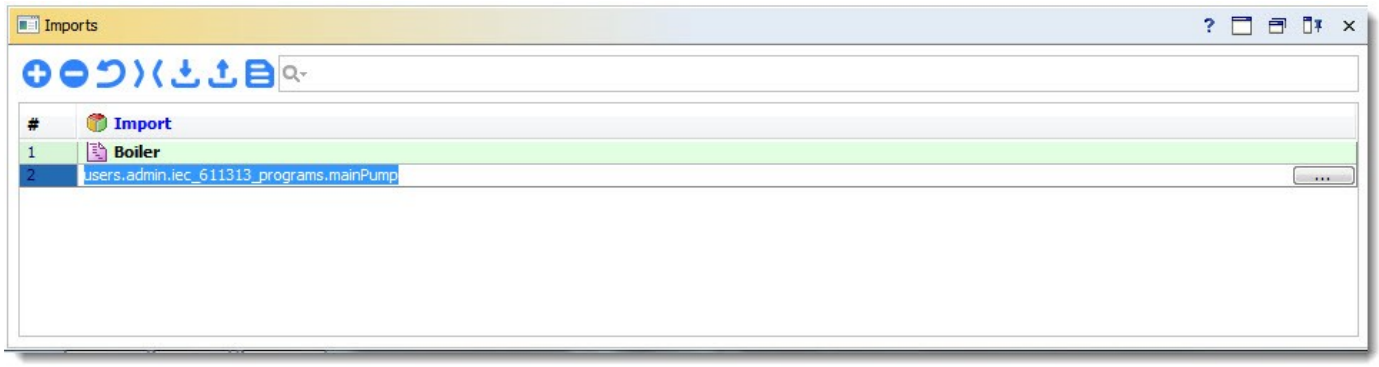


Все компоненты могут быть перемещены на форму посредством "drag and drop".

8.23.5 Панель импортов

Панель импортов необходима для добавления в программный компонент других программных компонентов, разработанных на языках Управление процессами. Импортированные программные компоненты могут использоваться пользователем в теле редактируемого программного компонента.

Для импортирования компонента необходимо добавить новую строку в таблицу.



После импорта компонента пользователю доступны переменные 1986 импортированного программного компонента или сам компонент в виде функции.

В области переменных 449 может быть создана переменная типа импортированного компонента если это функциональный блок 1992, через точку пользователь получает доступ к переменным блока.

1. Для программ на языке FBD 2003

Импортированный компонент может быть использован:

- в функциональном блоке, для этого ему необходимо задать имя соответствующее имени из импорта.

- в имени входа может быть написано выражение с использованием импортированного компонента.

2. Для программ на языке SFC 1998

Импортированный компонент может быть использован:

- в действии 2002. При написании выражения для использования функций.

- в переходе 1999. Для выражения перехода могут использоваться импортированные булевые функции.

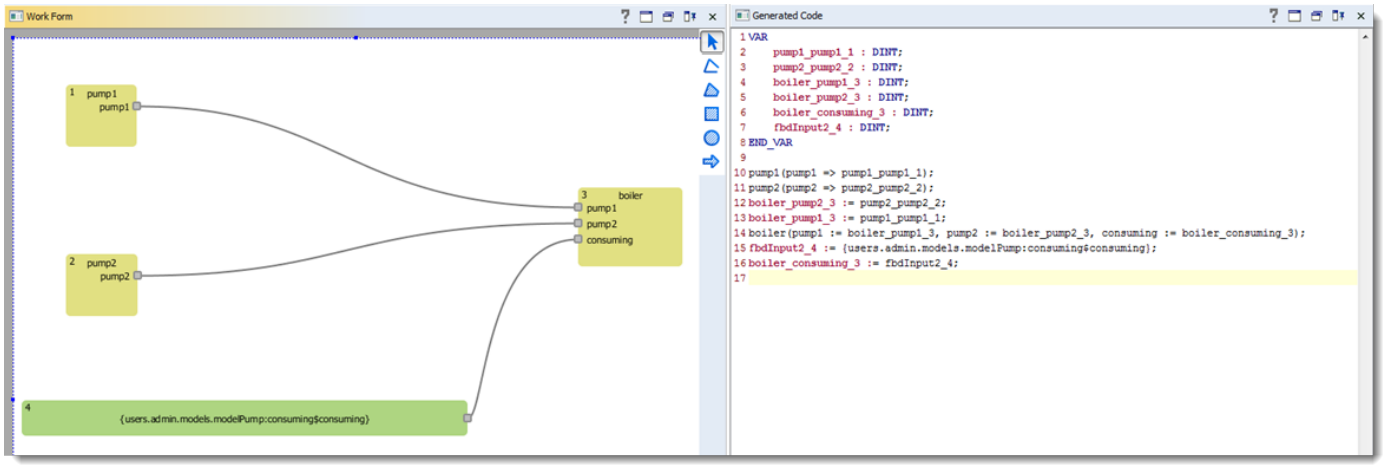
3. Для программ на языке ST 1994

Импортированный компонент может быть использован:

- в тексте программы. Переменные компонентов доступны через ".".

8.23.6 Сгенерированный код

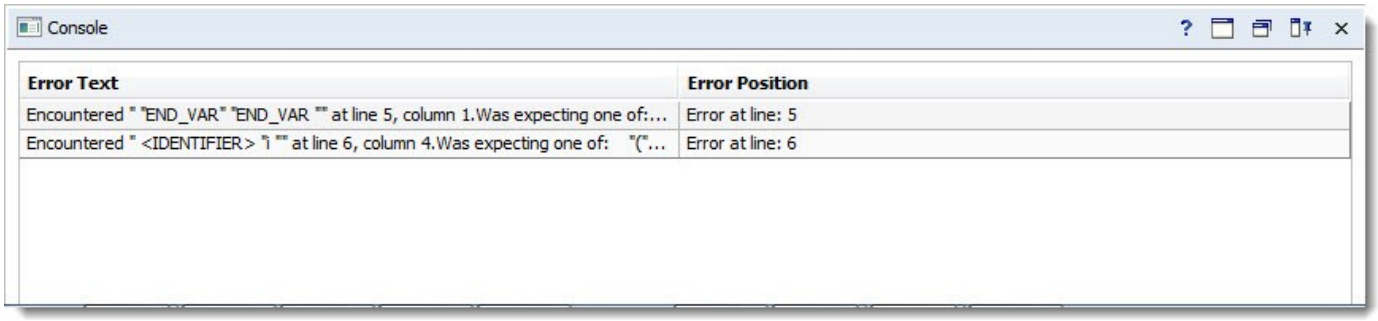
Этот фрайм показывает сгенерированный код для программ, написанных на FBD и LD:



Логически эти два фрагмента кода выполняют одно и то же, результаты их работы идентичны. Именно сгенерированный код выполняется на сервере.

8.23.7 Консоль

Консоль отображает ошибки выявленные после анализа программы. Анализ выполняется по нажатию кнопки "Analyze" в панели инструментов 452 или после нажатия клавиши "Enter" в текстовом редакторе.



В строке с ошибкой содержится:

- текст ошибки;

- местоположение ошибки.

1. Для программ на языке FBD 2003 и SFC 1998

В качестве положения выводится номер и имя блока соответственно.

2. Для программ на языке ST 1994

Выводится номер строки в котором содержится ошибка.

8.23.8 Панель инструментов

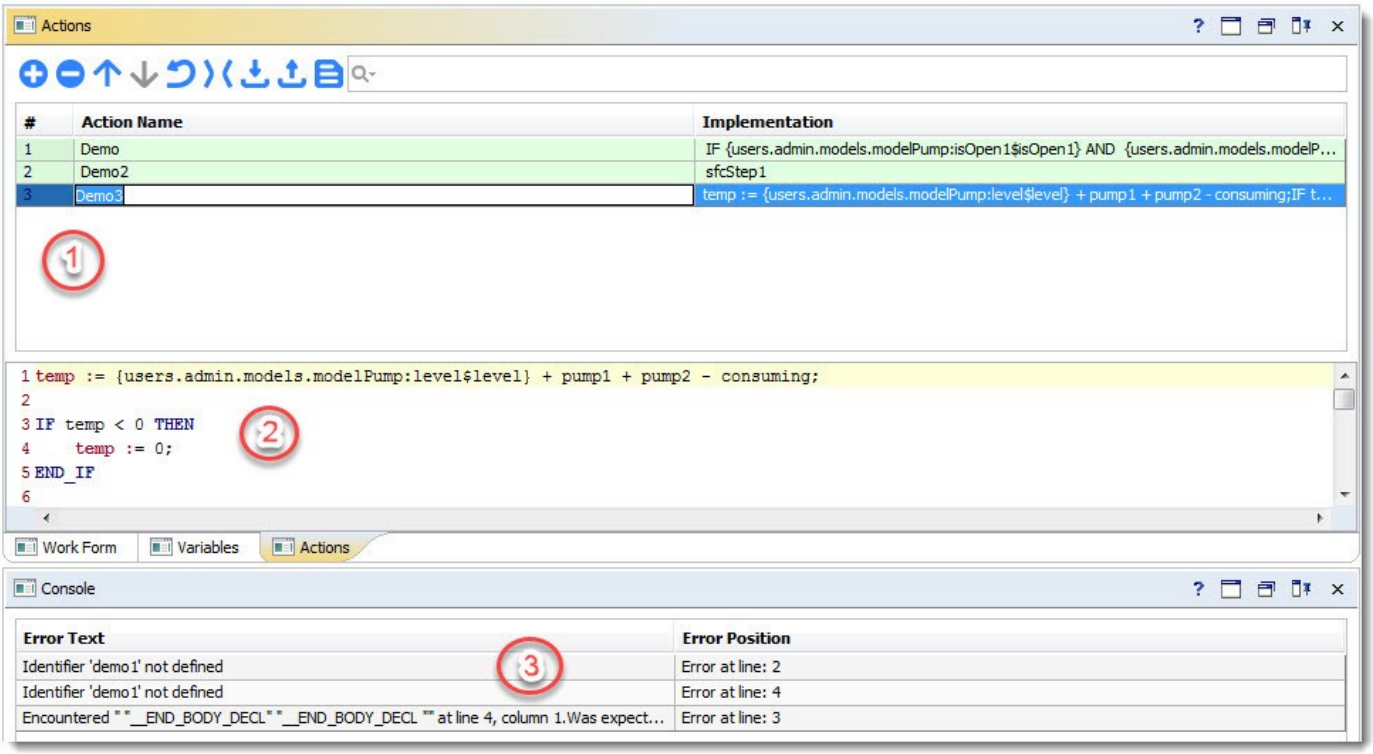
Панель инструментов предоставляет доступ к частым операциям над программой.

	**Готово**. Завершает редактирование шаблонов виджета и сохраняет изменения в AtomMind Server.
	**Отмена**. Отменяет редактирование и отбрасывает изменения.
	**Переключить декорации**. Включает или отключает оформление ячеек разметки панели. См. оформление ячеек 427.

	Сохранить шаблон. Сохраняет редактируемый шаблон на сервере обновлением поля *шаблон* в свойствах виджета 947. Редактирование не прерывается.
	Отменить. Отменяет предыдущую операцию. Для этой функции есть горячие кнопки **Ctrl-Z**.
	Восстановить. Повторяет предыдущую отмененную операцию. Для этой функции есть горячие кнопки **Ctrl-Y**.
	Запустить виджет. Запускает редактируемый виджет в режиме тестирования. У этой функции есть горячая кнопка: **F5**.
	Проанализировать программу. Выполняет статический анализ кодов.

8.23.9 Действия

Данное окно описывает действия языка SFC 2002. Действия играют ключевую роль в языке SFC 1998. Для добавления нового действия необходимо задать имя и само действие. Действие описывается только на языке ST 1994.



1. ТАБЛИЦА ДЕЙСТВИЙ

- В таблице отображаются доступные для использования и редактирования действия.

- При выборе строки действия, содержимое реализации будет доступно в редакторе кода "2".

2. РЕДАКТОР КОДА

- Используется для редактирования кода действия.

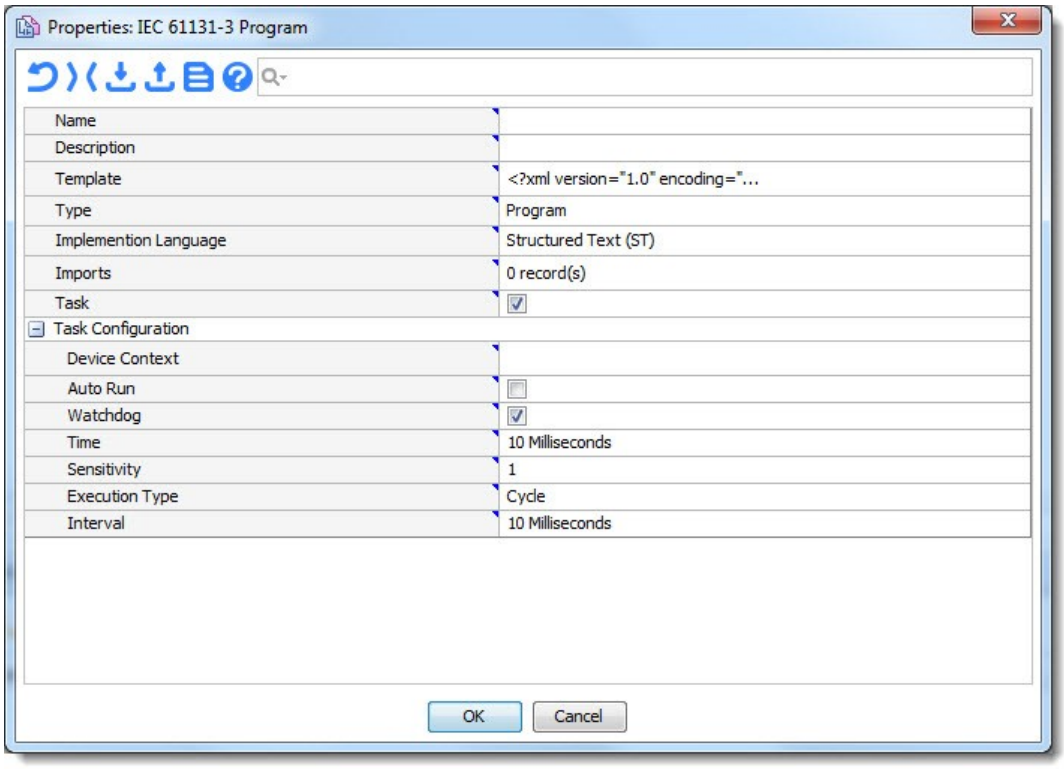
- При анализе программы анализируется так же код каждого действия.

3. КОНСОЛЬ

- Все ошибки в коде действий выводятся в консоль 452.

8.23.10 Конфигурация программы

Свойства программы могут быть просмотрены и изменены с помощью действия Настроить 1627 в контексте.



Описание поля	Имя поля
Имя программного компонента. Имя контекста. Должно удовлетворять соглашениям о наименовании 42 контекста. Имя необходимо для ссылки на программу из других частей системы.	name
Описание программного компонента. Текстовое описание 43 контекста.	description
Шаблон программного компонента. Текст шаблона 946 контекста в формате XML.	template
Тип программного компонента. Тип контекста, функция 1987, функциональный блок 1992 или программа 1993.	type
Язык программного компонента. Данное поле определяет язык реализации, он может быть SFC 1998, ST 1994 или FBD 2003.	implementationLanguage
Импорты. Список импортов программы.	imports
Задача. Если флаг активен, то эту программу можно выполнять и отлаживать.	task
Контекст устройства. Контекст по умолчанию, используемый при редактировании GPIO или аналоговых переменных в 449 AtomMind IDE.	deviceContext
Автозапуск. Данный флаг активирует запуск задачи при загрузке сервера.	autoRun
Сторожевой таймер. Активирует сторожевой таймер.	watchdog
Время. Время сторожевого таймера; если задача не завершена в этот период, она будет прекращена.	time
Чувствительность. Количество таймаутов сторожевого таймера до формирования сообщения об ошибке.	sensitivity

Тип исполнения. Данное поле определяет тип задачи, он может быть циклическим с заданным интервалом или с непрерывным циклом. Циклический : задача вызывается циклически через заданный интервал времени. Непрерывный цикл : задача вновь вызывается сразу же после окончания в непрерывном цикле, без задания времени цикла.	executionType
Интервал. Период времени, после которого задача должна быть вызвана в очередной раз.	interval

Данные свойства доступны через переменную childInfo 1628.

8.23.11 Компоненты

Данный раздел посвящен компонентам графических языков Управление процессами 1983.

8.23.11.1 Свойства компонентов

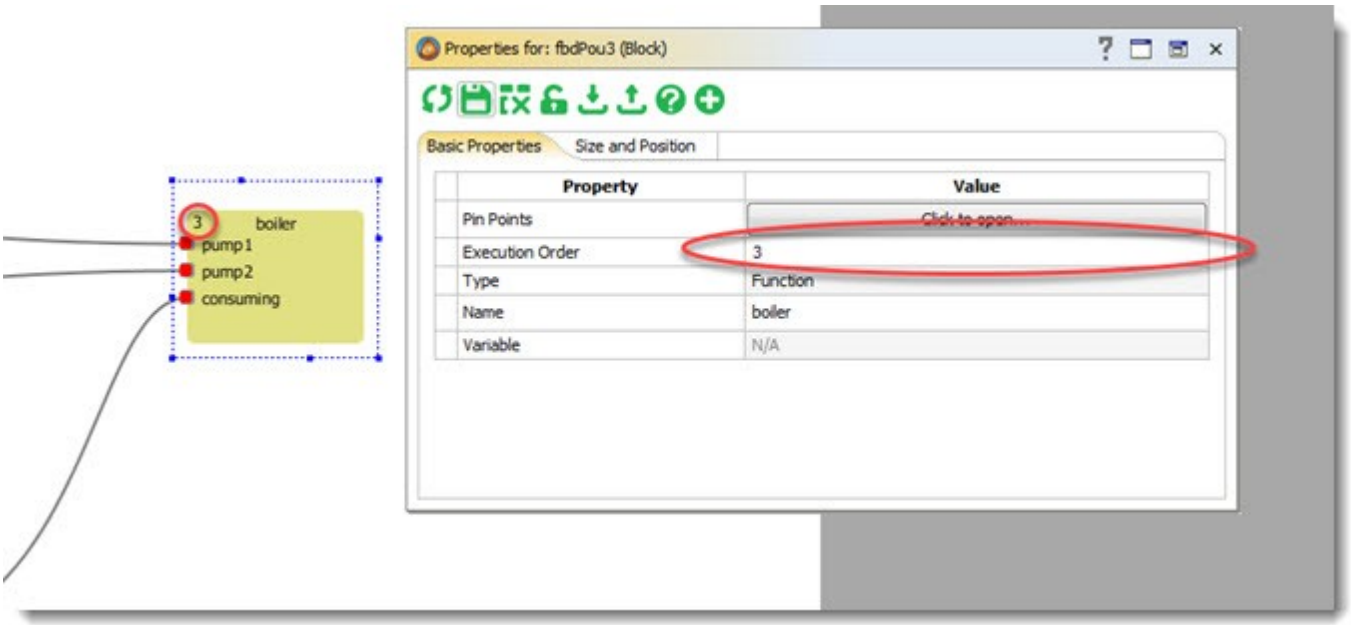
Каждый компонент обладает несколькими свойствами. Большинство компонентов обладает следующим набором общих свойств:

- Шрифт 1275
- Всплывающее меню 1276

Остальные свойства определяются типом компановки 950 рабочей области и его ограничениями 950.

8.23.11.1.1 Порядковый номер

Порядковый номер задает порядок выполнения схемы. Каждый элемент схемы обладает порядковым номером выполнения.
Номер элемента не изменяется при его перемещении. Каждому новому элементу присваивается следующий по порядку номер.



Установка номеров происходит автоматически после добавления компонента на форму или удаления компонента. Нумерация компонентов начинается с 1.

Добавление

- При добавлении элемента, ему будет присвоен следующий по порядку номер.

Удаление

- При удалении компонента, у всех следующих по порядку номер будет уменьшен на 1.

Ручное присвоение

- При ручном изменении номера компонента, у всех соседних компонентов номера будут изменены по принципу сдвига.

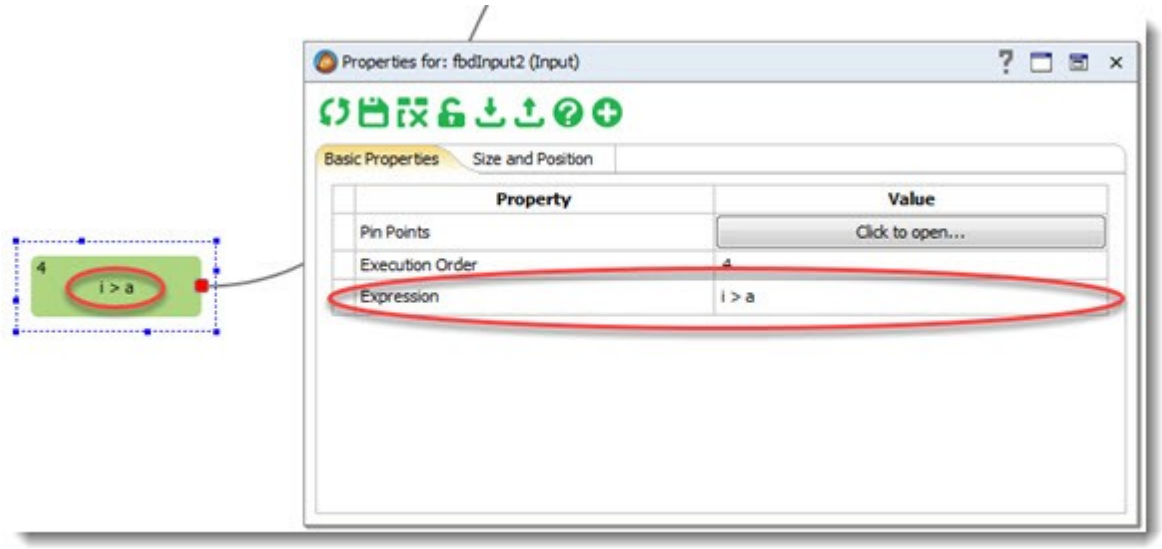
Запрещено устанавливать номер компонента больше чем кол-во элементов на форме.

Пример: На форме 7 компонентов. Изменим у второго номер на "5". Номера будут пересчитаны пос след. принципу:

1. компонент 2 = 5;
2. компонент 3 = 2;
3. компонент 4 = 3;
4. компонент 5 = 4;

8.23.11.1.2 Выражение

Выражение будет вычислено в момент выполнения элемента схемы.



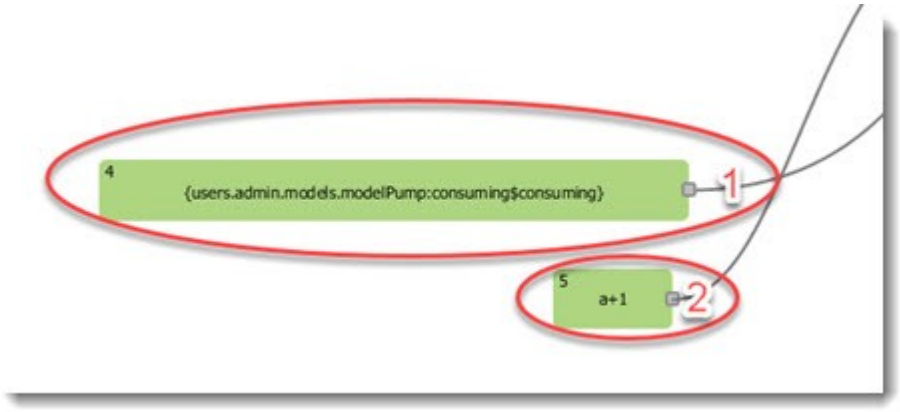
Выражения используются для упрощения процесса разработки.

Основное назначение выражений

Возможность использовать выражения на языке ST позволяет более компактно реализовать необходимую логику, что облегчает запись и чтение функциональных диаграмм.

Это также позволяет реализовать простую логику в блочном или переходном теле. Логика описывается на языке ST 1994. Использование выражений избавляет от необходимости создавать отдельную программу для реализации простой арифметики, логики (использование во входе 459) или операции присваивания (использование в выходе 459). Анализ схемы так же проводит проверку выражений на корректность. В случае появления ошибок, они будут выведены в консоль 452.

Примеры



1. `"{users.admin.models.modelPump:consuming$consuming}"` использование переменных контекста

2. `"a+1"` Использование языка ST. Изменение значения локальной переменной.

8.23.11.2 FBD компоненты

Палитра FBD компонентов состоит из:


Вход 458


Выход 459


Блок 459


Переход 461


Возврат 461


Метка 461

Компоненты предназначены для работы в абсолютной схеме компановки 954 рабочей области. Компоненты соединены между собой связями 1273.

Для перечисленных выше компонентов точки прикрепления заданы заранее, их количество зависит от доступного количества входов и выходов у компонента.

Общие свойства компонентов

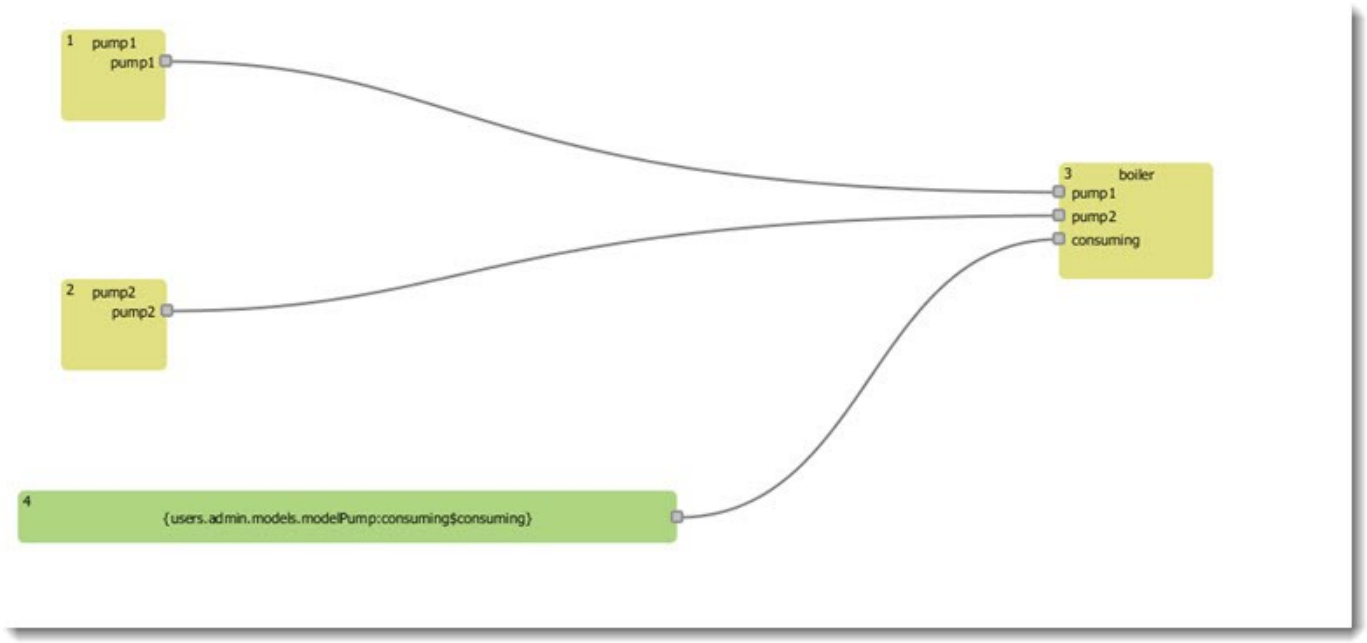
- Шрифт 1275
- Точки прикрепления 1284
- Всплывающее меню 1276
- Порядковый номер 455

Общие механизмы

- При создании компонента автоматически создаются точки для обозначения входов и выходов компонента, и компоненту присваивается следующий порядковый номер.

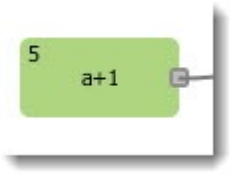
- При перетаскивании связи компонента следуют за ним.
- При изменении или удалении компонента изменяется значение шага компонента в соответствии с номером по порядку. Механизм описан в разделе порядковый номер 455.
- При удалении компонента удаляются связи, прикрепленные к нему.
- При изменении имени функционального блока 459 автоматически расчитываются и перерисовываются его входные и выходные точки.

Пример программы



8.23.11.2.1 Вход

Вход - компонент языка FBD.



Предназначен для вычисления простых переменных, с целью дальнейшей передачи в функцию или в выход 459.

Свойства:

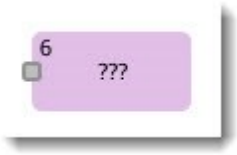
- Стандартные свойства 455 компонента
- Порядковый номер 455
- Выражение 456

Значения по умолчанию

- Компонент по умолчанию содержит одну точку прикрепления 1284. Точка типа " output код 2";
- Значение выражения "???";
- Номер устанавливается следующим по порядку.

8.23.11.2.2 Выход

Выход - компонент языка FBD.



Может быть переменной, для которой будет установлено значение после выполнения предыдущего шага.

Может содержать выражение, которое будет вычислено в момент выполнения текущего шага. Выражение может содержать переменные контекста.

Свойства:

- Стандартные свойства 455 компонента

- Порядковый номер 455

- Выражение 456

Значения по умолчанию

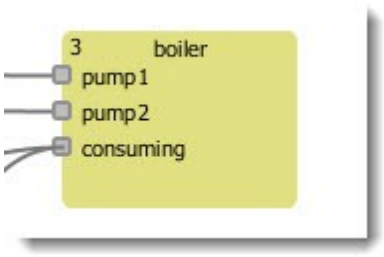
- Компонент по умолчанию содержит одну точку прикрепления 1284. Точка типа " input код 0";

- Значение выражения "???";

- Номер устанавливается следующим по порядку.

8.23.11.2.3 Блок

Блок - компонент языка FBD. С помощью этой команды в схему можно вставлять операторы, функции, функциональные блоки и программы. В функциях и функциональных блоках изображаются входные и выходные параметры.



Количество входов и выходов блока соотвествует количеству входных и выходных переменных. Количество входов и выходов расчитывается каждый раз при присвоении компоненту имени. Если компоненту будет присвоено несуществующее (некорректное) имя, то блок не будет содержать входов/выходов, а после анализа программы информация о ошибке будет выведена в консоль. Над функциональными блоками находится поле, в котором нужно ввести имя экземпляра функционального блока.

Свойства:

- **СТАНДАРТНЫЕ СВОЙСТВА** 455 компонента

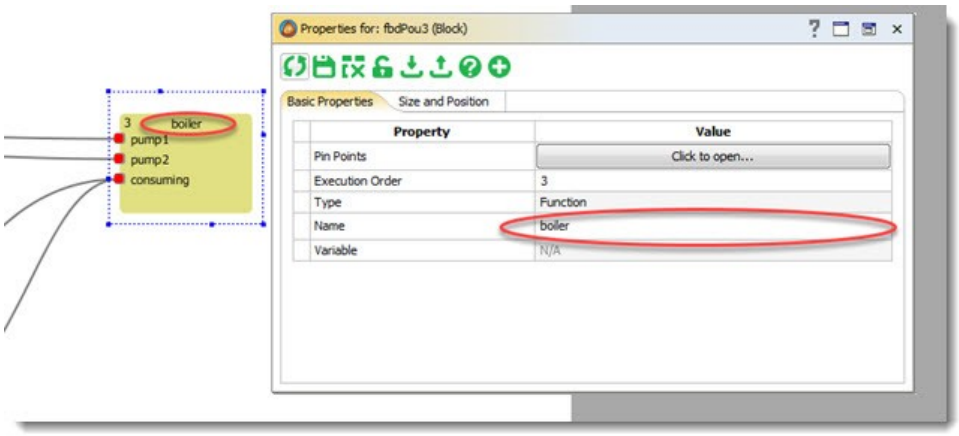
- **ПОРЯДКОВЫЙ НОМЕР** 455

ИМЯ

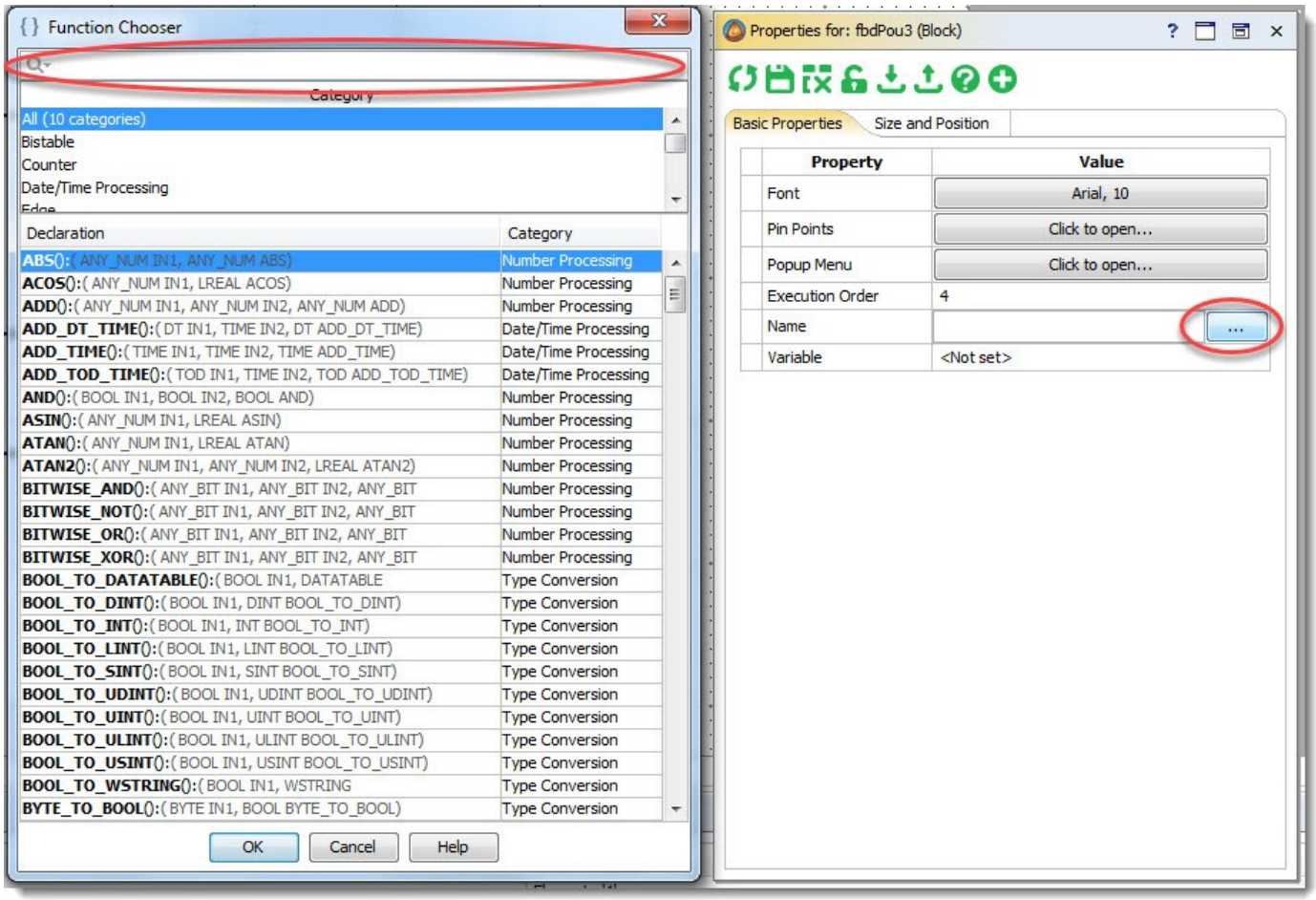
Определяет компонент программы, который будет выполнен в качестве транзакции в этом шаге.

Имя свойства: **name**

Тип свойства: **String**



Выбор выполняемого действия осуществляется при помощи компонента для фильтрации.



После нажатия "..." открывается форма выбора функции. В форме доступны разделы и фильтрация по имени.

Для каждой функции определены: имя, входные параметры, группа.

ПЕРЕМЕННАЯ

Имя локальной переменной, которая хранит экземпляр функционального блока. Переменная должна быть объявлена в области переменных[449] редактируемой программы. Тип переменной должен соответствовать типу функционального блока.

Имя свойства: **variable**

Тип свойства: **String**

8.23.11.2.4 Переход

Переход - компонент языка FBD. Служит для перехода на метку 461.

Инструкция перехода на метку позволяет изменять последовательность выполнения цепей для программирования условий и циклов.

Имя метки для перехода должно быть записано в поле "Выражение". Выполнение программы продолжится с компонента на который был осуществлен переход.

Свойства:

- Все стандартные свойства 455 компонента

- Порядковый номер 455

- Выражение 456

Значения по умолчанию

- Компонент по умолчанию содержит одну точку прикрепления 1284. Точка типа " input код 0" .

- Значение выражения "???"

- Номер устанавливается следующим по порядку

8.23.11.2.5 Возврат

Возврат - компонент языка FBD. Является окончанием программы.

Выход из программного компонента и возврат в вызывающую программу.

Свойства:

- Стандартные свойства 455 компонента

- Порядковый номер 455

- Выражение 456

Значения по умолчанию

- Компонент по умолчанию содержит одну точку прикрепления 1284. Точка типа "input код 0" .

- Значение выражения "???"

- Номер устанавливается следующим по порядку

8.23.11.2.6 Метка

Метка - компонент языка FBD.

Метка перехода - это идентификатор места назначения инструкции перехода. Метка перехода должна существовать для любой инструкции перехода, а для каждой метки должен существовать переход. Имя метки должно быть уникальным.

Важно! Между номером метки и перехода не может быть вставлен другой переход или метка.

Свойства:

- Стандартные свойства 455 компонента
- Порядковый номер 455
- Выражение 456

Значения по умолчанию

- Значение выражения "???"
- Номер устанавливается следующим по порядку

8.23.11.3 SFC компоненты

SFC 1998 - это графический язык, который позволяет описать хронологическую последовательность различных действий в программе. Для этого действия связываются с шагами (этапами), а последовательность работы определяется условиями переходов между шагами.

Палитра SFC 1998 компонентов состоит из:


Шаг 463


Переход 464


Ветвь 465


Прыжок 466

Компоненты предназначены для работы в относительной схеме компоновки 950 рабочей области. Каждый новый компонент добавляется в отдельную ячейку таблицы, позже может быть растянут на несколько ячеек.

Действия 2002 описываются в дополнительной таблице 453.

Общие свойства компонентов

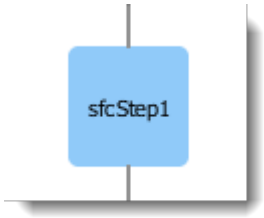
- Шрифт 1275;
- Всплывающее меню 1276.

Общие механизмы

- При создании компонента автоматически создаются точки для прикрепления соседнего или следующего компонента;
- Каждый компонент имеет точки на границах, благодаря которым может быть увеличено кол-во занимаемых ячеек;
- При добавлении в следующую по порядку ячейку компонент связывается с предыдущим, если это возможно.

8.23.11.3.1 Шаг

Шаг - компонент языка SFC 1998.



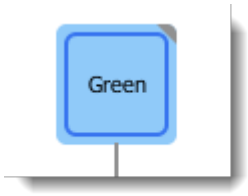
Общие свойства

Все стандартные свойства 455 компонента.

Пользовательские свойства

НАЧАЛЬНЫЙ ШАГ

С этого шага начинается выполнение программы. Когда свойство задано, шаг рисуется двойной рамкой:

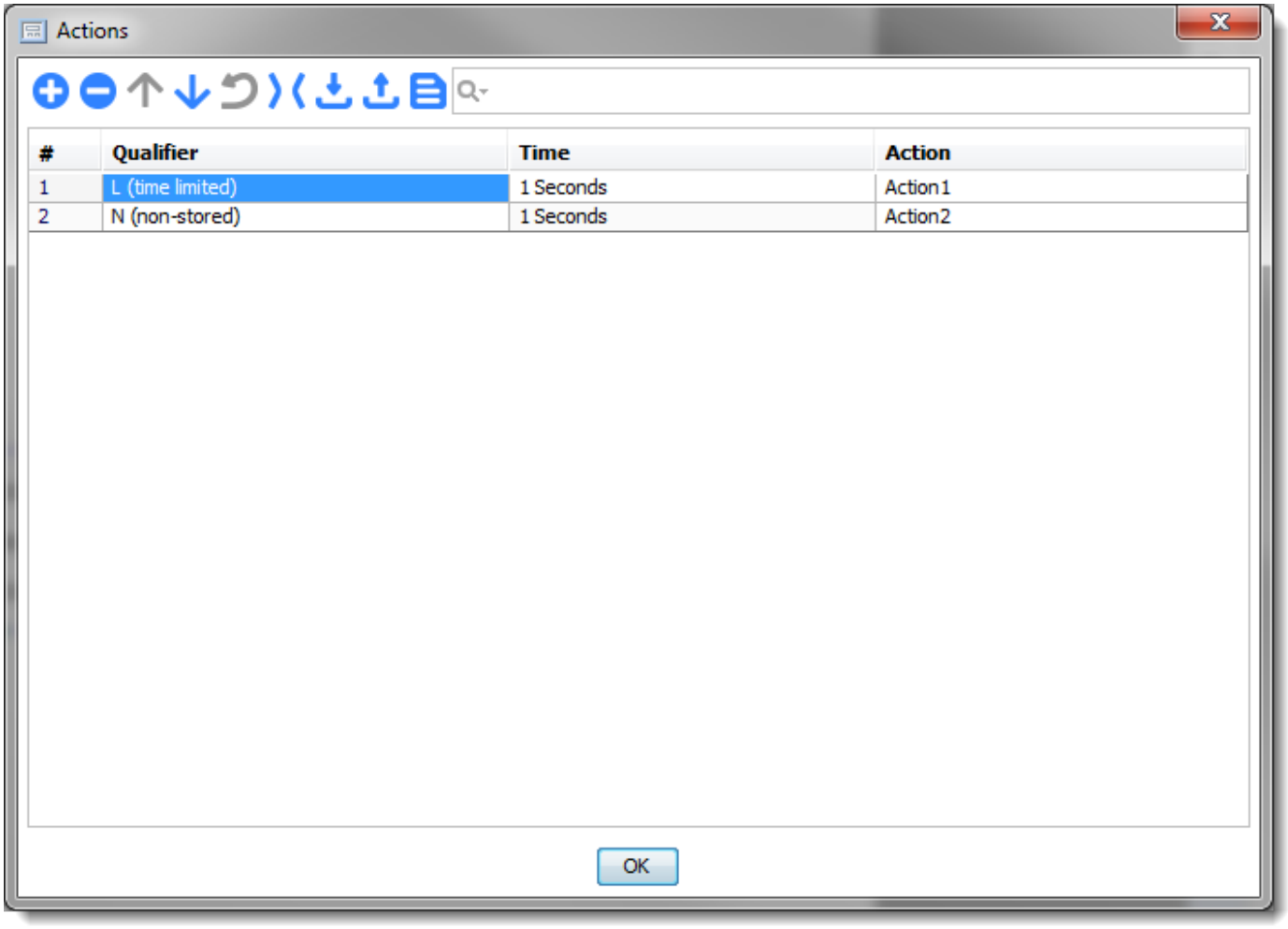


Имя свойства: **initStep**

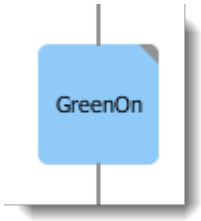
Тип свойства: **Boolean**

ДЕЙСТВИЯ

Свойство содержит список привязанных действий к шагу. Действия 2002 описываются отдельно от них и могут неоднократно использоваться в пределах программы:



Если к шагу привязаны действия, в верхнем правом углу рисуется закрашенный треугольник:



8.23.11.3.2 Переход

Переход - компонент языка SFC 1999.

Общие свойства

Все стандартные свойства 455 компонента.

Пользовательские свойства

ВЫРАЖЕНИЕ

Выражение пишется только на языке ST 1994. По умолчанию в качестве выражения записано "TRUE".

Имя свойства: **expression**

Тип свойства: **String**

Когда выражение записано, то в центре рисуется закрашенный треугольник:

8.23.11.3.3 Ветвь

Ветвь - компонент языка SFC 1998.

Общие свойства

Все стандартные свойства 455 компонента.

Пользовательские свойства

ПАРАЛЛЕЛЬНОСТЬ

Это свойство отвечает за то, что является ли ветвь ли параллельной 2000 или альтернативной 2000. Признаком параллельных ветвей на схеме является двойная горизонтальная линия.

Имя свойства: **parallel**

Тип свойства: **Boolean**

ТИП

Свойство описывает, начальная ли это ветвь или нет.

Имя свойства: **typeBranch**

Тип свойства: **Integer**

Начальная альтернативная ветвь рисуется так:

Конечная альтернативная ветвь рисуется так:

Начальная параллельная ветвь рисуется так:

Конечная параллельная ветвь рисуется так:

8.23.11.3.4 Прыжок

Прыжок - компонент языка SFC 2001.

Общие свойства

Все стандартные свойства 455 компонента.

Пользовательские свойства

ИМЯ ШАГА

Имя шага, на которое должен осуществляться прыжок.

Имя свойства: **jumpStep**

Тип свойства: **String**

8.23.11.4 LD компоненты

(Релейная диаграмма) LD 2005 - графически ориентированный язык программирования, реализующий структуры электрических цепей.

Лучше всего LD подходит для построения логических переключателей, но достаточно легко можно создавать и сложные цепи - как в FBD 2003. Кроме того, LD 2005 достаточно удобен для управления другими программными компонентами.

Палитра LD 2005 компонентов состоит из:

- Силовая линия 467
- Ветление 467
- Блок 468
- Возрат 469
- Переход 469
- Нормально разомкнутый контакт 469
- Нормально замкнутый контакт 470
- Контакт по фронту 471
- Контакт по спаду 471
- Катушка 472
- Инверсная катушка 472
- Катушка по фронту 473

Катушка по спаду 473

Set катушка 474

Reset катушка 474

Компоненты предназначены для работы в относительной схеме компоновки 950 рабочей области. Каждый новый компонент добавляется в отдельную ячейку таблицы и после, может быть растянут на несколько ячеек.

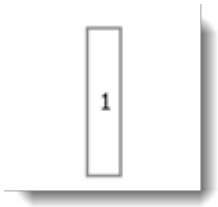
Общие механизмы

- Каждый компонент имеет точки на границах, благодаря которым может быть увеличено кол-во занимаемых ячеек;
- При добавлении в следующую по порядку ячейку компонент связывается с предыдущим, если это возможно.

8.23.11.4.1 Силовая линия

Этот компонент отображает "силовую линию" языка LD 2005.

Компонент "Силовая линия" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274

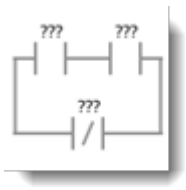
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

8.23.11.4.2 Ветвление

Этот компонент отображает "ветвление" языка LD 2005 .

Компонент "Ветвление" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274

Общие события

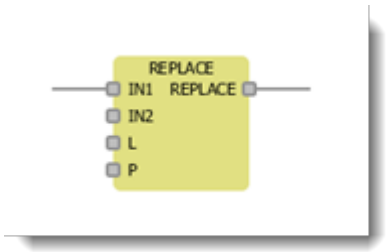
Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика

мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

8.23.11.4.3 Блок

Этот компонент отображает "блок" языка LD 2005.

Компонент "Блок" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274, Точки прикрепления 1284

Пользовательские свойства

ИМЯ

Определяет имя функции, функционального блока или программы.

Имя свойства: **name**

Тип свойства: **String**

ПЕРЕМЕННАЯ

Имя локальной переменной переменной, которая сохраняет экземпляр функционального блока. Переменная должна быть объявлена в области переменных 445 редактируемой программы. Тип переменной должен соответствовать типу функционального блока.

Имя свойства: **variable**

Тип свойства: **String**

ТИП

Тип блока : функция, функциональный блок или программа.

Имя свойства: **type**

Тип свойства: **Integer**

ВЫРАЖЕНИЯ

Выражения передаются в качестве входов или выходов 1986 при вызове блока.

Каждое выражение задается параметрами в таблице:

Поле	Имя	Тип	Описание
Имя	inOutExpressionName	строка	Имя входа или выхода блока
Выражение	inOutExpressionExpression	строка	Выражение которое будет передано в качестве входа или выхода блока

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

8.23.11.4.4 Возврат

Этот компонент отображает "возврат" языка LD 2005.

Компонент "Возврат" выглядит следующим образом:

Общие свойства

Ширина 1274, Высота 1274

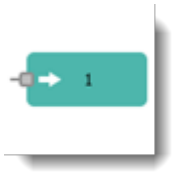
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

8.23.11.4.5 Переход

Этот компонент отображает "переход" языка LD 2005.

Компонент "Переход" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274

Пользовательские свойства

МЕТКА

Номер силовой линии, на которую следует перейти. Переменная по умолчанию - "0".

Имя свойства: **label**

Тип свойства: **Integer**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

8.23.11.4.6 Нормально разомкнутый контакт

Этот компонент отображает "нормально разомкнутый" контакт языка LD 2005.

Компонент "Нормально разомкнутый контакт" выглядит следующим образом:

Общие свойства

Ширина 1274, Высота 1274

Пользовательские свойства

ВЫРАЖЕНИЕ

Выражение пишется только на языке ST 1994. По умолчанию в качестве выражения записано "???".

Имя свойства: **expression**

Тип свойства: **String**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

8.23.11.4.7 Нормально замкнутый контакт

Этот компонент отображает "нормально замкнутый контакт" языка LD 2005.

Компонент "Нормально замкнутый контакт" выглядит следующим образом:

Общие свойства

Ширина 1274, Высота 1274

Пользовательские свойства

ВЫРАЖЕНИЕ

Выражение пишется только на языке ST 1994. "???" необходимо заменить на выражение.

Имя свойства: **expression**

Тип свойства: **String**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

8.23.11.4.8 Контакт по фронту

Этот компонент отображает "контакт по фронту" языка LD 2005.

Компонент "Контакт по фронту" выглядит следующим образом:

Общие свойства

Ширина 1274, Высота 1274

Пользовательские свойства

ВЫРАЖЕНИЕ

Выражение пишется только на языке ST 1994. "???" необходимо заменить на выражение.

Имя свойства: **expression**

Тип свойства: **String**

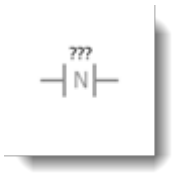
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

8.23.11.4.9 Контакт по спаду

Этот компонент отображает "контакт по спаду" языка LD 2005.

Компонент "Контакт по спаду" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274

Пользовательские свойства

ВЫРАЖЕНИЕ

Выражение пишется только на языке ST 1994. "???" необходимо заменить на выражение.

Имя свойства: **expression**

Тип свойства: **String**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

8.23.11.4.10 Катушка

Этот компонент отображает "катушку" языка LD 2005.

Компонент "Катушка" выглядит следующим образом:

Общие свойства

Ширина 1274, Высота 1274

Пользовательские свойства

ПЕРЕМЕННАЯ

"???" необходимо заменить на имя переменной.

Имя свойства: **variable**

Тип свойства: **String**

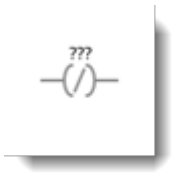
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

8.23.11.4.11 Инверсная катушка

Этот компонент отображает "инверсную катушку" языка LD 2005.

Компонент "Инверсная катушка" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274

Пользовательские свойства

ПЕРЕМЕННАЯ

"???" необходимо заменить на имя переменной.

Имя свойства: **variable**

Тип свойства: **String**

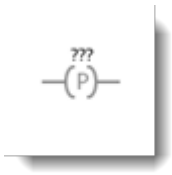
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

8.23.11.4.12 Катушка по фронту

Этот компонент отображает "катушку по фронту" языка LD 2005.

Компонент "Катушка по фронта" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274

Пользовательские свойства

ПЕРЕМЕННАЯ

"???" необходимо заменить на имя переменной.

Имя свойства: **variable**

Тип свойства: **String**

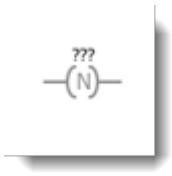
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

8.23.11.4.13 Катушка по спаду

Этот компонент отображает "катушку по спаду" языка LD 2005.

Компонент "Катушка по спаду" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274

Пользовательские свойства

ПЕРЕМЕННАЯ

"???" необходимо заменить на имя переменной.

Имя свойства: **variable**

Тип свойства: **String**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

8.23.11.4.14 SET катушка

Этот компонент отображает "SET катушку" языка LD 2005.

Компонент "SET катушка" выглядит следующим образом:

Общие свойства

Ширина 1274, Высота 1274

Пользовательские свойства

ПЕРЕМЕННАЯ

"???" необходимо заменить на имя переменной.

Имя свойства: **variable**

Тип свойства: **String**

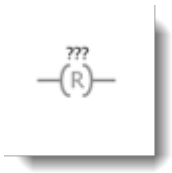
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

8.23.11.4.15 RESET катушка

Этот компонент отображает "RESET катушку" языка LD 2005.

Компонент "RESET катушка" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274

Пользовательские свойства

ПЕРЕМЕННАЯ

"???" необходимо заменить на имя переменной.

Имя свойства: **variable**

Тип свойства: **String**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

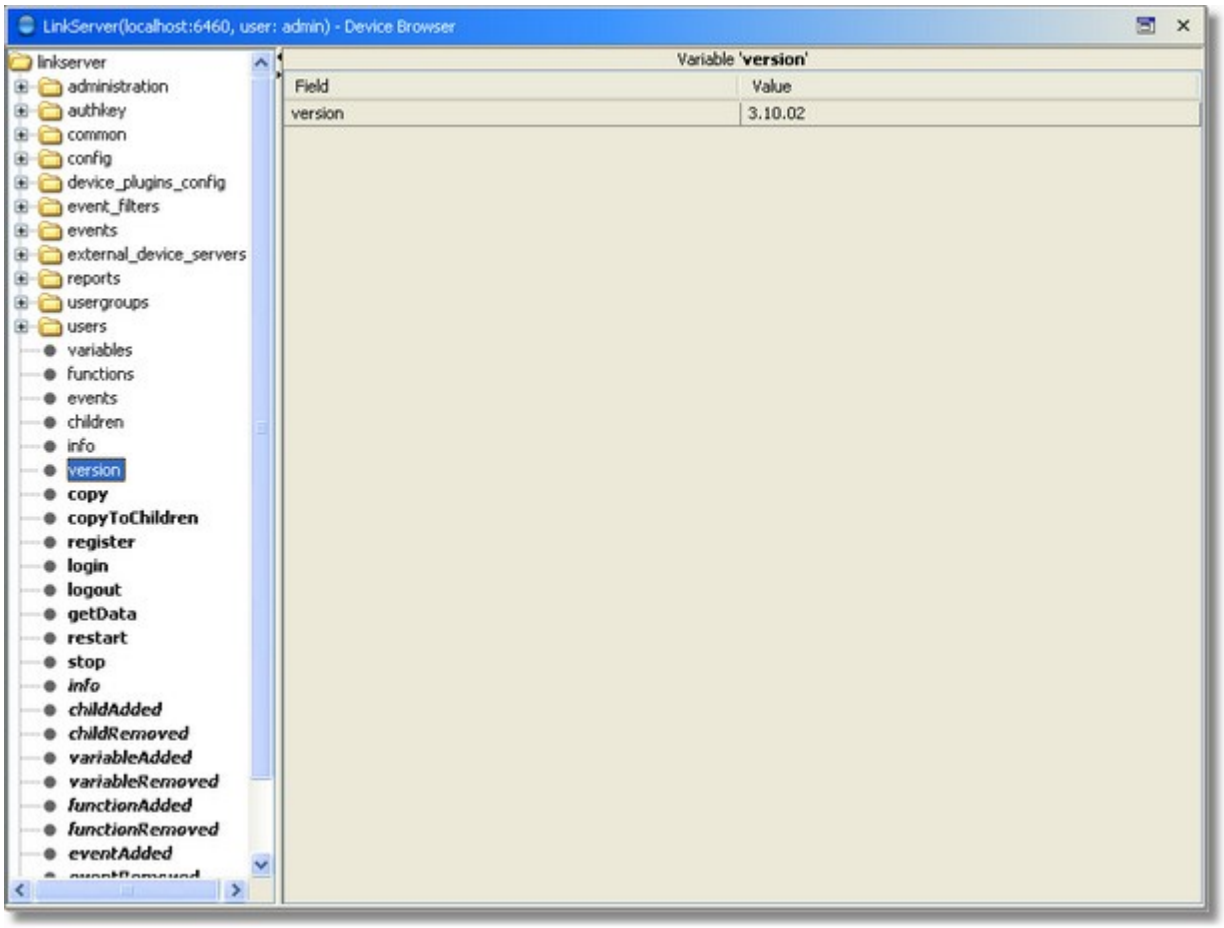
8.24 Браузер устройств

Браузер Устройств используется для навигации по контекстному дереву 41 AtomMind Server, а также для просмотра и изменения значений для переменных 61 контекста, работы с функциями 70 и просмотра событий 73 при помощи компонентов Журнала регистрации событий 398.

Компонент Браузер Устройств предназначен для отладки и решения технических проблем. Его не следует использовать во время обычной работы. Ненадлежащее использование этого компонента может повлечь за собой повреждение базы данных сервера или сбой в работе сервера.

Браузер устройств доступен только тогда, когда AtomMind Client запущен в режиме администратора 362.

Браузер устройств состоит из двух основных частей: контекстного дерева и панели браузера.



Существует четыре типа узлов Контекстного Дерева:

- Узлы-контексты, которые можно развернуть
- Изменяемые узлы, которые отображены нормальным шрифтом
- Узлы-функции, которые выделены **жирным** шрифтом
- Узлы-события, выделенные ***полужирным курсивом***

Развертывание узла-контекста заставляет Браузер Устройств посылать запрос на сервер о дочерних узлах данного узла-контекста.

Щелчок мышью по узлу-контексту отображает информацию о переменных контекста, функциях и событиях.

Двойной щелчок мышью по изменяемому узлу отображает его текущее значение в Редакторе таблиц данных 382.

Двойной щелчок мышью по узлу-функции позволяет ввести параметры функции, используя Редактор Таблиц Данных. В этом случае вызывается функция, и ее вывод отображается в Редакторе Таблиц Данных.

Двойной щелчок мышью по узлу-событию открывает Журнал регистрации событий 398, который начинает отслеживать текущее событие.

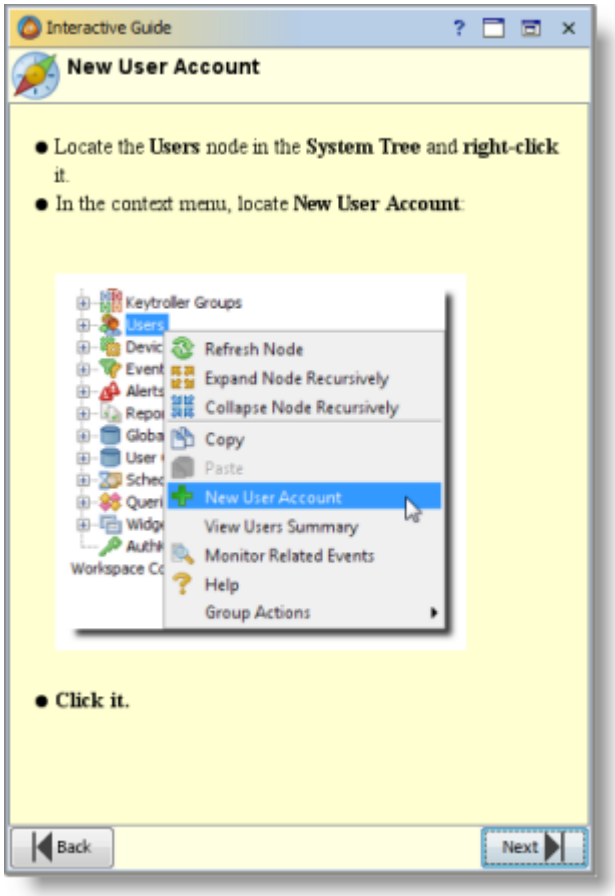
8.25 Интерактивный самоучитель

AtomMind - это легкая в использовании и устойчивая система. Вам следует выполнить несколько простых операций, чтобы заставить её работать. Чтобы сделать использование AtomMind Client максимально простым, мы разработали серию Интерактивных Самоучителей - по одному для каждого из ниже описанных действий. Эти самоучители помогут Вам быстро развернуть и запустить AtomMind.

Доступны следующие самоучители, их должен запускать последовательно любой новый администратор AtomMind Server:

- **Самоучитель 1:** создать новое соединение с сервером
- **Самоучитель 2:** создать учетную запись пользователя
- **Самоучитель 2:** подключиться к устройству
- **Самоучитель 3:** подключиться к серверу устройств

Мы называем этих самоучителей интерактивными потому, что его окно отслеживает Ваши действия с клиентом и подсказывает, что следует делать дальше. Воспринимайте его, как личного инструктора, который стоит рядом за спиной в то время, как Вы осваиваетесь в AtomMind. На протяжении всего времени от Вас требуется лишь внимательно читать и кликать мышкой по нужной кнопке:



ОТОБРАЖЕНИЕ ОКНА САМОУЧИТЕЛЯ

Окно умолчанию отображается автоматически при запуске клиента. Однако, если Вы хотите настроить отображение окна руководства вручную, выберите **Главное меню** > **Помощь** > **Открыть интерактивный самоучитель**.

ТРУДНОСТИ И ОШИБКИ

Мы попытались сделать руководство как можно более простым. Кнопка **Next** (продолжить) включается лишь тогда, когда вы выполнили запрашиваемое действие. Однако, можно вызвать "сбой" в руководстве, если нажать кнопку **Cancel** (отменить) для некоторых диалоговых окон и т.п. Нашей целью было не предупредить все *неправильные* действия, которые можно совершить, а наметить "безопасный путь" при установке программы. Если внимательно следовать руководству, Вы придете к нужному результату.

9 Безопасность и контроль доступа

AtomMind включает в себя очень гибкую и удобно настраиваемую систему безопасности.

Система разработана для обеспечения безопасности созданных на базе платформы приложений на уровне предприятия, а также для создания защищенных публичных веб сервисов с миллионами пользователей.

Регистрация событий безопасности

AtomMind Server позволяет проследить аудиторский след, используя два вида отчетов по всем важным событиям и действиям, включая события, связанные с безопасностью:

- Внутренние события 73, которые могут постоянно храниться в базе данных 692 сервера или маршрутизироваться согласно пользовательским правилам
- Механизм журналирования 166, по умолчанию использующий файловое хранилище

Безопасность постоянного хранилища

AtomMind Server полностью управляет собственным хранилищем данных 692. Ни один сценарий использования системы не подразумевает необходимости прямого доступа к данным, сохраненным в базе данных сервера.

Поскольку любое обращение к базе данных проходит через ядро сервера AtomMind Server, все попытки доступа регулируются внутренним механизмом управления доступом на основе ролей 485.

Все данные, требующие внимания с точки зрения безопасности, такие как данные аутентификации/авторизации для устройств и внешних систем, подлежат шифрованию перед сохранением в БД конфигурации.

Безопасность коммуникаций

Коммуникационный протокол 2119 AtomMind используется для обмена данными между основными компонентами AtomMind: серверами 144, клиентами 359 и агентами 684. Протокол поддерживает SSL/TLS-шифрование, данная опция активна по умолчанию.

Безопасность обмена данными между сервером AtomMind Server и устройствами зависит от возможностей безопасности и шифрования коммуникационного протокола, поддерживаемого устройством. В большинстве случаев, если определенный протокол поддерживает настройки безопасности и шифрования данных, такие же настройки есть и у соответствующего драйвера устройства 518, что обеспечивает безопасный обмен данными.

Внутренний контроль доступа на основе ролей

Все подключения к AtomMind Server через AtomMind Client 359, Web UI 220 или любой API (такие как SOAP 1417 или REST 1401 API) всегда аутентифицируются и авторизуются.

Любая попытка доступа к единой модели данных 41 проверяется согласно правам доступа авторизованного пользователя.

Активные системные объекты (такие, как тревоги или модели) наследуют разрешения от своих владельцев, когда обращаются к единой модели данных.

Для большей информации см. раздел Управление доступом на основе ролей 485.

Безопасность экземпляра сервера

AtomMind Server - это самостоятельное Java-приложение, разворачиваемое специальным инсталлятором. Сервер не использует специфичных для операционной системы методов (таких как Windows Registry или Linux IPC) для обмена данными с другими процессами и приложениями, запущенными на этом же хосте. Весь обмен данными осуществляется через файлы и IP-соединения через локальный хост.

Инсталлятор следит за настройками прав доступа к файлам внутри установочной папки AtomMind Server, и ручная настройка прав доступа к файлам не требуется.

9.1 Пользователи

AtomMind Server построен как *многопользовательский сервер*. Как правило, одни люди устанавливают и контролируют сам AtomMind Server, другие - разрабатывают приложения в режиме low-code, третьи разворачивают и тестируют эти приложения, а конечные пользователи ежедневно работают с ними. Все эти администраторы/разработчики и конечные пользователи (операторы системы) могут работать в одной компании, но бывает, что конечный пользователь - это покупатель продукта или сервиса на базе AtomMind. В таком сложном окружении очень важно обеспечить общую безопасность и ограничить доступ к важным данным.

Пользователи получают доступ к серверу через *учетные записи пользователей.* Сразу после установки AtomMind Server в системе создается как минимум одна учетная запись, администратор по умолчанию 479.

Для того чтобы получить доступ к AtomMind Server, используя один из пользовательских интерфейсов (Web UI 220, AtomMind Client 359 и т.д.), необходимо провести аутентификацию и авторизацию на сервере. Единственной операцией, не требующей предварительной аутентификации, является саморегистрация 484.

AtomMind Server может иметь неограниченное количество учетных записей пользователя. Обычно ресурсами системы владеет пользователь, создавший их. Права доступа 477 для пользователя настраиваются путем редактирования таблицы прав доступа 487, в которой определяется уровень прав доступа для пользователя к каждому ресурсу системы. Это дает возможность администратору внедрять сложные схемы обеспечения безопасности, фактически отражающие роль пользователя в организации. Например:

- Разрешить доступ одним пользователям к просмотру/модификации устройств и ресурсов (тревог, отчетов...), принадлежащих другому пользователя, т.е. совместное использование ресурса.
- Разрешить одному пользователю (руководителю группы) менять права доступа некоторых других пользователей (членов группы).
- Ограничить доступ пользователя к его собственным устройствам и ресурсам, например, включить просмотр устройства и отчетов в режиме "только чтение".
- Временно деактивировать учетную запись пользователя, отозвав все права доступа.
- Каждая запись в таблице прав доступа может определять уровень прав доступа пользователя для одного ресурса, группы ресурсов или даже под-дерева зависимых ресурсов.

Редактируемые права доступа пользователя также облегчают жизнь системному администратору, позволяя ему просматривать только относящиеся к его работе ресурсы. Например, следующая схема предоставления прав доступа часто используется для систем учета рабочего времени и контроля доступа:

- Системный администратор имеет полный права доступа.
- Руководство компании имеет доступ к отчетам.
- Служба HR может просматривать/редактировать профили сотрудников и настраивать смены.
- Служба безопасности может просматривать в реальном времени события входа/выхода, а также историю этих событий.
- IT-инженеры могут редактировать шаблоны отчетов, создавать новые отчеты, просматривать историю событий и вносить изменения в базу данных сотрудников.
- У каждой учетной записи пользователя имеется набор предпочтений, таких как часовой пояс, формат даты/времени и предпочтительный язык.

Ссылка по теме: Предоставление доступа одному пользователю к объектам другого пользователя 1640.

Обычные и ролевые пользователи

Учетная запись пользователя может соответствовать либо одному физическому лицу (например, John Doe или Mary Shelley), либо определенной роли системных операторов (например, "Оператор зоны Лос-Анджелес", "Дизайнер отчетов" или "Сетевой инженер"). Учетные записи физических лиц могут либо иметь собственные таблицы прав доступа, либо наследовать права доступа от учетных записей на основе ролей.

Более подробно см. в разделе Обычные и ролевые аккаунты пользователей 480.

Внешняя аутентификация пользователей

С большими инсталляциями AtomMind работают тысячи людей, и каждый из них наследует одну или несколько ролей. Создание и поддержание индивидуальных учетных записей для такого количества пользователей слишком трудозатратно. В то же время, есть возможность аутентификации пользователей через внешние системы (такие как LDAP или Microsoft Active Directory), а авторизация (присвоение прав пользователя) происходит с использованием ролевых учетных записей пользователя AtomMind Server.

Более подробно см. в разделе Внешняя аутентификация 490.

Саморегистрация пользователей

Саморегистрация пользователя очень полезна на первых стадиях развертывания системы, либо при предоставлении публичного облачного сервиса. Пользователи системы могут создавать свои собственные аккаунты и предоставлять некоторые личные данные (имя, e-mail, компания/отдел, номер телефона и т.д.). После прохождения регистрации они получают свои собственные логин и пароль.

Более подробно см. в разделе Саморегистрация пользователей 484.

Владение

Каждый пользовательский аккаунт *владеет* различными системными объектами: Тревогами 779, Виджетами 943 и т.д. Таблица прав доступа пользователя может не разрешать ему доступ к контекстам 41 объектов других пользователей или даже к контекстам некоторых собственных объектов. Объекты, к которым есть доступ у вновь созданных пользователей, определяются глобальной настройкой AtomMind Server Права доступа по умолчанию 202.

Администрирование пользователей

Для администрирования пользователей применяются два контекста. Один из них - это общий контекст Пользователи 1604 для действий, относящихся ко всем учетным записям пользователей. Другой - контекст Пользователь 1604, относящийся к отдельной учетной записи.

9.1.1 Управление учетными записями пользователей

Существуют два способа создания новых учетных записей пользователей:

- Создание учетных записей пользователями, обладающими правами администратора
- Саморегистрация

В большинстве случаев следует использовать первый способ. Системные администраторы могут создавать учетные записи пользователей, используя опцию Новая учетная запись 1607 в контексте Пользователи. Единственные параметры, которые необходимо уточнить при создании учетной записи, это имя пользователя и пароль. Все остальные параметры выставлены по умолчанию. Способ создания таблицы доступов по умолчанию для новой учетной записи пользователя описан 488 в разделе Безопасность и Доступ.

Кто-то из ваших пользователей забыл свой пароль? Смотрите раздел Изменение пароля учетной записи пользователя 484!

Удаление учетных записей пользователей

Удаление учетных записей пользователей происходит при помощи действия Удалить 1608 контекста Пользователь 1608.

Удаление учетной записи приводит к полному удалению всех ресурсов, принадлежащих пользователю, таких как аккаунты устройств 497, тревоги 779, модели 810 и т.д. Удаленная учетная запись не подлежит восстановлению.

Пользователь не может самостоятельно удалить собственную учетную запись.

9.1.2 Учетная запись администратора по умолчанию

Учетная запись администратора по умолчанию является учетной записью, создаваемой при каждой новой установке AtomMind Server:

- Имя пользователя: `admin`
- Пароль: `admin`

Советуем изменить пароль как можно быстрее.

Администратор по умолчанию имеет полный доступ. Говоря на техническом языке, его таблица доступа 487 содержит единственную запись, наделяющую его доступом уровня 486 **Администратора** ко всем контекстам 41 AtomMind Server:

Маска контекста	**Доступ**
*	Администратор

Учетную запись администратора по умолчанию невозможно удалить, также невозможно вносить изменения в таблицу доступа.

9.1.3 Учетная запись оператора по умолчанию

Учетная запись оператора по умолчанию является учетной записью пользователя, создаваемой при большинстве установок AtomMind Server :

- Имя пользователя: `operator`
- Пароль: `operator`

Советуем изменить пароль кам можно быстрее.

Оператор по умолчанию имеет:

- Доступ к собственным ресурсам уровня 486 *Оператора*
- Доступ к ресурсам администратора по умолчанию уровня 486 *Наблюдателя*

9.1.4 Обычные и ролевые аккаунты пользователей

Учетная запись пользователя в AtomMind Server может соответствовать одному физическому лицу (например, Джон До или Мэри Шелли) или определенной роли системного оператора (например, "оператор в Лос-Анджелесе", "дизайнер отчетов" или "сетевой инженер"). Учетные записи пользователей, которые соответствуют отдельным лицам, называются *обычными* учетными записями, а соответствующие ролям - *ролевыми*.

Тип учетной записи пользователя контролируется следующим образом:

- Если учетная запись пользователя имеет выключенный флажок **Наследовать права доступа** в свойствах учетной записи 482, учетная запись использует собственную таблицу Прав доступа 483. Учетная запись может логически соотнести живого человека с его собственными правами доступа или логически соотнести роль пользователя, которая будет наследоваться другими пользователями (обычно соответствующими живым людям).
- Если учетная запись пользователя имеет включенный флажок **Наследовать права доступа**, учетная запись использует таблицу прав доступа других пользователей. Эта учетная запись будет, скорее всего, соответствовать живому человеку, наследующему системную роль их ролевой учетной записи.
- Если пользователь аутентифицируется из Active Directory (или другого сервера LDAP), он будет авторизовываться путем использования определенной ролевой учетной записи пользователя. Таким образом, права доступа пользователя будут наследоваться из этой учетной записи. См. подробности в Аутентификации Active Directory (LDAP) 491.

9.1.5 Конфигурация учетной записи пользователя

В данном разделе содержится информация о свойствах конфигурации учетной записи пользователя.

Все свойства, описанные здесь, доступны через действие Конфигурировать 105 в Контексте пользователя 1608.

9.1.5.1 Свойства

Пользовательские *Свойства* определяют базовые опции учетной записи.

Имя пользователя	Имя учетной записи, а также имя контекста Пользователя 1608. Как только учетная запись создана, данное свойство становится доступным только для чтения и не может быть изменено.
Имя	Имя владельца учетной записи.
Фамилия	Фамилия владельца учетной записи.
Пароль	Пароль учетной записи, используемый пользователем для входа. Пароль должен соответствовать Настройкам паролей 201.
Инициализационный пароль	Определяет, является ли пользовательский пароль паролем для инициализации. Если выставлено в TRUE, пользователю будет предложено заменить пароль при первом входе в систему. Это свойство автоматически выставляется в FALSE при изменении пароля.
Игнорировать настройки режима подключения	Заставляет систему игнорировать Режим пользовательского подключения 202 для данного пользователя.
Использовать внешнюю аутентификацию	Определяет поведение аутентификации 202 пользователя. Если опция включена, аутентификация пользователя происходит через плагин внешней аутентификации 490, в противном случае - через AtomMind Server.
Страна	Страна пользователя.
Регион/Штат/Провинция/Область	Регион пользователя.
Почтовый индекс	Почтовый индекс пользователя.
Город	Город пользователя.
Адрес 1	Адрес пользователя.
Адрес 2	Вторая строка для ввода адреса пользователя.
Комментарии	Дополнительная информация об учетной записи.
Компания	Компания пользователя.
Отдел	Отдел пользователя.
E-mail адрес	E-mail адрес пользователя.
Телефон	Номер телефона пользователя.
Факс	Номер факса пользователя.
Временная зона	Временная зона, отображающая корректное время для данного пользователя. Также Device, зарегистрированные под этой учетной записью, будут расположены в данной временной зоне при отсутствии временной зоны, определенной в Учетной записи Device. Для более подробной информации см. раздел Временные зоны 911.
Место нахождения	Настройки места нахождения пользователя определяются кодом страны, состоящим из двух букв, который влияет на различные параметры интерфейса пользователя.
Формат даты	Шаблон даты 2146 определяет формат даты, представляемой пользователю.
Формат времени	Шаблон времени 2146 определяет формат времени, представляемого пользователю.
Стартовый день недели	Контролирует первый день недели, отображенный в календаре. Возможные варианты - это воскресенье и понедельник.

Все эти свойства доступны через переменную childInfo 1609.

9.1.5.2 Настройки учетной записи

Настройки учетной записи пользователя определяют различные параметры учетной записи.

Выражение ограничений соединения	Выражение вычисляется при входе пользователя в систему. Если возвращает FALSE, попытка входа будет неудачной, и появится сообщение об ошибке.
Учетная запись активирована	Вход пользователя не разрешен, если учетная запись не активирована.
Время активации	Если установлено время активации, вход пользователя не разрешен до заданного времени.
Время окончания действия	Если установлено время окончания действия, вход пользователя запрещен по истечении данного времени.
Унаследовать права доступа	Определяет, имеет ли пользователь собственную таблицу прав доступа, или наследует права доступа от другого (на основе ролей) пользователя. Если этот флаг включен, таблица прав доступа 483 пользователя не используется и недоступна для редактирования. Более подробно см. в разделе Обычные и ролевые аккаунты пользователей 480.
Унаследовать права доступа от	Указывает на пользователя на основе ролей, который предоставит права доступа текущему пользователю. Поле доступно только если активирована опция **Унаследовать права доступа.**
Использовать разрешения на основе ролей	Включает использование прав доступа на основе ролей.

Данные параметры для учетной записи администратора по умолчанию 479 доступны только для чтения.

Все эти свойства доступны через переменную настройки 1611.

9.1.5.3 Ресурсы

Таблица *Ресурсов* определяет, какого типа ресурсы (такие как тревоги, виджеты или отчеты) имеет пользователь. Если определенный тип ресурса не активирован в этой таблице, у учетной записи пользователя не будет контекста 41 контейнера для ресурсов этого типа, т.е. создавать и управлять такими ресурсами будет невозможно.

Если у пользователя уже были ресурсы определенного типа, и ресурсы этого типа были деактивированы в таблице Ресурсов, существующие ресурсы удаляться не будут. Однако эти ресурсы будут недоступны до тех пор, пока тип ресурса не будет активирован повторно.

Деактивирование ресурсов отличается от деактивирования доступа 483 к этим ресурсам. Если доступ к определенному ресурсу или контейнеру ресурсов не открыт в таблице прав доступа пользователя, другие пользователи все равно смогут получить доступ к этим ресурсам и управлять ими. Однако если определенный тип ресурсов деактивирован в таблице ресурсов, контейнер и его дочерние ресурсы будут недоступны для всех пользователей AtomMind Server.

Доступность по умолчанию для всех типов ресурсов глобально настраивается через таблицу Права доступа пользователя по умолчанию 202.

К таблице ресурсов можно получить доступ через переменную ресурсы 1611.

9.1.5.4 Фотография пользователя

Свойство *Фотография пользователя* используется для хранения личной фотографии пользователя, размером до 200 Кб.

На фотографию могут ссылаться различные дополнительные средства, такие как отчеты входа/выхода. Доступ к фотографии возможен через переменную фотография 1612 пользователя.

9.1.5.5 Интерфейс пользователя

Свойство *Интерфейс пользователя* определяет специфические настройки пользовательского интерфейса.

Эти настройки используются, только если приложение AtomMind Client 359 запущено в браузере или в режиме 359 Java Web Start. Если же оно запущено в режиме Unified Operations Console, AtomMind Client поддерживает несколько серверных соединений и, следовательно, большинство настроек пользовательского интерфейса настраиваются из его командной строки 361.

Простой режим	Запускает работу Web UI в простом режиме 220 и работу AtomMind Client в простом режиме 362. Эта настройка имеет решающее значение для построения операторских интерфейсов или решений, продуктов и сервисов на базе платформы. Если она отключена для пользователя, он не увидит стандартные административные компоненты (Системное дерево 370, Журнал Событий 398 и т.д.) при входе в систему. Будут отображены только компоненты, настроенные в Действиях автозапуска 941 (например, панели инструментов 912), предоставляя полностью настраиваемый операторский интерфейс.
Задержка подсказок	Контролирует задержки подсказок, отображаемых всеми компонентами интерфейса. Установите в этой настройке большое значение, чтобы эффективно деактивировать подсказки. Данная настройка не работает: • В Web UI 220 • В AtomMind Client 359 при работе в режиме Единая консоль 359.
Упрощенные сообщения об ошибках	Если эта настройка включена, пользователь будет видеть укороченные сообщения об ошибках при возникновении проблем на любом уровне платформы, с которыми он может столкнуться при использовании продуктов или сервисов на базе AtomMind. В частности, такие сообщения не будут содержать трассировку стека Java.

9.1.5.6 Права доступа

Представленное в виде таблицы свойство *Права доступа* содержит таблицу прав доступа пользователя. Всю информацию относительно прав доступа пользователя можно найти в разделе Безопасность и контроль доступа 477 .

Маска контекста	Маска 44 тех контекстов, к которым будет применена данная запись
Тип объекта	Тип объектов контекста, для которых будет использоваться выбранный уровень прав доступа: **Все**, **Переменная**, **Функция**, **Событие**, **Действие**, **Группа переменных**, **Группа функций**, **Группа событий** или **Группа действий**.
Объект	Имя определенного объекта контекста, для которого будет использоваться выбранный уровень прав доступа. В зависимости от **Типа объекта,** может быть именем переменной, группы функций и т.д.
Права доступа	Уровень 486 прав доступа для данной записи.

Пользователь не может менять свои права доступа.

Права доступа пользователя доступны для просмотра через переменную permissions 1610.

9.1.5.7 Тревоги

Свойство *Тревоги* определяет, какие всплывающие уведомления-тревоги 791 будут видны этому пользователю:

- Уведомления о тревогах, которыми владеет пользователь
- Уведомления о всех тревогах, к которым имеет доступ 477 пользователь

9.1.6 Статус учетной записи пользователя

Информацию о статусе учетной записи можно посмотреть, используя действие Посмотреть статус 111 в Контексте пользователя 1608. Она содержит следующие разделы:

Время создания учетной записи	Предназначено только для чтения, показывает время создания учетной записи.
Время последнего обновления учетной записи	Предназначено только для чтения, показывает время последнего обновления основных свойств 481 учетной записи.

9.1.7 Изменение пароля учетной записи пользователя

Иногда случается, что пользователь забывает пароль. В таком случае пользователь должен предупредить системного администратора, который установит новый пароль.

В большинстве случаев системный администратор может просто открыть настройки учетной записи пользователя в пользовательском интерфейсе и изменить значение в поле **Пароль**.

В других случаях, например, когда потерян пароль самого системного администратора, смену пароля можно осуществить через командную строку.

Изменение пароля учетной записи пользователя через командную строку

Изменение пароля осуществляется при помощи опции командной строки 164 `-p`. Синтаксис данной опции следующий `-p <username> <password>` .

Итак, предположим, что пользователь *johndoe* забыл свой пароль и сообщает об этом администратору, который для смены пароля *johndoe* на *foobar* должен сделать следующее:

- Остановить AtomMind Server (см. Выключение сервера 161).
- Запустить AtomMind Server снова с параметром командной строки `-p johndoe foobar`.
- Теперь пароль `johndoe` изменен на `foobar`, и вход может быть осуществлен с новым паролем.

9.1.8 Временное отключение учетных записей

Для временного отключения учетной записи пользователя измените Настройки учетной записи 482 и снимите флажок Учетная запись активирована.

Чтобы отключить доступ пользователя к определенным ресурсам или операциям, измените Таблицу прав доступа пользователя 487 и установите **Нулевой** уровень доступа для необходимых контекстов.

9.1.9 Саморегистрация пользователей

Саморегистрация пользователя позволяет операторам AtomMind создавать собственные учетные записи, используя любой из Пользовательских Интерфейсов AtomMind Server. Данный метод доступен только при включенной функции Включить саморегистрацию пользователя 180 в глобальных настройках.

После включения саморегистрации пользователя, ее поведение зависит от разных пользовательских интерфейсов (UI) AtomMind Server:

- В Web UI 220 саморегистрация осуществляется через специальную страницу **Register,** на которую идет отсылка со страницы входа в систему. Видимость и обязательность полей саморегистрации можно настроить 204.
- В AtomMind Client 359, если пользователь пытается войти в систему под несуществующим именем, ему предложат зарегистрировать новый аккаунт. Если пользователь подтверждает регистрацию, будет создана новая учетная запись AtomMind Server с авторизационными данными, указанными пользователем. Более подробно см. здесь 376 .

9.1.10 Пользователи в распределенной архитектуре

Этот раздел описывает особенности поведения пользователей в распределенной инсталляции[1332] AtomMind.

Если учетная запись пользователя из сервера-поставщика подключена к дереву контекстов на сервере-потребителе, сама учетная запись пользователя и все ресурсы, которыми владеет пользователь, становятся доступны для пользователей сервера-потребителя. Однако пользователи сервера-поставщика не могут подключаться к серверу-потребителю, используя их реквизиты доступа к системе.

Когда пользователи сервера-потребителя получают доступ к ресурсам подсоединенной пользовательской учетной записи сервера-поставщика, их эффективный уровень прав доступа[486] к определенному ресурсу сервера-поставщика определяется следующим:

- Уровень прав доступа сервера-поставщика для пользователя сервера-потребителя, который вошел в систему
- Уровень прав доступа сервера-потребителя для пользователя, который выполняет операцию

См. права доступа в распределенной среде[1338] для получения более подробной информации.

9.2 Управление доступом на основе ролей

Каждый AtomMind Server проходит *аутентификацию* и *авторизацию* во время процедуры входа. После завершения авторизации, сеанс пользователя связывается с определенной таблицей прав доступа[487]. Каждая попытка этого пользователя получить доступ к любой части единой модели данных[41] проверяется на обоснованность. Доступ может быть предоставлен или запрещен в результате этой проверки.

Активные системные объекты, т.е. объекты со своим отдельным жизненным циклом и взаимодействующие с другими объектами сервера без прямого участия оператора, всегда принадлежат какому-либо пользователю[478] системы. Таким образом, подобный активный объект выполняет все действия с использованием прав доступа своего владельца.

Пример: Каждая запланированная задача[823] наследует во время выполнения права доступа своего владельца.

Подключения к AtomMind Server через любой API (такой как SOAP[1417] или REST[1401] API) аутентифицируются и авторизуются как обычное подключение любого оператора.

УРОВНИ ПРАВ ДОСТУПА

Большинство системных ресурсов связаны с соответствующим уровнем прав доступа[486]. *Уровень прав доступа* описывает, кто может получить доступ к ресурсу.

Ресурсами с соответствующими уровнями прав доступа являются:

- Контексты[41]
- Переменные[61]
- Функции[70]
- События[73]
- Действия[87]

ТАБЛИЦА ПРАВ ДОСТУПА

Каждая учетная запись пользователя[478] имеет таблицу прав доступа[487], которая устанавливает *уровни прав доступа* для групп контекстов[41], определяемых контекстными масками[44], а также для отдельных сущностей контекстов (переменных, функций, событий и действий).

ПРОЦЕСС ПРОВЕРКИ ПРАВ ДОСТУПА

Для подробной информации о том, как сервер определяет, предоставлять ли доступ к определенному объекту системы, см. раздел Проверка прав доступа[489].

9.2.1 Уровни прав доступа

Уровень прав доступа определяет разрешение доступа к какому-либо ресурсу. Если уровень прав пользователя, запрашивающего доступ, *включает* уровень прав доступа к требуемому ресурсу, доступ разрешается. В противном случае, доступ будет запрещен.

AtomMind Server взаимодействует со следующими уровнями прав доступа:

Уровень прав доступа	**Включает**	**Комментарии**
Нет прав	Нет прав	Нет заданных прав доступа. Данный уровень относится к каждому пользователю, который пытается получить доступ к системе без предварительной авторизации. С таким уровнем прав доступа могут быть выполнены немногочисленные операции, такие как авторизация, регистрация новой учетной записи пользователя (если включена опция глобальной конфигурации 179 **Включить саморегистрацию пользователей**) и т.д.
Наблюдатель	Нет прав, Наблюдатель	Минимальный уровень прав доступа. Этот уровень позволяет просматривать большинство информации без разрешения на ее редактирование. Может оказаться полезным предоставление "демо"-прав доступа для просмотра определенных объектов. Наблюдатели могут видеть настройки и события устройств, но не могут их конфигурировать и выполнять операции, предоставляемые аппаратными устройствами.
Оператор	Нет прав, Наблюдатель, Оператор	Уровень прав доступа большинства обычных пользователей. Позволяет просматривать несистемные данные и выполнять некоторые базовые операции по изменению данных. Операторы могут конфигурировать аппаратные устройства и выполнять их операции, но не могут создавать/удалять или редактировать настройки подключения и обработки данных.
Менеджер	Нет прав, Наблюдатель, Оператор, Менеджер	Данный уровень позволяет просматривать все данные и вносить некоторые изменения, например, контролировать аппаратные устройства, изменять шаблоны отчетов, создавать и контролировать тревоги и т.д.
Инженер	Нет прав, Наблюдатель, Оператор, Менеджер, Инженер	Уровень, разрешающий операции с несистемными данными, включая потенциально опасные, таких как запуск скриптов. Данный уровень должен относиться к системным инженерам.
Администратор	Нет прав, Наблюдатель, Оператор, Менеджер, Инженер, Администратор	Данный уровень прав доступа позволяет выполнение административных действий: вносить изменения в глобальную конфигурацию AtomMind Server, останавливать/перезапускать сервер, управлять учетными записями пользователей и т.д. Если пользователь имеет права доступа Администратора к контексту, он может делать в нем, что угодно.

Если вы хотите проверить, какой требуется уровень прав доступа к определенному контексту, переменной, функции, событию или действию, пожалуйста, посмотрите раздел Контекстная ссылка 1450.

Может показать нелогичным тот факт, что события имеют уровень прав доступа. Событие просто происходит, хотите вы этого или нет. Устройство отключается от системы - это событие. Однако, вы можете выбрать, кто может просматривать или контролировать его. В этом случае применяются уровень прав доступа. Они позволяют вам определить, кто может просматривать события в Журнале событий, создавать Тревоги 779 для данного события и т.д.

Дополнительная информация: Чтобы понять, что мы имеем в виду под фразой "Право доступа *включает* другое право доступа", вам нужно рассматривать права доступа как двоичные битовые *битовые маски*. Если вы не знаете наверняка, что такое битовая маска, пожалуйста, обратитесь к статье http://en.wikipedia.org/wiki/Bitmask.

Уровень прав доступа внутренне выражается как битовая маска. Например, внутренняя битовая маска для уровня Не определен является 00000000, для уровня Наблюдатель - 00000001, для уровня Администратор - 00011111. Итак, как видите, маска уровня прав доступа Администратор (это маска, а не просто число) *включает* маски уровней Наблюдатель и Не определен, потому что имеет **1** в каждой позиции, где другие маски имеют **1**.

Данный метод используется, потому что он более универсален и эффективен, чем просто число (т.е. Не определен - 0, Наблюдатель - 1 и т.д.)

Пример 1

Если уровень прав доступа пользователя, запрашивающего доступ к определенному ресурсу, является **Менеджер**, а требуемый уровень прав - **Администратор**, доступ будет запрещен, потому что **Менеджер** *не включает* в себя уровень **Администратор**.

Пример 2

Если уровень прав доступа пользователя, запрашивающего доступ к определенному ресурсу, является **Менеджер**, и требуемый уровень прав доступа тоже **Менеджер**, доступ будет разрешен, т.к. **Менеджер** включает в себя уровень **Менеджер**.

9.2.2 Уровень прав доступа к контексту по умолчанию

Каждый контекст имеет *уровень прав доступа по умолчанию*. Чтобы выполнить некоторые операции в контексте, пользователь должен иметь соответствующий или более высокий уровень прав доступа. Особые переменные, функции, события или действия могут требовать другие уровни прав доступа.

Уровень прав доступа по умолчанию, определенный практически для всех контекстов в AtomMind Server, является **Наблюдатель**. Далее приведен список исключений (контекстов, уровень прав доступа по умолчанию которых отличается от **Наблюдателя**):

Контекст	**Уровень прав доступа по умолчанию**
Администрирование 1450	Администратор
Корневой 1559	Не определен
Пользователи 1604	Администратор

Пример 1

Уровень прав доступа по умолчанию Корневого контекста 1559 AtomMind Server является **Не определен**. Это значит, что даже не аутентифицированные пользователи могут выполнять некоторые действия с данным контекстом. Однако, действия **Перезапустить сервер** и **Остановить сервер**, определенные в данном контексте, требуют уровень прав доступа **Администратор** для данного контекста. Таким образом, только пользователи с уровнем прав доступа Администратор для корневого контекста смогут остановить или перезапустить сервер.

Пример 2

Уровень прав доступа по умолчанию контекста Пользователь 1608 является **Наблюдатель**. Однако, действие *Удалить* для данного контекста доступно на уровне **Администратор**. Таким образом, пользователь с уровнем прав доступа **Наблюдатель** сможет просматривать статус учетной записи, но не сможет удалить ее. Для этого требуется уровень прав доступа **Администратор**.

9.2.3 Таблица прав доступа

Каждая учетная запись пользователя имеет свойство таблица прав доступа 483. Данная таблица используется для определения необходимого уровня прав доступа для контекста. Это уровень 486, при котором пользователь пытается получить доступ к данному контексту. Если уровень прав доступа пользователя включает уровень прав доступа к ресурсу, доступ будет разрешен.

Таблица прав доступа имеет четыре столбца:

- **Маска контекста**
- **Тип объекта**
- **Объект**
- **Права доступа**

В большинстве случаев, **Тип объекта** и **Объект** установлены на **Any**. Такая запись в таблице прав доступа включает определенный уровень доступа для всего контекста. В то же время, указанные **Тип объекта** и **Объект** позволяют давать права определенным **Переменным**, **Функциям**, **Событиям**, **Действиям** и их **группам**.

Когда пользователь пытается получить доступ к ресурсу, AtomMind Server использует таблицу прав доступа для определения, разрешать ли доступ. Для более подробной информации см. Проверка прав доступа 489.

Маска контекста и уровень прав доступа похожи на понятия "*Домен безопасности*" и "*Роль пользователя*", которые широко используются для описания архитектуры многих систем безопасности.

Последняя строка таблицы прав доступа **должна** определять уровень прав доступа для всех контекстов, т.е. содержать *Маску контекста* "*".

Права доступа члена группы

Таблица прав доступа позволяет предоставить оператору доступ ко всем членам группы[751]. Для получения таких прав доступа используйте маску контекста, состоящую из пути контекста группы и суффикса `.*`, например:

```
users.admin.devgroups.my_device_group.*
```

Такая запись в таблице прав доступа позволяет получить доступ ко всем членам группы `my_device_group` с уровнем[486] прав доступа, определенном в поле **Права доступа**.

Предоставляя доступ ко всем членам группы методом, описанным выше, создается потенциальный риск безопасности! Содержимое групп может меняться операторами или даже автоматически[753], открывая новые добавленные члены пользователю с правами доступа ко всем членам группы.

Предоставляйте права доступа к членам группы операторам только в случае крайней необходимости.

9.2.3.1 Таблица прав доступа по умолчанию

При создании новой учетной записи пользователя, AtomMind Server строит для нее *таблицу прав доступа по умолчанию*. Содержание этой таблицы зависит от имени учетной записи и от содержания двух глобальных свойств AtomMind Server: Дополнительные права доступа новых пользователей[202] и Права доступа по умолчанию[202].

Последние записи

Последние три записи таблицы прав доступа нового пользователя имеют особое значение. Пример:

Маска контекста	**Уровень прав доступа**
users.*NAME_OF_USER*	Менеджер
users.*	Не определен
*	Менеджер

Первая запись объявляет уровень прав доступа **Менеджер** ко всем контекстам, определенным под контекстом Пользователь[1608] самого пользователя (например, *users.NAME_OF_USER.alerts* etc.). Данный уровень будет использоваться, если предыдущие записи не определили назначенный уровень прав доступа к ресурсу пользователя.

Вторая строка запрещает доступ к контекстам всех других пользователей AtomMind Server, установив уровень прав доступа **Не определен**. Таким образом, требуемый уровень прав доступа для контекста *users.user123.widgets* будет **Не определен** (если только новая учетная запись не окажется *user123*).

Третья строка определяет уровень прав доступа **Менеджер** для всех других контекстов, которые не относятся к учетным записям пользователей. Это не позволит пользователю выполнить действия администрирования. Например, новый пользователь не сможет просматривать события администрирования, потому что Уровнем прав доступа по умолчанию[487] контекста Администрирование[1450] является **Администратор**. Он также не сможет остановить или перезапустить AtomMind Server, потому что действия **Остановить сервер** и **Перезапустить сервер** (определенные в Корневом контексте[1559]) требуют уровень прав доступа **Администратор**.

Права доступа пользователя по умолчанию

На каждую записи таблицы Права доступа пользователя по умолчанию[202] новая запись добавляется вверху таблицы прав доступа нового пользователя. Если запись прав доступа пользователя по умолчанию отключена, пользователю назначается уровень прав доступа **Не определен** к ресурсу, указанному в записи Прав доступа пользователя по умолчанию. В противном случае, для ресурса используется уровень по умолчанию, заданный во время регистрации пользователя.

Пример таблицы прав доступа нового пользователя:

Маска контекста	**Уровень прав доступа**

users.*NAME_OF_USER*.devices	Менеджер
users.*NAME_OF_USER*.filters	Не определен
users.*NAME_OF_USER*.alerts	Менеджер
users.*NAME_OF_USER*.jobs	Не определен
users.*NAME_OF_USER*.queries	Не определен
users.*NAME_OF_USER*.dashboards	Менеджер
users.*NAME_OF_USER*.autorun	Не определен
users.*NAME_OF_USER*.favourites	Не определен
users.*NAME_OF_USER*	Менеджер
users.*	Не определен
*	Менеджер

В вышеуказанном примере только записи Устройства, Тревоги и Инструментальные панели были активированы в Правах доступа пользователя по умолчанию во время создания учетной записи.

Дополнительные права доступа для новых пользователей

Все записи из таблицы Дополнительные права доступа для новых пользователей 202 добавляются вверху таблицы прав доступа и, таким образом, имеют высший приоритет. Предположим, что у нас есть две записи в таблице Дополнительных прав доступа:

Маска контекста (используйте % вместо имени пользователя)	**Уровень прав доступа**
users.%.dashboards.specialDashboard	Администратор
users.admin.models.specialModel	Администратор

Поскольку символ % заменяется именем пользователя сразу после обработки таблицы **Дополнительные права доступа для новых пользователей**, мы получим следующую таблицу прав доступа для нового пользователя:

Маска контекста	**Уровень прав доступа**
users.*NAME_OF_USER*.dashboards.specialDashboard	Администратор
users.admin.models.specialModel	
...	...

9.2.4 Проверка прав доступа

Когда пользователь пытается получить доступ к ресурсу, AtomMind Server просматривает его таблицу прав доступа построчно, начиная с самой первой (верхней).

Если путь контекста 41 ресурса, к которому пользователь пытается получить доступ, совпадает 45 или продолжает 44 маску контекста, определенную в текущей строке таблицы прав, уровень прав доступа в этой строке будет *эффективным уровнем прав доступа* для этого пути. Как только совпадающая строка найдена, другие строки не проверяются на совпадение. Так, например, если вы начинаете таблицу с контекстом ***** (т.е. первая строка в таблице), другие строки не будут проверяться, потому что эта строка совпадает со всеми контекстами. Именно по этому контекст ***** всегда является последней строкой в таблице.

Требуемый уровень прав доступа, определенный на предыдущем шаге, сравнивается с уровнем прав доступа, запрашиваемым ресурсом, к которому осуществляется доступ. Если эффективный уровень включает 486 уровень прав доступа источника, доступ будет разрешен. Иначе, в доступе будет отказано.

Если контекстный путь запрашиваемого ресурса не соответствует или не расширяет любую маску в таблице, эффективный уровень прав доступа запрашиваемого ресурса определяется последней строкой в таблице прав доступа, потому что она всегда определяет уровень прав доступа для всех контекстов (*Маска контекста* *****).

Пример 1

Предположим, что таблица прав доступа пользователя **john** выглядит следующим образом:

Строка	Маска контекста	Права доступа
1	users.test	Менеджер
2	users.*	Не определен
3	*	Менеджер

1. John пытается получить доступ к **user.abc.alerts**. Необходимый уровень прав для данного ресурса является **Менеджер**. Первая строка таблицы, **users.test**, не совпадает и не продолжает маску **users.abc.alerts**, поэтому мы переходим ко второй строке **users.***, и видим, что она *продливает* **users.test**. Данная вторая строка задает **Не определен** в качестве уровня прав доступа, и поэтому доступ будет запрещен.

2. John пытается получить доступ к **event_filters.filter1**, требующий уровень прав доступа **Менеджер**. Эффективный уровень прав доступа для **event_filters.filter1** будет определен третьей строкой в таблице, и, таким образом, доступ будет запрещен.

3. John пытается получить доступ к **users.test.queries**, требующий уровень прав доступа Администратор. Его эффективный уровень для этого контекста определяется первой строкой в таблице. Он будет уровнем **Менеджер** (ниже, чем **Администратор**), и, таким образом, доступ к **users.test.queries** будет запрещен.

Пример 2

Таблица прав доступа [администратора](479) по умолчанию:

Строка	Маска контекста	Права доступа
1	*	Admin

Эффективный уровень прав доступа администратора по умолчанию для любого контекста - **Администратор**, потому что все контекстные пути продливают маску ***. Поэтому администратор может выполнять любую операцию в контексте.

Окно отказа в доступе

Если прав доступа пользователя недостаточно для осуществления требуемой операции, в доступе будет отказано, и появится сообщение об ошибке "Нет прав".

Временное отключение доступа к некоторым объектам

Когда таблица прав доступа пользователя настраивается так, чтобы запретить ему доступ к некоторым из ресурсов, никаким образом не влияя на сами ресурсы. Фактически, какой-либо другой пользователь может иметь доступ к этим ресурсам, даже если для их владельца доступ запрещен.

ПРИМЕР

John, пользователь из последнего примера, имеет контекст Тревоги[1452], **users.john.alerts**. Данный путь существует всегда. Изначально John может получить доступ к его Тревогам, потому что это позволяет Права доступа пользователя по умолчанию. Он входит на AtomMind Server и создает тревогу **alert1**. Затем администратор настраивает таблицу прав доступа John'a так, чтобы он больше не смог видеть его контекст Тревоги (например, добавив запись с *Маской контекста* **users.john.alerts** и *Уровнем прав доступа* **Не определен** в начало таблицы). Это не удалит **alert1** и даже не прекратит ее работу. Некоторые пользователи AtomMind Server все еще имеют доступ к этой тревоге и смогут конфигурировать и контролировать ее. Как только администратор разрешает John'у получить доступ к его тревогам, **alert1** снова появится в его контексте Тревоги.

9.2.5 Внешняя аутентификация

AtomMind Server поддерживает [плагины](207) внешней аутентификации, позволяющие проверять учетные данные пользователя через внешнюю систему.

Например, почти все инсталляционные пакеты AtomMind Server включают модули внешней аутентификации, которые [верифицируют пользователей через сервер LDAP/Active Directory](491).

9.2.5.1 LDAP-аутентификация (Active Directory)

Большие инсталляции AtomMind управляются сотнями людей, каждый из которых имеет одну или множество ролей. Создание и поддержка отдельных учетных записей пользователей[478] AtomMind Server для всех них слишком трудозатратна. В этом случае возможно аутентифицировать пользователей через сервер LDAP (такой как Microsoft Active Directory), в то время как авторизация (назначение прав доступа пользователя[477]) будет использовать учетные записи пользователя AtomMind Server на основе ролей[480].

Аутентификация LDAP включается и контролируется через общую настройку сервера Аутентификация через Active Directory и LDAP[491]. Она имеет следующие поля:

- **Адрес**. IP-адрес или имя хоста сервера LDAP.
- **Порт**. Порт для подключения к серверу LDAP, по умолчанию 389.
- **Время ожидания**. Время ожидания для процесса аутентификации LDAP. Если время ожидания истекает во время аутентификации, пользователь AtomMind Server, пытающийся зайти при помощи аутентификации LDAP, будет отклонен.
- **Тип**. Выберите ваш тип аутентификации: `Use RDN Prefix` или `Use Authentication User`.
- **Соответствие контейнеров по умолчанию**. Таблица, использующаяся для преобразования **primaryGroupID** LDAP-пользователя (из столбца **ID**) в имя группы (указано с столбце **Name**). Используется только если выбран тип аутентификации **Использовать RDN префикс**.
- **Соответствие параметров пользователя LDAP пользователям AtomMind**. Таблица, определяющая, как параметры LDAP-пользователей соотносятся с учетными записями пользователя AtomMind Server на основе ролей. После успешной аутентификации LDAP-пользователя, системе нужно будет найти соответствующего LDAP-пользователю пользователя AtomMind Server. Это происходит следующим образом:
  - Таблица **Соответствие параметров пользователя LDAP пользователям AtomMind** обрабатывается построчно
  - Для каждой записи вычисляется **Выражение значения атрибута**. Его контекст по умолчанию - контекст, указанный значением столбца **Пользователь**.
  - Результат **Выражения значения атрибута** сравнивается со значением атрибута LDAP-пользователя, указанным в столбце **Имя атрибута**. Если они совпадают, пользователь, указанный в столбце **Пользователь** авторизуется для текущей сессии.

После того, как найдена соответствующая запись в таблице **Соответствие параметров пользователя LDAP пользователям AtomMind**, обработка данных продолжается. Если будут найдены другие записи, совпадающие с текущим LDAP-пользователем (т.е. совпадение более, чем одной записи), аутентификация закончится ошибкой.

- **Префиксы RDN**. Имя RDN для поиска пользователя, без сегментов ЦОД. Это поле относится только к аутентификации по **Типу** `Use RDN Prefix`.
- **Домен по умолчанию**. Имя домена для использования, если он не был определен в имени пользователя при входе.
- **Имя пользователя.** Поле предоставляет имя пользователя. Относится только к аутентификации по **Типу** `Use Authentication User`.
- **Пароль.** Пользовательский пароль. Это поле относится только к аутентификации по **Типу** `Use Authentication User`.
- **Домен для поиска**. Домен для поиска пользователей. Либо **Использовать доменную часть имени пользователя** (например, часть логина аутентификации после символа `@`), либо **Использовать домен по умолчанию** (например, значение параметра домена по умолчанию).
- **Использовать SSL**. Определяет, будет ли использоваться безопасный протокол LDAPs.
- **Файл доверенного хранилища**. Путь к доверенному хранилищу Java, который содержит сертификат SSL для использования при подключениях по LDAPs.
- **Пароль доверенного хранилища**. Пароль от доверенного хранилища Java, который содержит сертификат SSL для использования при подключениях по LDAPs.

аутентификация с использованием RDN префикса

Вот как происходит аутентификация по **Типу** `Use RDN Prefix`:

- LDAP запросы делаются к пользователям из списка **Домен для поиска**, один запрос на каждую запись в таблице **Префиксы RDN**. Доменная часть запроса формируется путем соединения текущего **Префикса RDN** и `,DC=Lookup Domain`. Область поиска LDAP устанавливается на Поддерево. Поиск пользователей осуществляется по LDAP-параметру **userPrincipalName**.

- Все перечисленные выше запросы должны возвращать строго всего один результат, иначе аутентификация завершится ошибкой.
- Если найденный LDAP-пользователь имеет атрибут **primaryGroupID**, имя группы пользователя вычисляется путем скрининга таблицы **Соответствие контейнеров по умолчанию**. **Имя** группы, найденной в записи с **ID** то же, что и используемый **primaryGroupID**.
- Наконец, пользователь AtomMind Server ищется в таблице **Соответствие параметров пользователя LDAP пользователям AtomMind**.

аутентификация с использованием пользователя аутентификации

Вот как происходит аутентификация по **Типу** `Use Authentication User`:

- Делается LDAP-запрос, чтобы найти пользователя с LDAP-атрибутом **sAMAccountName,** равным логину аутентификации. ля выполнения запроса используются настройки **Имя пользователя** и **Пароль**.
- Пользователь AtomMind Server ищется в таблице **Соответствие параметров пользователя LDAP пользователям AtomMind** по атрибутам пользователя, найденным упомянутым запросом.

9.2.5.2 OAuth

Плагин аутентификации OAuth делает возможной авторизацию пользователей Web UI 220 через внешнюю систему, поддерживающую протокол OAuth 2.0.

Когда добавлен и настроен OAuth *поставщик,* на странице авторизации web UI появляется возможность кликнуть на имя поставщика, чтобы начать аутентификацию через стороннюю систему.

OAuth активируется и управляется через таблицу Поставщиков услуг в общих настройках 207 плагина OAuth. Таблица имеет следующие поля:

- **Поставщик**. Уникальный ID поставщика услуг. Определяется пользователем.
- **Описание**. Описание поставщиков, которые будут показаны на странице авторизации пользователя web UI. Нажатие на описание поставщика запустит аутентификацию с поддержкой OAuth через выбранного поставщика.
- **Настройки**. Настройки для поставщика услуг:

URL защищенного ресурса	URL OAuth запроса, т.е. URL веб-страницы поставщика, на которой содержится информация об авторизуемом пользователе
Идентификатор клиента	Идентификатор клиента поставщика, полученный от сторонней системы, чаще всего, *ключ API*
Секрет клиента	Пароль клиента поставщика, полученный от сторонней системы, чаще всего, *API Секрет*
Конечная точка токена доступа	URL веб-страницы поставщика, которая получает запросы токена доступа
URL авторизации	URL веб-страницы поставщика, на которую будет перенаправлен пользователь во время OAuth-авторизации
Конечная точка для отзыва токена	URL веб-страницы поставщика, который позволяет AtomMind Server оповещать сервер аутентификации, что полученный ранее токен обновления или доступа больше не нужен
Область видимости	OAuth область видимости, необходимая только некоторым API
URL обратного вызова	OAuth URL обратного вызова - адрес, на который будет перенаправлен пользователь сервером аутентификации после успешной аутентификации. Должен содержать действительное имя хоста AtomMind Server и далее `/wd?provider=provider_id`.
Идентификационный параметр токена	Имя Идентификационного параметра токена, которое будет сравниваться со свойством AtomMind Server пользователя 478, указанным в Поле идентификации пользователя, чтобы найти локальный пользовательский аккаунт для аутентификации.
Поле идентификации пользователя	Поле настроек аккаунта пользователя 481 AtomMind Server, значение которого будет сравниваться с Идентификационными параметрами токена, чтобы найти локальный пользовательский аккаунт для аутентификации.

9.2.6 Последовательность аутентификации и авторизации

Этот раздел объясняет, что происходит, когда кто-либо пытается зайти в AtomMind Server при помощи определенного имени пользователя и пароля:

1. Во-первых, сервер ищет учетную запись пользователя 478 с данным именем пользователя.
2. Если находится локальная учетная запись, пользователь аутентифицируется путем сравнения предоставленного пароля с паролем, хранящимся в настройках учетной записи:

   2.1. Если пароли не совпадают, пользователь отклоняется.

   2.2. Если пароли совпадают, пользователь авторизуется:

3. Если локальная учетная запись не найдена, предпринимается попытка аутентификации пользователя через плагин внешней аутентификации 490:

   3.1. Если Внешняя аутентификация 202 **отключена**, пользователь отклоняется.

   3.2. Если Внешняя аутентификация 202 настроена на использование определенного плагина аутентификации, запрос аутентификации перенаправляется на этот плагин. Далее плагин определяет, отклонить или принять запрос аутентификации.

4. Как только пользователь прошел аутентификацию, начинается процесс авторизации:

   4.1. Если флаг **Унаследовать права доступа** отключен в настройках учетной записи пользователя, пользователь авторизуется с использованием его собственной Таблицы прав доступа.

   4.2. Если флаг **Унаследовать права доступа** включен в настройках учетной записи пользователя, пользователь авторизуется с использованием Таблицы прав доступа аккаунта на основе ролей, указанного в настройке **Унаследовать права доступа от**.

9.3 Безопасность учетных данных внешних систем

Различные партнеры AtomMind часто задают вопрос: "почему пароли хранятся в виде открытого текста и могут просматриваться и экспортироваться операторами AtomMind?"

AtomMind является системой мониторинга и контроля и поэтому подключается и проходит аутентификацию на различных внешних устройствах, системах и базах данных. Чтобы это было возможно, AtomMind Server должен хранить значения всех учетных данных внешней системы.

По сути, этот простой факт означает, что любой посторонний, имеющий доступ уровня администратор к операционной системе устройства, на которое установлен AtomMind Server, сможет получить доступ ко всем паролям, используемым AtomMind Server для подключения к сторонним устройствам и системам.

Избежать этого не представляется возможным: даже если бы все пароли были зашифрованы, исходный код AtomMind Server всегда будет содержать достаточно информации, чтобы их расшифровать.

То же самое относится к пользователям AtomMind Server: любой пользователь AtomMind Server, имеющий неограниченный доступ 477 ко всем объектам системы (например, администратор по умолчанию 479), сможет получить доступ к паролям, используемым AtomMind Server для доступа к внешним устройствам/системам.

Политика защиты паролей

Ниже изложен ряд правил, которые необходимо соблюдать администраторам AtomMind Server во избежание раскрытия паролей.

Нижеизложенные правила предполагают, что *доверяемый пользователь* - это человек, имеющий законный доступ к любому устройству/системе, к которым подключен AtomMind Server.

Правила защиты паролей:

- Доступ на уровне операционной системы к сетевому компьютеру, на который установлен AtomMind Server, должен предоставляться исключительно доверяемым пользователям.
- Если по какой-то причине необходимо предоставить доступ к AtomMind Server недоверяемому лицу, ему нельзя предоставлять право доступа к чтению папки установки AtomMind Server и любых других папок с данными (например, папки баз банных). Так же необходимо закрыть доступ этом улицу к памяти процессов AtomMind Server.
- Любой ненадежный пользователь AtomMind Server не должен иметь доступа к переменным 61, как и прав для выполнения функций 70, хранящих пароли.

10 Связь и сбор данных

Эта глава описывает модули, функции и возможности, связанные с подключением устройств и управлением ими, а также со сбором данных.

Есть два основных способа сделать так, чтобы существующее или новое устройство работало с AtomMind:

- Первый способ - при помощи драйвера устройства 518, который преобразует протокол устройства в формат единой модели данных 41 на стороне сервера. Использование драйвера подразумевает, что устройство и сервер общаются через протокол связи устройства. Драйвер позволяет AtomMind Server правильно анализировать и 'понимать' данные с определенного устройства, и тем самым делает их доступными через различные средства AtomMind Server (события, отчеты, модели и т.д.). Это либо стандартный 518 драйвер, либо специальный Java 1345 / Low-code 551 программный компонент, работающий на стороне сервера.
- Второй способ - с помощью агента 684, который преобразует данные с устройства в формат единой модели данных 41 внутри самого устройства и/или периферийного шлюза, к которому устройство подключено. При использовании агента, связь между ним и сервером AtomMind осуществляется через IP-сеть с использованием протокола AtomMind 2119. Агент - это программная библиотека, наследованная в прошивку шлюза/устройства, или отдельное небольшое приложение, работающее на модуле связи, встроенном в устройство. Такое программно-аппаратное сочетание выступает посредником между "основной" прошивкой аппаратного устройства и AtomMind Server.

ПОдключение устройства с использованием драйвера устройства

Этот раздел описывает сценарии подключения устройств через драйверы устройств.

(1) ПОДКЛЮЧЕНИЕ УСТРОЙСТВ С ИСПОЛЬЗОВАНИЕМ СТАНДАРТНОГО ПРОТОКОЛА

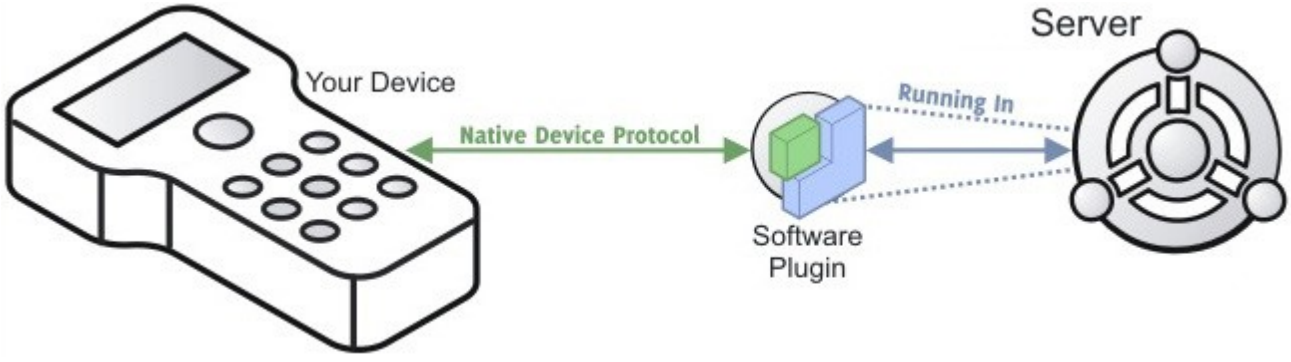
Если ваши устройства поддерживают один из стандартных протоколов связи (например SNMP), они могут быть непосредственно подключены к AtomMind, используя встроенный драйвер устройства 518. Преобразование протоколов ПО или оборудования в данном случае не требуется. Просто установите все необходимые физические соединения и ваше устройство готово к работе с AtomMind Server.

(2) СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ДРАЙВЕРА УСТРОЙСТВА

Вы также можете решить проблему непосредственно на уровне ПО. Драйвер - это компонент ПО, позволяющий AtomMind "понимать" протокол уже существующего устройства.

В большинстве случаев Flexible драйвер 551 делает возможной коммуникацию с устройством и синтаксический анализ протокола в режиме "low code" (не прибегая к программированию).

Для очень сложных протоколов устройств используйте Набор разработки драйверов 1345, чтобы написать код на языке Java.



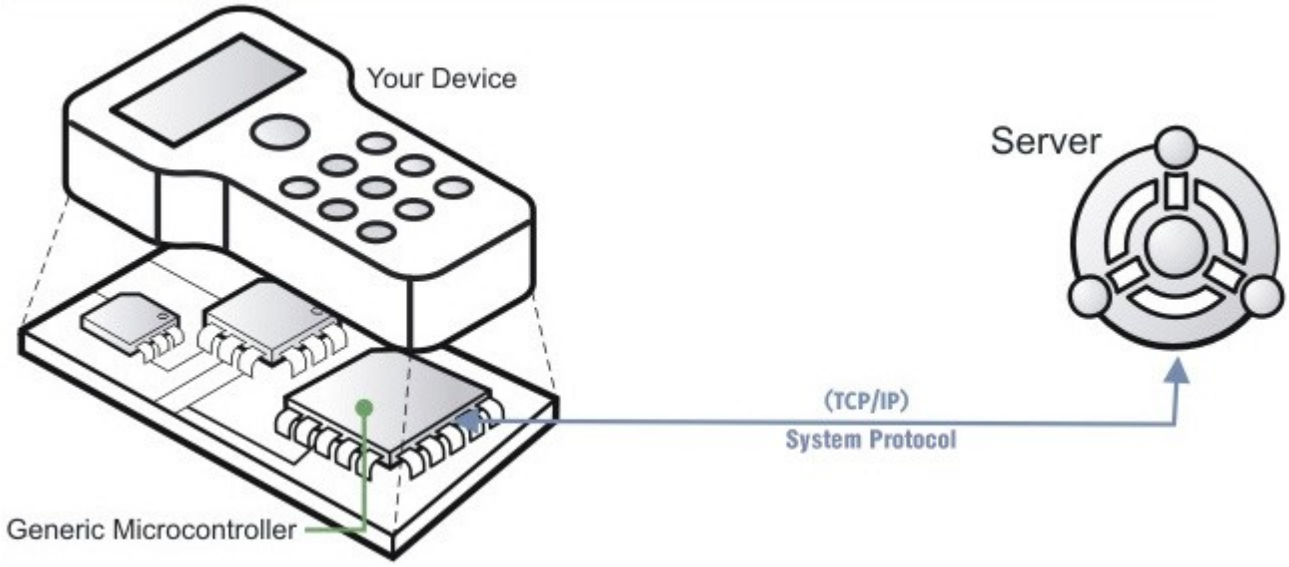
Подключение устройств при помощи агента

Этот раздел описывает сценарии использования агента 684.

(1) РАЗРАБОТКА НОВОГО УСТРОЙСТВА С ИПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ АГЕНТА

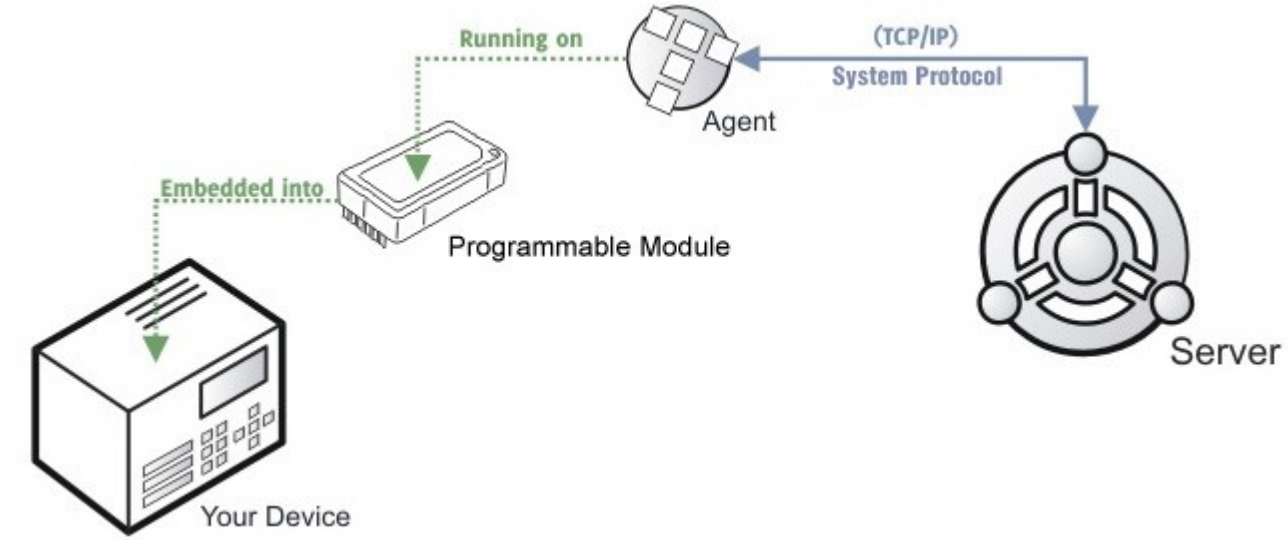
Этот метод предполагает встраивание программной библиотеки Агента во внутреннее ПО/прошивку вашего устройства. Библиотека обеспечит перевод данных на язык протокола AtomMind, а также соединение с AtomMind Server и передачу данных. Библиотека Агента реализована на многих языках программирования, включая Java, .NET, C++ и другие.

Протокол связи AtomMind 2119 является открытым и хорошо задокументированным. В случае создания крупных приложений может быть выгодна полная реализация поддержки данного протокола для микроконтроллеров, которые Вы уже используете. Тогда эти микроконтроллеры смогут соединиться с AtomMind так же, как и библиотека Агента. Для системы не будет никакого различия, и устройство, основанное на вашем микроконтроллере, будет без проблем работать как часть системы.

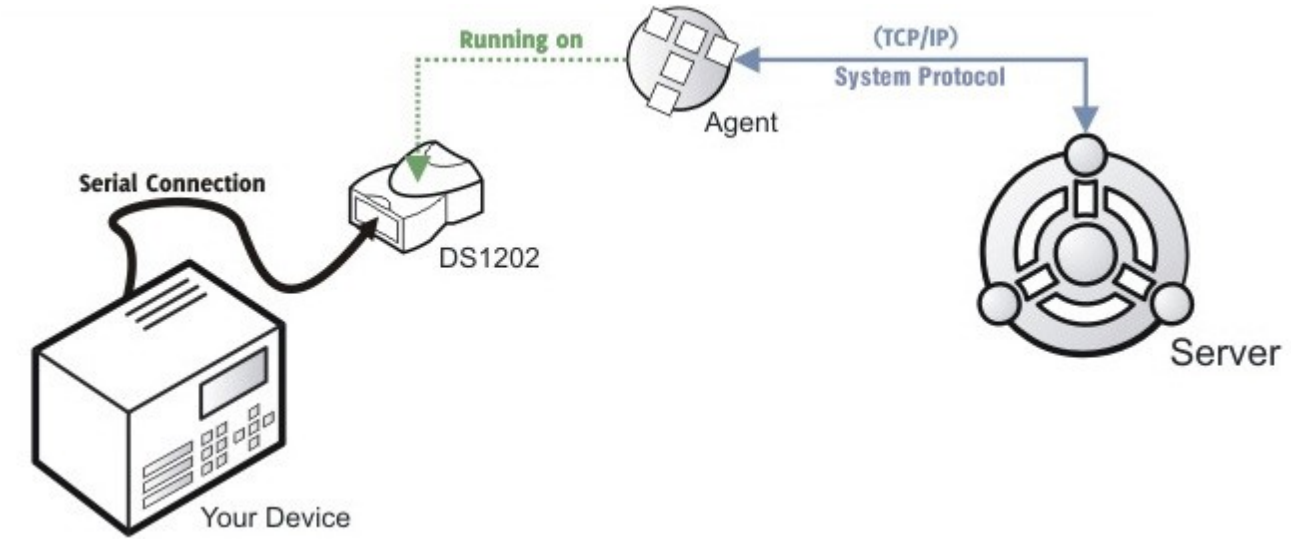


(2) ПОДКЛЮЧЕНИЕ УЖЕ СУЩЕСТВУЮЩИХ УСТРОЙСТВ К АГЕНТУ ATOMMIND

При использовании данного метода, вы встраиваете модуль ТВЭЛ, такой как EM1000, в ваше устройство. Этот модуль выполняет BASIC-приложение - Agent библиотека. Вы изменяете исходный код приложения Agent так, чтобы оно взаимодействовало с вашим устройством, "понимая" его протокол связи. После этого, приложение используется как прозрачный интерфейс между AtomMind и вашим устройством, что позволяет получить доступ ко всем настройкам устройства, данным и событиям внутри AtomMind.

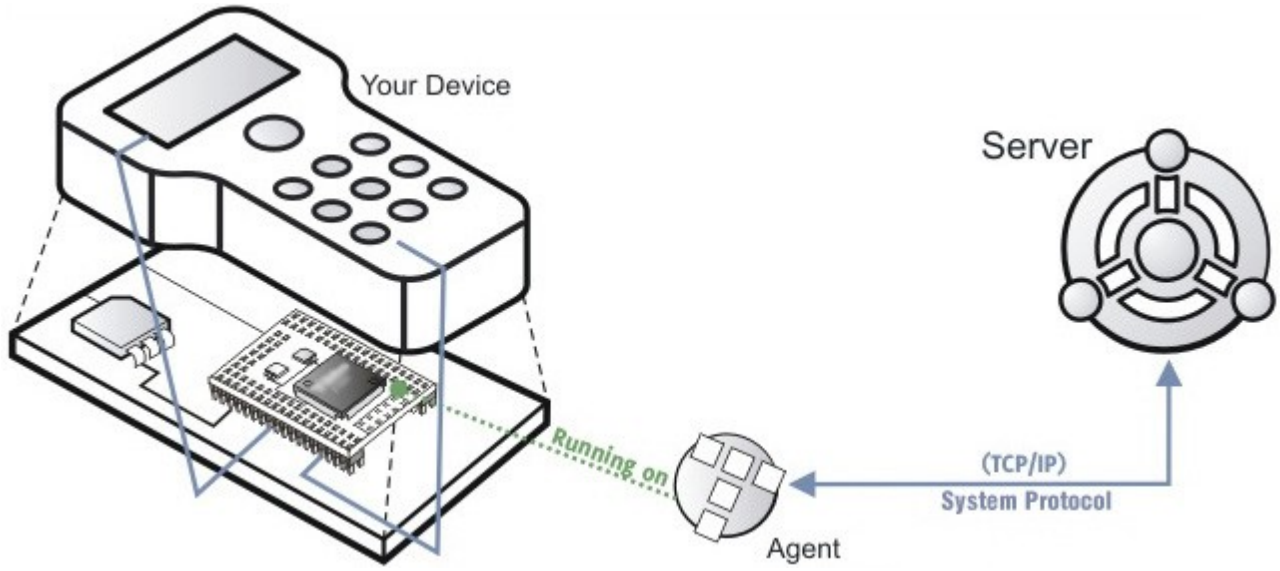


Обратите внимание, что если по какой-то причине вы не можете встроить устройство в уже существующую схему, вы всегда сможете использовать внешний BASIC-программируемый контроллер (такой как DS1202, автономная версия EM1202), выполняющий приложение Agent.



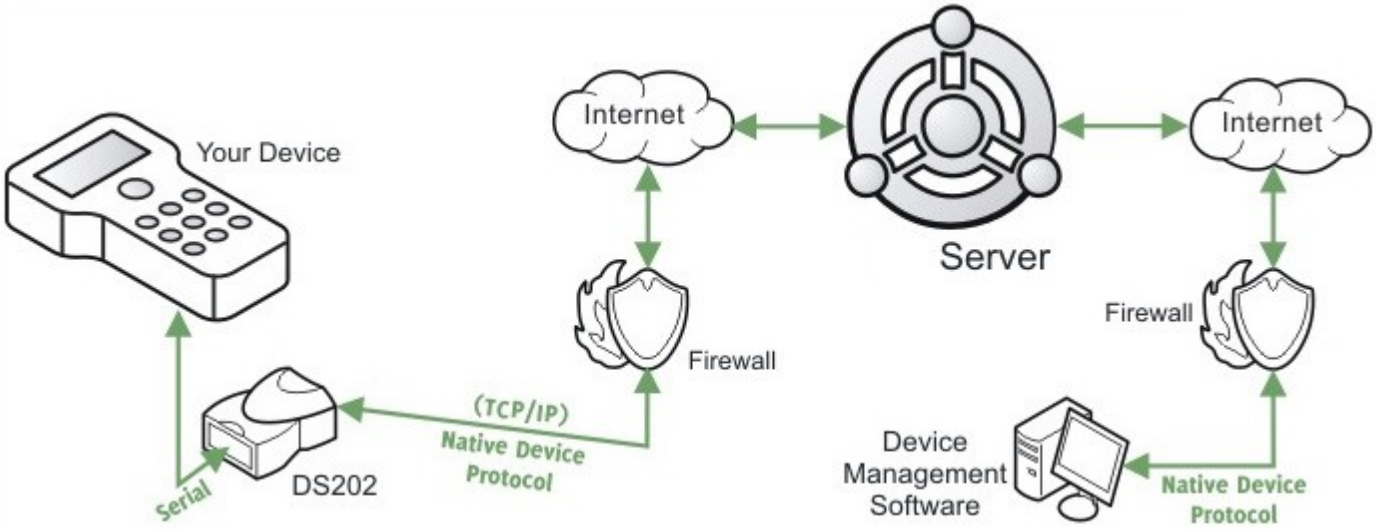
(3) РАЗРАБОТКА НОВОГО УСТРОЙСТВА НА ОСНОВЕ ПРОГРАММИРУЕМОГО МОДУЛЯ

Модуль ТВЭЛ, такой как EM1000, является достаточно мощным и может служить центральным процессором для вашего приложения. Вы можете непосредственно соединить его с датчиками и схемами вашего устройства, и использовать в качестве главного ЦП, локально управляя работой вашего устройства. По сути, вы берете приложение Агент 684, значительно расширяя и настраивая его. Таким образом, вы экономите на стоимости ЦП для вашего устройства, и сохраняете возможность соединения его с AtomMind с использованием всех преимуществ системы.



использование AtomMind для передачи "сырых" данных (без обработки)

Возможно, вы хотите использовать AtomMind для передачи данных между вашими устройствами и/или компьютерами, работающими с особым клиентским ПО. В этом случае, программе AtomMind не нужно "понимать", хранить или обрабатывать данные, которые необходимо передать. *Сервер устройств* преобразовывает данные из поддерживаемого устройством формата (RS232, USB и др.) в Ethernet-трафик и направляет их от/к AtomMind Server.



Данный режим не использует в полную силу возможности AtomMind по обработке данных, но может быть весьма полезным в нескольких случаях:

- Когда устройства, расположенные в различных локальных сетях, защищенных брандмауэрами, не могут установить прямое соединение друг с другом. В этом случае, AtomMind Server выступает в роли посредника и передает данные между этими устройствами.

- Когда устройства не имеют постоянного IP-адреса. В этом случае, устройства можно зарегистрировать в DNS с помощью AtomMind Server и использовать их имена хостов вместо IP-адреса.

Иногда устройства, расположенные в защищенных сетях, имеют встроенный web-сервер. Непосредственный доступ к таким web-серверам получить нельзя, но AtomMind Server предоставляет HTTP-прокси сервис, позволяющий установить безопасное HTTP соединение с сервером вместо непосредственного соединения с устройством. AtomMind Server переадресует данное соединение на устройство.

Для получения дополнительной информации, просмотрите Управление серверами устройств 208 .

10.1 Устройства

Устройство - это электронное оборудование или другой источник данных, настраиваемый и контролируемый AtomMind. "Представление" устройства в AtomMind Server называется *"Аккаунт Устройства".*

AtomMind Server работает с разными видами устройств, используя Драйверы Устройств 518 . Эти драйверы знают, как "общаться" с определенным устройством, используя его собственный протокол. Каждый драйвер устройства обеспечивает *нормализованное* представление одного или более Device в виде Контекстов 41 . Данный процесс нормализации включает следующие действия:

- Создание одного Контекста 1494 Device для каждого Device, который работает с AtomMind.
- Преобразование внутренних настроек и конфигурации Device в контекст Переменных (Свойств) 61 .
- Преобразование операций управления, проводимых Device, в контекст Функции 70 , и соответствующие действия 87 , помогающие выполнять данные функции в диалоговом режиме (действия вызова функции) 103 .
- Преобразование событий, созданных Device, в контекст События 73 .

Все эти операции помогают обеспечить единые методы конфигурации, контроля и мониторинга Device в пределах инфраструктуры AtomMind. Когда Device представлено как контекст с переменными, функциями и событиями, вы можете использовать все особенности управления и мониторинга AtomMind (Тревоги 779 , Отчеты 928 , Виджеты 943 и т.д.) для работы с ним.

Администрирование Device

Два контекста[41] используются для администрирования Device: одним из них является общий контекст Device, который выступает в роли контейнера для всех Device или одного пользователя[478]. Другим является контекст Device, относящийся к одному Device.

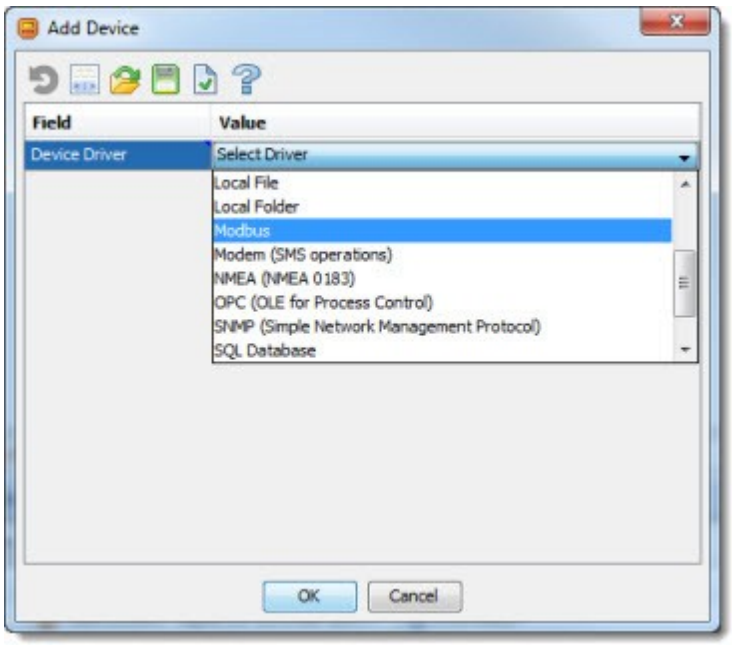
Существуют два дополнительных контекста, относящихся к Device: контекст **Все Devices** под контекстом Пользователь[1608] и **Все Devices** под контекстом Корень[1553]. Данные контексты обеспечивают доступ к группированным действиям[101] всех Device, принадлежащим к определенной учетной записи пользователя[478], и ко всем устройствам в системе соответственно.

Сопутствующие ссылки:

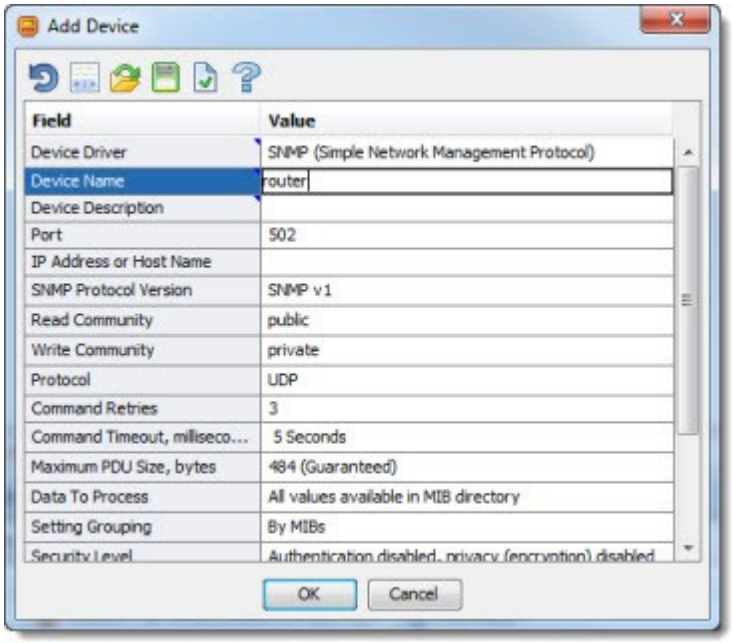
- Создание виджета для управления устройством[1643]
- Создание инструментальной панели для контроля устройств в реальном времени[1660]
- Управление параметрами устройства, используя график[1666]
- Построение отчета о статусе устройства[1678]
- Добавление пользовательских настроек[1686]
- Составление графика простоя устройства[1697]

10.1.1 Добавление новых устройств

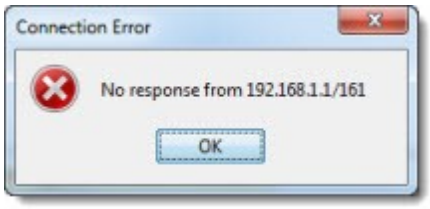
Новое устройство подключается к системе при помощи опции Добавить устройство[1492] в контексте Device. Для начала выберите Драйвер устройства[518]:



Теперь необходимо ввести **Имя** и **Описание** учетной записи устройства и уточнить параметры подключения:



Нажмите OK для проверки соединения с устройством. Если соединение не установлено, вы увидите диалоговое окно сообщения об ошибке:



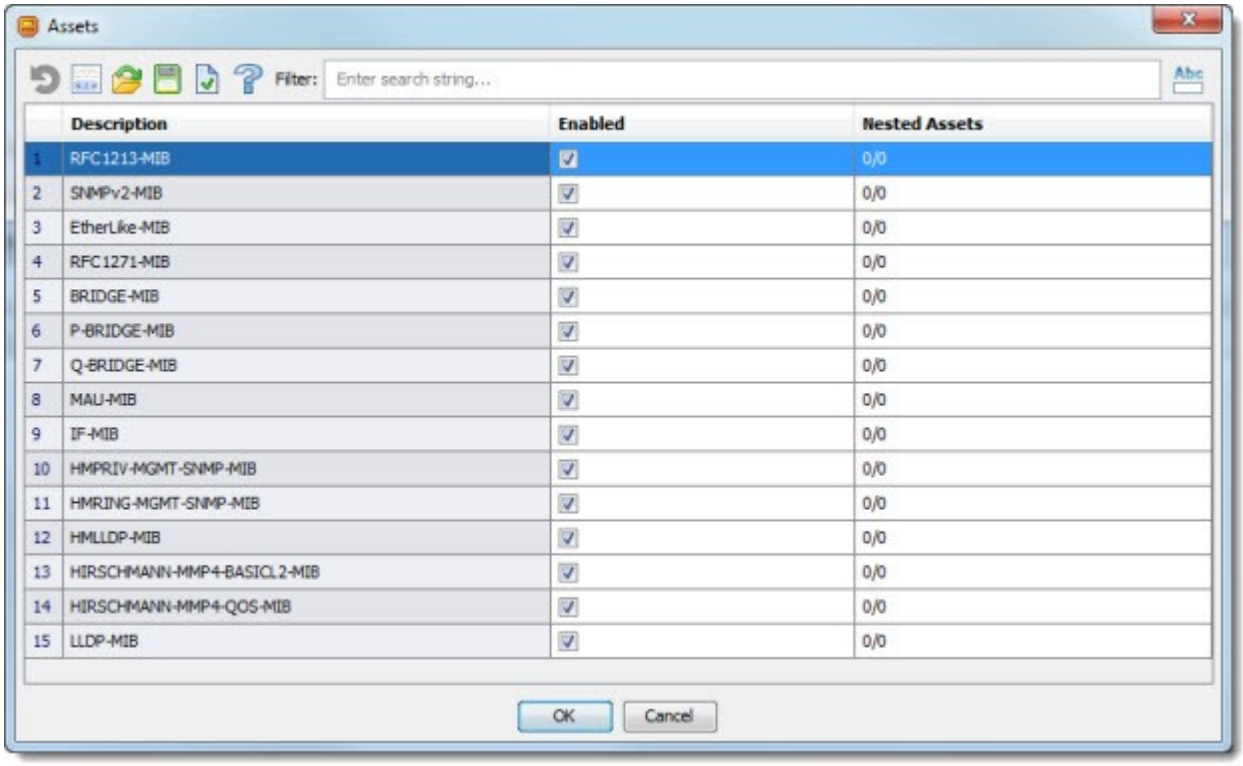
В случае ошибки система разрешит исправить параметры подключения и подключиться повторно.

Чтобы пропустить проверку соединения для последующего создания учетной записи, просто нажмите **Отмена**, когда диалоговое окно со свойствами подключения появится во второй раз. Вы сможете изменить параметры подключения позже.

Как только соединение с устройством установлено, система считывает описания активов 501, полученных от устройства. Это может занять несколько минут.

Если драйвер устройства не поддерживает выбор актива, это шаг будет пропущен.

Когда чтение актива завершено, система показывает оператору дерево активов устройства, позволяя выбрать, какие активы будет контролировать AtomMind:



Нажмите OK для сохранения выбора и завершите создание устройства.

По завершению процесса создания появится новый Контекст 149 Device, представляющий данное устройство в системе. Вот так это выглядит в Системном Дереве 370 AtomMind Client :

Затем вам будет предложено определить **свойства синхронизации** и другие **параметры устройства.** Нажмите OK для закрытия диалогового окна свойств устройства и завершите регистрацию.

Как только регистрация учетной записи завершена, устройство будет синхронизировано 514 с сервером. После первой полной синхронизации все настройки, операции и события устройства будут кэшированы 502 и доступны для других средств системы, таких как тревоги или виджеты. Устройство перейдет в режим **Онлайн**, статус 515 **Синхронизирован**, что будет отмечено иконкой в виде зеленой галочки: .

10.1.2 Метаданные устройства

Каждое устройство предоставляет некоторые или все объекты данных из следующего списка:

- Настройки 501 устройства, доступные в форме переменных 61 контекста Устройство.
- Операции 509 устройства, доступные как функции 70 контекста Устройство. AtomMind Server также создает одно действие 87 на каждую операцию устройства для удобного интерактивного или неинтерактивного выполнения этих функций.
- События 510 устройства, т.е. события 73 контекста Устройство.
- Активы 501 устройства для логической группировки настроек, операций и событий в иерархическом дереве группы.

Общее название настроек, операций, события и активов устройства - *метаданные устройства*.

Метаданные извлекаются из устройства во время полной синхронизации 514.

10.1.3 Активы устройств

Современные устройства имеют сотни и даже тысячи настроек, операций и типов событий. Такое огромное количество ресурсов требует некой группировки для удобного пользования. В AtomMind группы ресурсов устройств называются *Активами Устройств.*

Активы организованы в виде иерархического дерева. Если активы поддерживаются определенным драйвером устройства 518, то он предоставляет список *корневых активов,* каждый из которых имеет список *дочерних активов.* Системные операторы могут включать/отключать активы для уточнения, что обрабатываются только необходимые данные устройства.

Ниже представлены некоторые примеры активов устройств:

- Для SNMP 637 устройств в активе расположен MIB файл, поддерживаемый устройством. Его отключение останавливает чтение всех идентификаторов OID, определяемых MIB файлом.
- Каждый актив для устройства WMI 661 является единым классом WMI 661. При отключении данного актива система выявляет свойства/методы/события всех экземпляров данного класса.
- Для устройств BACnet 532 корневые активы представляют типы объекта BACnet 532, в то время как дочерние активы согласуются с экземплярами объекта. Таким образом, возможно отключение обработки как типов всего объекта, так и отдельных экземпляров.
- Каждая группа свойств OPC Сервера 617 представлена отдельным активом. Дочерние активы являются подгруппами свойств.

Управление Активами Устройств

Существуют два способа включения/отключения активов устройств:

- Список активов может быть изначально скорректирован во время создания учетной записи устройства 498.
- Управление активами уже существующих учетных записей устройств осуществляется через действие Изменить свойства устройства 1494.

Повторное чтение активов устройства

Обычно список активов устройства считывается с аппаратного оборудования только один раз во время создания учетной записи устройства и кэшируется на сервере. Он не считывается заново даже во время полных циклов синхронизации 514 для увеличения производительности системы. Для принудительного повторного чтения информации об активах устройства (например, после изменения конфигурации) используйте действие Перезагрузка драйвера устройства 1495.

Если свойство Режим Чтения Метаданных 511 учетной записи устройства обозначено как **Читать все**, активы считываются заново при каждом цикле синхронизации.

10.1.4 Переменные устройств (настройки)

Переменные устройств (также называемые *переменные настроек устройств* или *настройки устойств*) - это переменные 61 контекста устройства, которые устанавливают соответствие с настройками аппаратного устройства при помощи соответствующего драйвера устройства 518.

Важно отличать **переменные настроек устройств** (таких как измеряемая температура) от **переменных контекста устройств, которые соответствуют свойствам конфигурации учетной записи устройств** (например, IP-адрес устройства и номер порта). Переменные настроек устройств считываются/записываются из/в фактическое аппаратное устройство или источник данных, в то время как переменные свойств учетной записи просто определяют серверные параметры связи устройств.

Переменные настроек устройств имеют некоторые особенности:

- Во время первичной синхронихации 514 устройства сервер создает моментальный снимок устройства 502, т.е. кэш серверных настроек, содержащий самые последние значения настроек устройств, полученные с аппаратного устройства или источника данных.
- Каждая переменная настроек устройств может иметь период пользовательской синхронизации (опроса), время хранения истории и другие предварительные опции синхронизации 502.
- Каждая переменная настроек устройств имеет определенный статус синхронизации 506. Эти статусы синхронизации определяют объединенный статус синхронизации всей учетной записи устройства 516.
- Как только переменная настроек устройства считывается любым системным модулем или внешним приложением, ее значение извлекается из кэша северных настроек устройств 502 и возвращается. Не происходит ввода/вывода устройств до тех пор, пока настройка не использует режим прямой синхронизации 503.

- Как только переменная настроек устройств записывается любым системным модулем или внешним приложением, новое значение записывается в кэш серверных настроек устройств и запрашивается частичная синхронизация[515]. Не происходит синхронного ввода/вывода устройств до тех пор, пока настройка не использует режим прямой синхронизации[503].
- В добавок к настройке периода хранения базы данных[692] для "сырых" исторических значений переменных настроек устройств возможно создать статистические каналы[507] для их объединения в циклическую базу данных, чтобы обеспечить компактное хранение и быстрый доступ.

Примеры настроек устройств:

- "Температурная" настройка термометра только для чтения, поддерживающая протокол Modbus TCP.
- Табличная настройка хоста сети SNMP только для чтения, представляющая собой таблицу интерфейсов сети устройств.
- Перезаписываемая настройка "Ориентация" терминала контроля доступа. Настройку можно выставить как "Рабочий стол" или "Настенное крепление".

10.1.4.1 Моментальные снимки устройств (кэш настроек)

Когда AtomMind Server впервые считывает значения внутренних настроек устройства, он сохраняет эти значения в базе данных. Это называется созданием *моментальных снимков устройства* или *кэшированием настроек*. Пользователи затем могут просмотреть и изменить настройки для любого устройства, даже если оно временно не подключено к AtomMind Server.

НОВЫЙ ТЕРМИН: *Кэш* - это набор данных, дублирующих оригинальные значения, сохраненные где-либо или обработанные ранее. Кэш используется, когда оригинальные данные трудно извлечь или обработать (обычно с точки зрения времени доступа), в то время как чтение кэша "выгоднее" (т.е. легче и быстрее).

Когда вы просматриваете или изменяете настройки устройства, в действительности вы работаете с моментальным снимком устройства. Напрямую изменить актуальные настройки устройства невозможно (если только не включен прямой режим синхронизации)[503]. Новые настройки записываются в устройство во время следующей синхронизации.

Кэш настроек периодически синхронизируется[514] с устройством. Синхронизация происходит в следующих случаях:

- Когда устройство подключается к AtomMind Server
- Когда значения настроек изменяются в кэше (например, используя AtomMind Client) при подключенном к серверу устройстве, если настройка Начать синхронизацию в период изменения настроек[510] установлена на true (значение по умолчанию)
- Периодически, как определено настройкой Период Синхронизации[510]

Каждая настройка синхронизируется по следующему алгоритму:

- Если значение настройки не существует в кэше сервера, оно кэшируется (т.е. значение считывается с устройства и сохраняется в кэше сервера)
- Если значение настройки в кэше сервера было изменено позже, чем на устройстве, кэшированное значение записывается на устройство
- Если значение настройки в устройстве было изменено позже, чем в кэше сервера, кэшированное значение заменяется текущим значением настройки устройства
- В остальных случаях никакие действия не выполняются

10.1.4.2 Параметры синхронизации настроек

Возможно определить опции пользовательской синхронизации для каждой из настроек Device. Доступ к предварительным опциям синхронизации осуществляется через опцию изменить настройки[1494] Device контекста Device. Ниже приведен список доступных опций синхронизации для каждой настройки Device:

Опция	Описание

Режим синхронизации	Существует несколько режимов синхронизации: • **Обычная синхронизация.** Значение настройки синхронизируется между AtomMind Server и Device во время цикла синхронизации. • **Синхронизация отключена.** Для данной настройки не выполняется синхронизация. • **Только от устройства к серверу.** Заставляет сервер делать серверное значение доступным только для чтения и запрещает устройству записывать операции для данной настройки. Это оказывается полезным, когда устройство/драйвер сообщает о настройке как о записываемой, хотя на самом деле операция по записи не была произведена (например, из-за отсутствия необходимых прав доступа). • **Прямой доступ к Device.** Значение настройки считывается напрямую с Device, когда его запрашивают компоненты системы. Если некоторые системные компоненты пытаются модифицировать системные значения, изменения также записываются прямо на Device. При такой настройке считывание, запись операций могут быть не произведены в случае, если не осуществлен ввод/вывод Device или Device сообщает об ошибке. • **Прямая запись Device**. Так же, как и выше, но применяется только для операций записи. Чтение значения производится из кэша настроек 502, как и в режиме Обычной Синхронизации. • **Использовать заданное значение сервера**. В этом режиме синхронизация от сервера к устройству всегда выполняется для этой настройки. Таблица Данных, возвращенная **Выражением Заданного Значения** (см. ниже), закладывается в устройство вместо значения, взятого из кэша настроек. • **Игнорировать время модификации.** Игнорирует временные отметки модификаций настройки как сервера, так и устройства. Настройка синхронизируется от сервера к устройству, если значение было изменено на сервере, или же от устройства к серверу. • **Нестандартный.** Использует средство контроля пользовательской синхронизации, предлагаемое системой, при наличии такого. Этот режим обычно включается автоматически системой, если средство контроля доступно, но может быть запущен также и вручную.
Время хранения истории	Если данная опция установлена на ненулевое значение (т.е. хранение истории обновлений активировано), постоянное событие Обновленный 84 будет появляться каждый раз, когда заданное значение изменено в серверном кэше. Это может происходить, если новое значение, отличающееся от предыдущего, считывается с удаленного устройства, или какой-либо компонент системы модифицирует кэш. Событие обновления содержит в себе значение переменной, поэтому его история 73 может использоваться как источник информации для отчетов 928 и таблиц 1051. Сохранение истории отключено по умолчанию во избежание лавинообразного увеличения базы данных.
Режим записи истории	Определяет, какие значения сохранены в моментальном снимке устройства и в хранилище истории. Отрицательные и нулевая опции имеют особое значение.

Режим истории	Описание	Значение
0	Все нормальные значения	Все нормальные значения, включая дупликаты, сохранены.
-1	Только измененные значения	Сохранены только нормальные измененные значения. Дупликаты (повторяющиеся значения) пропускаются.
-2	Измененные, ложные	Сохраняются измененные и ошибочные значения (образцы в плохом качестве).
-3	Все нормальные и неподключенные	Сохраняются все нормальные (различные) значения. Значения также сохраняются, когда устройство отключено.
-4	Все нормальные и ошибочные	Сохраняются все нормальные (различные) значения. Также сохраняются значения с плохим качеством.
-5	Измененные, неподключенные и ошибочные	Нормальные значения сохраняются, если они отличаются. Однако сохраняются значения с плохим качеством. Значения также сохраняются, когда устройство отключено.

	<table><tr><td>-6</td><td>Все значения</td><td>Сохраняются все нормальные значения (включая дупликаты) и значения с плохим качеством. Значения также сохраняются, когда устройство отключено.</td></tr></table> Если **Режим записи истории** выставлен на любое положительное целое число, исторические значения будут сохраняться в базу данных только каждый энный цикл синхронизации, где N - это значение Режима Записи Истории. Однако события обновления генерируются во время каждого цикла синхронизации, и заинтересованные стороны будут уведомляться обо всех изменениях значений.  Вы также сможете контролировать тот факт, как систему оповещают об изменениях значений: о каждом обновлении значений переменных или только об изменениях (см. **Обновления доставки**).  Если **Время Хранения Истории Обновлений** не установлено (т.е. история не сохраняется), период синхронизации скорее короткий, а у самой настройки сложный формат (т.е. много полей или вложенных таблиц). Настройка **Режим записи истории** во **Всех нормальных значениях** может улучшить производительность, потому что сервер не будет загружать старые кэшированные значения и сравнивать их с новыми во время каждой синхронизации.
Период синхронизации	Данная опция может использоваться для определения периода пользовательсьской синхронизации у определенной настройки. Значение по умолчанию - NULL (<Не установлено>), поэтому значение настройки синхронизируется во время полного цикла синхронизации. Пользовательская синхронизация полезна, когда необходимо прочитать некоторые быстроменяющиеся данные с Device.
Обновления доставки	Этот флажок контролирует отправление событий обновления переменной. Когда активирован, событие обновления переменной будет инициироваться для каждой синхронизации, даже если значение переменной остается таким же (см. **Режим записи истории**).
Состояние	Если определено выражение 112 состояния, синхронизация будет пропущена, если возвращается значение false. Полезно для отключенной синхронизации в период технического обслуживания, ночных часов и т.д. <table><tr><td colspan="2">Среда вычисления 114 выражения состояния:</td></tr><tr><td>Контекст по умолчанию 119</td><td>Контекст текущего устройства.</td></tr><tr><td>Таблица данных по умолчанию 120</td><td>Текущее значение переменной, которое хранится в кэше настроек 502.</td></tr><tr><td>Ряд по умолчанию 119</td><td>0</td></tr><tr><td>Переменные среды 123</td><td>Только стандартные 123 переменные.</td></tr></table>

Фильтр	Если определено выражение 112 фильтра, то будут фильтроваться значения, собранные как с устройства, так и с сервера: • Если значение синхронизируется от устройства к серверу и выражение фильтра возвращает FALSE, значение не будет записано в кэш 502 сервера. Это помогает распознать "плохие" значения устройства и удалить их. • Если значение синхронизируется от устройства к серверу и выражение фильтра возвращает Таблицу данных 49, оно будет рассматриваться как конвертированное значение настроек устройства. Это конвертированное значение будет записано в кэш сервера вместо значения, возвращенного драйвером. Это подходит для фильтрации табличных данных, нормализации диапазона и т.д. • Если значение изменяется системным оператором, серверным компонентом или внешней системой и выражение фильтра возвращает FALSE, запрос на изменение прошел с ошибкой.

Среда вычисления 112 выражения фильтра

Контекст по умолчанию 119	Контекст текущего устройства.
Таблица данных по умолчанию 120	Текущее значение, взятое от устройства или же предоставленное системным оператором, серверным модулем или внешней системой.
Строка по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

Пример: `abs({.:temperature$celsius} - {celsius}) > 0.5`

Это выражение фильтра сравнивает температуру по Цельсию, извлеченную во время предыдущей синхронизации, с новым значением, только что полученным из устройства. Синхронизация происходит, только если разница между ними превышает 0.5 градусов, позволяя избежать колебания значения сервера.

Выражение заданного значения	Выражение 112, используемое для извлечения значения и записи его на устройстве, если активен режим синхронизации **Использовать заданное значение сервера**.

Среда вычисления 112 выражения заданного значения:

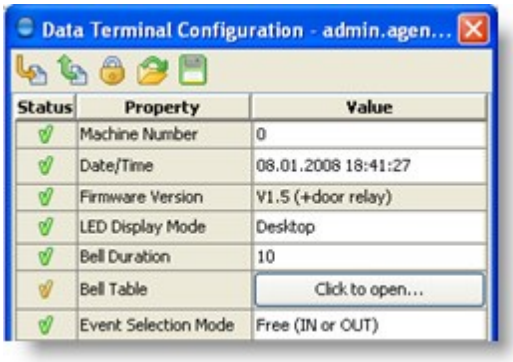
Контекст по умолчанию 119	Контекст текущего устройства.
Таблица данных по умолчанию 120	Отсутствует.
Строка по умолчанию 119	0

	Переменные среды 123 — Только стандартные 123 переменные
Добавить предыдущее значение в событие обновления переменной	Флаг, указывающий, что событие обновления 84 переменной будет содержать предыдущее значение переменной

Все это доступно для просмотра через переменную settingSyncOptions 1499.

10.1.4.3 Статус синхронизации настроек

Каждое значение настройки в серверном кэше содержится в переменной 61 в серверном контексте 41, относящемся к Device. Когда настройки просматриваются или изменяются (например, через Редактор свойств 377 AtomMind Client), статус синхронизации каждой настройки отображается в виде описания настройки.



Доступные статусы синхронизации

Статус синхронизации отображается в данных значениях:

Иконка	Статус	Описание
	Синхронизировано от Device к серверу	Показывает, что значение получено от Device и сохранено в серверном кэше.
	Синхронизировано от сервера к Device	Показывает, что значение было изменено в серверном кэше и записано на Device.
	Ожидание синхронизации	Указывает, что значение в серверном кэше было изменено во время последней синхронизации, но еще не было записано на Device.
	Ошибка синхронизации	Указывает, что произошла ошибка во время последней попытки синхронизации настройки.
	Синхронизация отключена	Режим синхронизации 503 настройки установлен на **Синхронизация отключена**.
	прямой доступ к Device	Режим синхронизации настройки установлен на **Прямой доступ к Device**.

10.1.4.4 Статистика настройки устройства

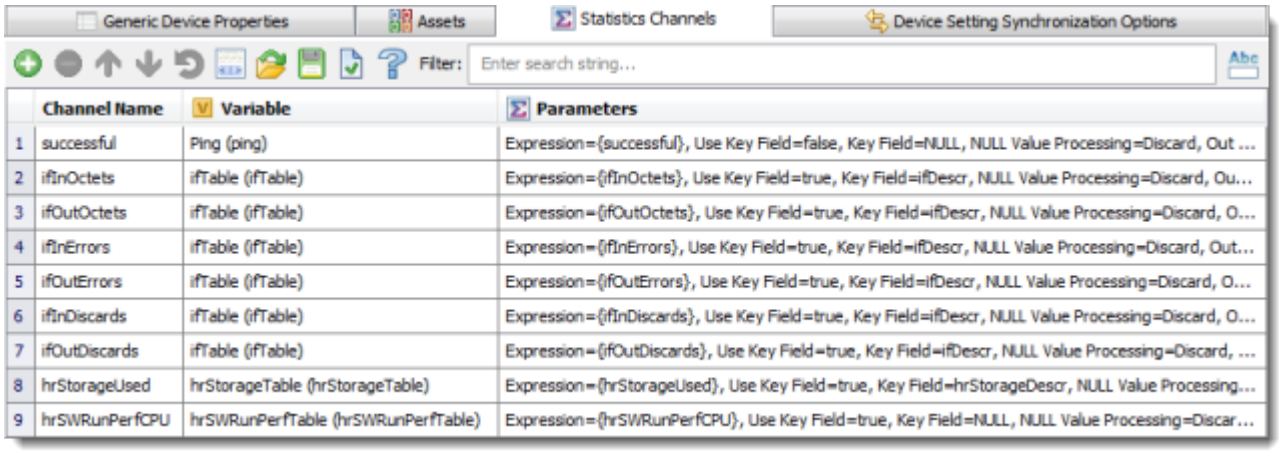
AtomMind Server позволяет создавать каналы статистики 718 для переменных настроек устройства. Каналы статистики хранят долгосрочную собранную историю для различных переменных, таких как трафик, мощность, энергозатраты, напряжение, уровни шума, температура, счетчик посетителей и другие.

Каналы управления

Каналы статистики устройства могут контролироваться через функцию Изменить свойства устройства 1494. Это обеспечивает доступ к таблице **Статистики** (Σ), содержащей описание каналов. Каждое описание включает в себя следующее:

- **Название канала.** Уникальное имя канала, используемое для дальнейшей ссылки на него, например, во время создания таблиц 1051.
- **Переменная**. Имя переменной 61, на которой основан канал. Иначе говоря, AtomMind Server создает новый срез первичных данных 727 для канала во время каждого изменения переменной
- **Активирован**. Контролирует, активен ли канал и собирает ли он данные. Каналы по умолчанию нужно деактивировать вместо их удаления, чтобы предотвратить их повторное создание.
- **Параметры**. Параметры 719 сбора статистических данных, их хранения и обработки.

Вот так выглядит список каналов в AtomMind Client:

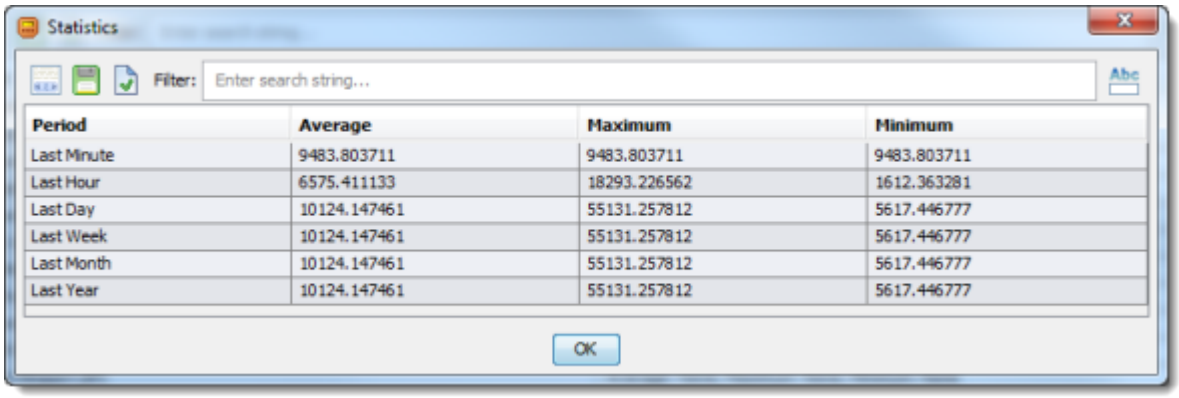


	Channel Name	Variable	Parameters
1	successful	Ping (ping)	Expression={successful}, Use Key Field=false, Key Field=NULL, NULL Value Processing=Discard, Out ...
2	ifInOctets	ifTable (ifTable)	Expression={ifInOctets}, Use Key Field=true, Key Field=ifDescr, NULL Value Processing=Discard, Ou...
3	ifOutOctets	ifTable (ifTable)	Expression={ifOutOctets}, Use Key Field=true, Key Field=ifDescr, NULL Value Processing=Discard, O...
4	ifInErrors	ifTable (ifTable)	Expression={ifInErrors}, Use Key Field=true, Key Field=ifDescr, NULL Value Processing=Discard, Out...
5	ifOutErrors	ifTable (ifTable)	Expression={ifOutErrors}, Use Key Field=true, Key Field=ifDescr, NULL Value Processing=Discard, O...
6	ifInDiscards	ifTable (ifTable)	Expression={ifInDiscards}, Use Key Field=true, Key Field=ifDescr, NULL Value Processing=Discard, O...
7	ifOutDiscards	ifTable (ifTable)	Expression={ifOutDiscards}, Use Key Field=true, Key Field=ifDescr, NULL Value Processing=Discard, ...
8	hrStorageUsed	hrStorageTable (hrStorageTable)	Expression={hrStorageUsed}, Use Key Field=true, Key Field=hrStorageDescr, NULL Value Processing...
9	hrSWRunPerfCPU	hrSWRunPerfTable (hrSWRunPerfTable)	Expression={hrSWRunPerfCPU}, Use Key Field=true, Key Field=NULL, NULL Value Processing=Discar...

Просмотр краткой статистики

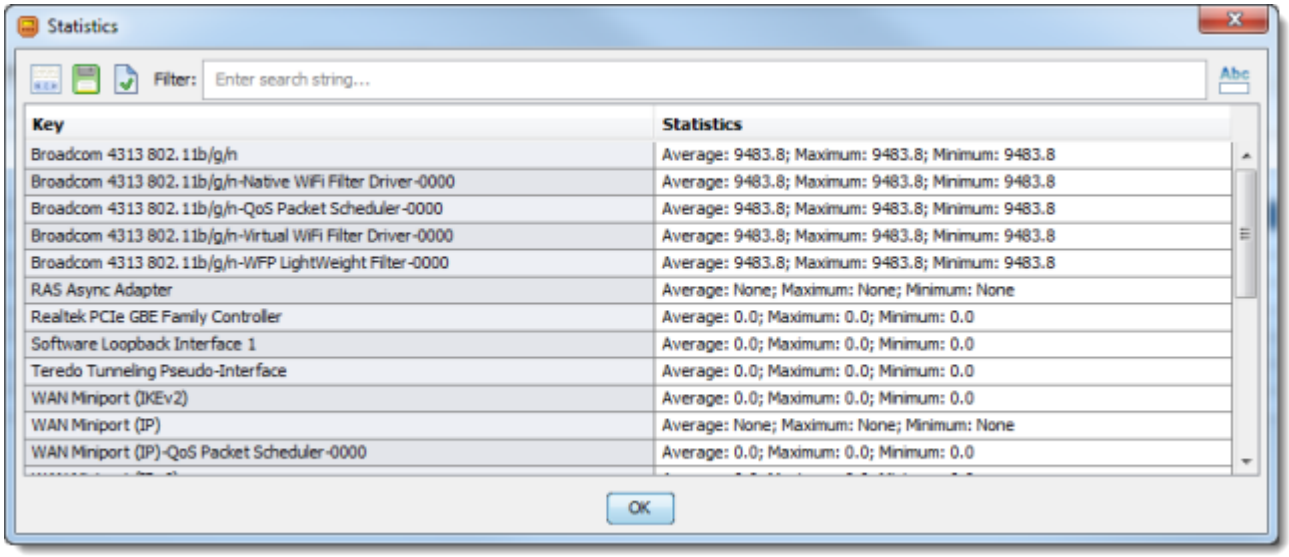
Для просмотра краткой статистики всех каналов, запустите Показать статус 111 (📈) и переключитесь на вкладку Статистика. Имейте в виду, что будут отображаться только те каналы, у которых включена опция **Показать в статистике.**

Вот так выглядит краткая статистика каналов в AtomMind Client:



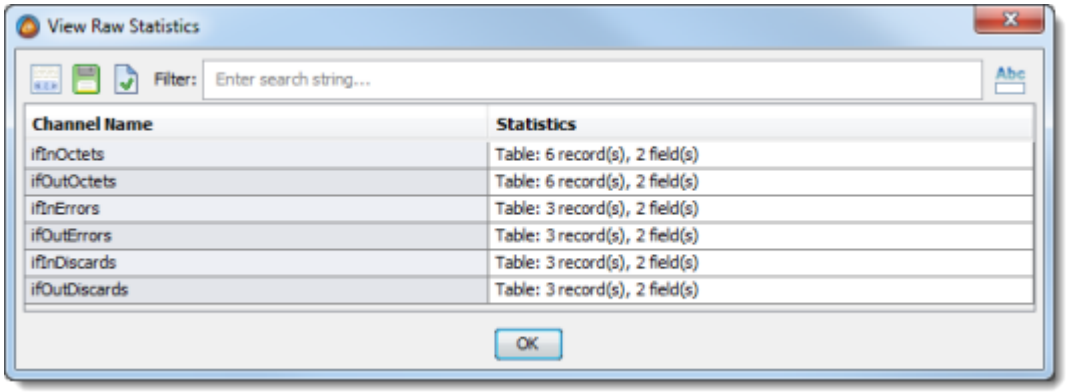
Period	Average	Maximum	Minimum
Last Minute	9483.803711	9483.803711	9483.803711
Last Hour	6575.411133	18293.226562	1612.363281
Last Day	10124.147461	55131.257812	5617.446777
Last Week	10124.147461	55131.257812	5617.446777
Last Month	10124.147461	55131.257812	5617.446777
Last Year	10124.147461	55131.257812	5617.446777

Если канал использует ключи 724, то сначала появляется диалоговое окно сводки набора данных, позволяющее выбрать набор данных (ключ) для просмотра его краткой статистики:

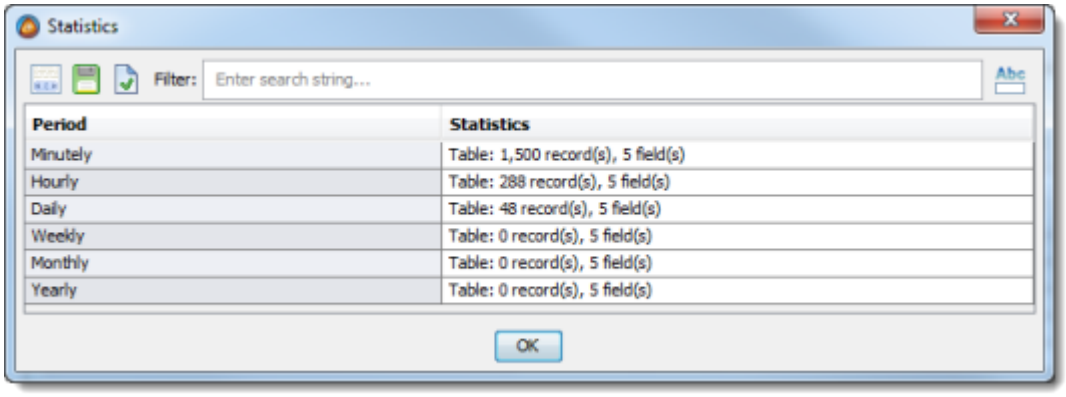


Просмотр подробной статистики

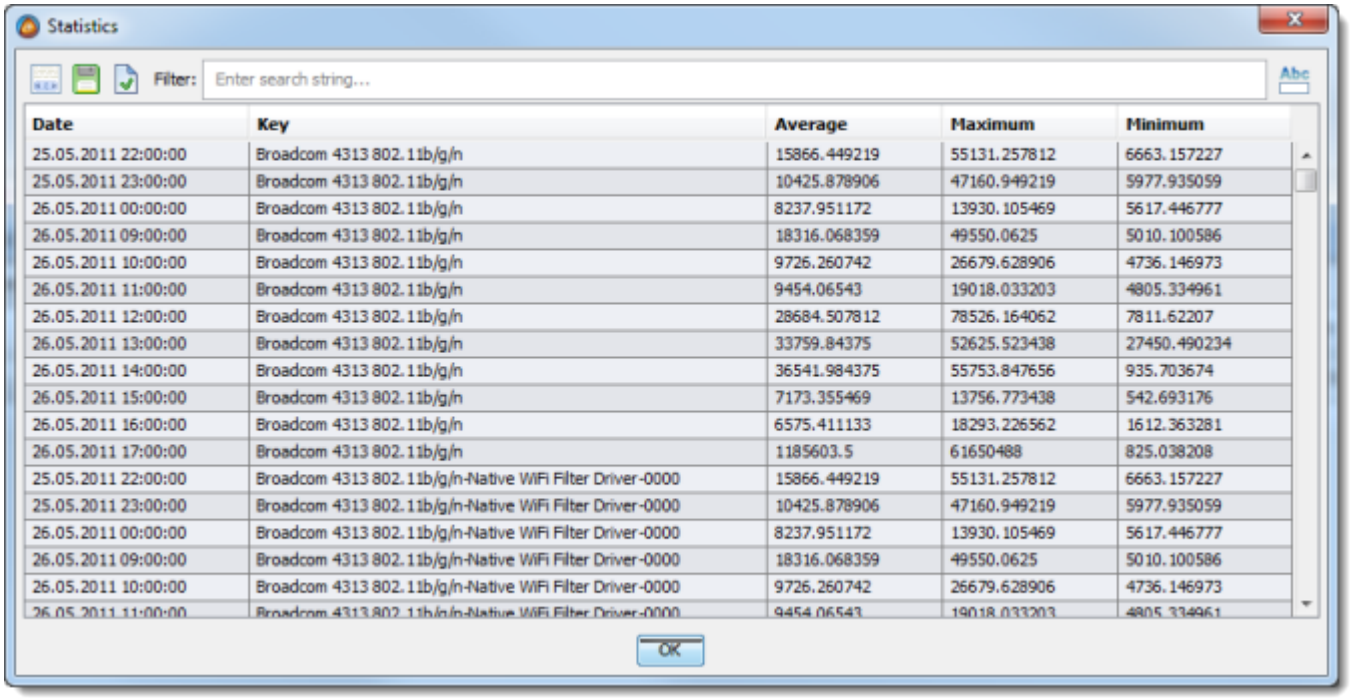
Для просмотра подробной статистики каналов запустите действие Конфигурировать устройство 1495 ( ), кликнув правой кнопкой на переменную (настройку) и выбрав **Показать Статистику** ( ). Сперва появится список каналов, соотнесенных с данной переменной:



Выберите канал, чтобы просмотреть список активных Архивов 718:



Наконец, выберите архив для просмотра его подробной статистики, сгруппированной по временным периодам:



Date	Key	Average	Maximum	Minimum
25.05.2011 22:00:00	Broadcom 4313 802.11b/g/n	15866.449219	55131.257812	6663.157227
25.05.2011 23:00:00	Broadcom 4313 802.11b/g/n	10425.878906	47160.949219	5977.935059
26.05.2011 00:00:00	Broadcom 4313 802.11b/g/n	8237.951172	13930.105469	5617.446777
26.05.2011 09:00:00	Broadcom 4313 802.11b/g/n	18316.068359	49550.0625	5010.100586
26.05.2011 10:00:00	Broadcom 4313 802.11b/g/n	9726.260742	26679.628906	4736.146973
26.05.2011 11:00:00	Broadcom 4313 802.11b/g/n	9454.06543	19018.033203	4805.334961
26.05.2011 12:00:00	Broadcom 4313 802.11b/g/n	28684.507812	78526.164062	7811.62207
26.05.2011 13:00:00	Broadcom 4313 802.11b/g/n	33759.84375	52625.523438	27450.490234
26.05.2011 14:00:00	Broadcom 4313 802.11b/g/n	36541.984375	55753.847656	935.703674
26.05.2011 15:00:00	Broadcom 4313 802.11b/g/n	7173.355469	13756.773438	542.693176
26.05.2011 16:00:00	Broadcom 4313 802.11b/g/n	6575.411133	18293.226562	1612.363281
26.05.2011 17:00:00	Broadcom 4313 802.11b/g/n	1185603.5	61650488	825.038208
25.05.2011 22:00:00	Broadcom 4313 802.11b/g/n-Native WiFi Filter Driver-0000	15866.449219	55131.257812	6663.157227
25.05.2011 23:00:00	Broadcom 4313 802.11b/g/n-Native WiFi Filter Driver-0000	10425.878906	47160.949219	5977.935059
26.05.2011 00:00:00	Broadcom 4313 802.11b/g/n-Native WiFi Filter Driver-0000	8237.951172	13930.105469	5617.446777
26.05.2011 09:00:00	Broadcom 4313 802.11b/g/n-Native WiFi Filter Driver-0000	18316.068359	49550.0625	5010.100586
26.05.2011 10:00:00	Broadcom 4313 802.11b/g/n-Native WiFi Filter Driver-0000	9726.260742	26679.628906	4736.146973
26.05.2011 11:00:00	Broadcom 4313 802.11b/g/n-Native WiFi Filter Driver-0000	9454.06543	19018.033203	4805.334961

По-другому посмотреть статистику настроек устройства можно через общую опцию Показать статистику 1562.

Общие каналы

Общая серверная конфигурация определяет список каналов статистики по умолчанию 205, применяемых для только что созданных устройств. Конфигурация каждого общего канала включает в себя имя переменной. Если новое устройство имеет переменную с таким именем, создается новый канал для хранения значения переменной для данного устройства. Свойства канала копируются из определения канала по умолчанию и могут быть точно настроены позже.

10.1.4.5 Мастер-значения настроек устройства

Мастер-значение - это значение переменной настроек устройства, общей для множества устройств.

Можно установить мастер-значения для всех настроек устройства. Мастер-значения можно хранить в любом месте дерева контекстов AtomMind Server (например, в переменной модели) вместо обычного кэша сервера.

Синхронизация 514 настроек, использующих мастер-значения, всегда осуществляется от AtomMind Server к устройству.

Если определенная настройка должна использовать мастер-значение, для нее необходимо настроить Выражение заданного значения 505.

10.1.5 Функции устройства (операции)

Функции устройства (также называемые *операционные функции устройства* или *операции устройства*) - это выполняемые функции 70 контекста устройства, соединенного с непосредственной аппаратной частью соответствующим драйвером устройства 518.

Как только системный модуль или внешнее приложение запрашивает выполнение функции устройства (операции), драйвер берет входную функцию, конвертирует ее в собственный формат устройства и отправляет в аппаратную часть с запросом выполнить операцию. Как только операция завершается, драйвер конвертирует ответ устройства в выходную функцию и возвращает этот выход вызывающей стороне.

Примеры операций устройства:

- Операция "открыть дверь" терминала контроля доступа, которая активирует дверную передачу.

- Операция "собрать мусор" приложения для предприятия Java, соединенная с AtomMind при помощи протокола JMX.
- Операция "запланировать самопроверку" терминала контроля транспорта. Эта операция может принимать параметр "время самопроверки" и возвращать отчет теста в виде таблицы, если время самопроверки было установлено на "сейчас".

10.1.6 События устройства (нотификации)

События устройства - это события 73 контекста устройства, сгенерированные соответствующим драйвером устройства 518, когда от непосредственного аппаратного устройства или источника данных приходят асинхронные сообщения.

Примеры событий устройства:

- Операция "чтение карты" терминала контроля доступа. Экземпляр события содержит идентификационный номер считываемой карты RFID.
- Событие контроллера "превышен порог давления", производящее проверку показателей давления относительно заданного значения.
- Событие "вскрыта дверь шкафа" большого промышленного устройства в виде контейнера, оборудованного собственными датчиками безопасности.

10.1.7 Общие свойства устройств

Существуют общие настройки для различных видов Device. Они доступны через действие редактировать свойств подключения 1494 контекста любого устройства.

Описание поля	Наименование поля
Имя Device. Имя контекста устройства 1494, необходимое для ссылки на данный Device из других частей системы. Оно должно соответствовать контексту соглашения о наименованиях 42.	name
Описание Device. Текстовое описание Device. Оно может указывать тип Device, расположение, предназначение и другие важные характеристики.	description
Показать полное описание Device . Активирование этой опции предоставляет полный просмотр описания Device.	showFullDeviceDescription
Тип Device. Тип устройства. В большинстве случаев тип определяется автоматически, но иногда может возникнуть необходимость определить его вручную для правильной обработки данных устройства. Вид **типового** устройства сообщает AtomMind Server, что нет необходимости в обработке данных устройства или коммуникативной деятельности пользователя.	type
Период синхронизации. Данная настройка определяет, как часто выполняется полная синхронизация 514 Device с сервером. Однако индивидуальные настройки Device могут иметь пользовательский период синхронизации, определенный Установкой свойств синхронизации 502 Device .	syncPeriod
Начать синхронизацию в период изменения настроек. Эта настройка определяет, производится синхронизация или нет, когда свойство устройства меняется.	startSyncOnSettingChange
Длина запроса синхронизации. Эта настройка определяет длину запроса синхронизации. Обычно использует асинхронные драйверы.	syncQueueLength
Прерывание синхронизации и переподключение при ошибке. Приводит к остановке процесса синхронизации в случае возникновения ошибки во время синхронизации какого-либо параметра. Это также приводит к принудительному отключению AtomMind Server от устройства и переподключению перед следующей синхронизацией. Включение данной опции может оказаться очень полезным, когда ошибка ввода/вывода или устройства во время синхронизации одного параметра, вероятно, может повториться во время синхронизации остальных параметров.	interruptOnError

Device не активировано. Приостановленные устройства никогда не синхронизируются с сервером.	suspend
Отключить чтение/запись настроек при синхронизации. В процессе синхронизации устройства синхронизация (чтение/запись) настроек производиться не будет. Обычно применяется, если настройки обновляются асинхронно.	disableSynchronousSettingValueRW
Включить расширенную статусную информацию. включает/отключает статус расширенной синхронизации устройства. 516	extendedStatus
Временная зона. Временная зона, в которой расположено Device. Может использоваться драйвером устройства для изменения отметок времени, например, во время синхронизации внутренних часов устройства с сервером.	timeZone
Режим чтения метаданных. Определяет, когда сервер должен читать определения настроек, операций, событий и активов устройства. Существует три режима: • **Не читать**. Все определения читаются всего один раз, после соединения с устройством. Этот режим подходит для улучшения производительности, если определения уже подсоединенных устройств никогда не меняются. • **Читать определения настроек, операций и событий**. Это режим по умолчанию, подходящий для большинства случаев. Сервер перечитывает все метаданные, кроме активов по каждому циклу синхронизации, отражая изменения, найденные в устройстве. Определения активов 501 не читаются. • **Читать все**. В этом режиме сервер полностью читает метаданные устройства, включая активы 501, по каждому циклу синхронизации. Этот режим подходит, если активы подсоединенного устройства могут измениться в любое время. Производительность может уменьшиться, если у устройства большое количество активов.	metadata
Сущности. Определяет, какие переменные, функции и события устройства будут доступны для этого устройства: • **Все сущности**. Все переменные/функции/события, которые обнаружены драйвером, будут доступны в контексте аккаунта устройства. • **Выбранные сущности**. Только сущности, проверенные в списке Переменных 1494 /Функций 1494/Событий 1494, могут быть добавлены в контекст аккаунта устройства. Некоторые драйверы устройства не поддерживают такой мелкоструктурный контроль над выбранными сущностями. Аккаунты устройства, использующие такие драйверы, создадут отчеты для пустых таблиц Переменных 1494 /Функций 1494/Событий 1494, даже если режим **Выбранные сущности** активирован. Все сущности будут подвергаться воздействию контекста аккаунта устройства в этом случае. Изменение активных сущностей на выбранные помогает в различных сценариях: • Когда устройство генерирует множество событий (даже в определенном выбранном массиве 501), и сервер должен только подписываться на выбранный тип события. • Когда определенный массив устройства содержит множество переменных и задержка их даже в редких случаях будет влиять на производительность устройства или AtomMind Server. • Когда некоторые неиспользуемые переменные устройства содержат большие блоки данных, которые потребляют чрезмерную оперативную память сервера и/или место хранения базы данных.	activeEntities
Режим кэша настроек. Определяет местоположение кэша настроек 502: • в базе данных 692 (режим по умолчанию, настройки доступны сразу после перезапуска сервера) • в памяти (настройки недоступны после перезапуска сервера вплоть до конца синхронизации 514 первого устройства)	cache
Качество настроек по умолчанию. Определяет качество будут иметь значения переменных настроек устройства, если драйвер не предоставляет качество значения.	settingsDefaultQuality
Период хранения события устройства. Определяет, как долго по умолчанию хранятся события устройства 510 в таблице данных 692 сервера. Нулевое значение	eventStoragePeriod

отключает постоянное хранение события для этого устройства. Настройки хранения для каждого события могут быть определены в глобальных Правилах обработки событий[196].	
Выражение зависимости устройства. Данное выражение[112] вычисляется перед каждой синхронизацией[514]. Если результат оценки **false**, синхронизация не проводится. Среда вычисления[112] выражения зависимости: Контекст по умолчанию[119]: Контекст данного устройства. Таблица данных по умолчанию[120]: Отсутствует. Строка по умолчанию[119]: 0 Переменные среды[123]: Только стандартные[123] переменные. Вполне распространенным является выражение зависимости устройства, относящееся к *онлайн статусу* какого-либо другого Device. Например, в системе управления сетью может оказаться полезным отключение синхронизации для группы устройств, подключенных к AtomMind Server через роутер, если сам роутер не подключен к сети (т.е. недоступен). В данном случае последующее выражение зависимости может быть использовано для каждого из этих устройств: `{users.admin.device.router:status$connectionStatus} == 0` Возможно определить числовое значение статуса подключения **Не в сети** (ноль в выражении выше), просмотрев значения выборки данных переменных для переменной status[1494] контекста Device.	dependency
Выражение статуса. Выражение пересчитывается в конце каждого цикла синхронизации[514]. Оно должно возвращать текстовое описание нынешнего статуса устройства. Статус может отображаться на картах устройства[1315], панелях инструментов[912] и т.д. Чтобы просмотреть полученный в результате статус, см. переменную Статус[1494] контекста Устройства. Среда вычисления[114] выражения статуса: Контекст по умолчанию[119]: Контекст текущего устройства. Таблица данных по умолчанию[120]: Отсутствует. Строка по умолчанию[119]: 0 Переменные среды[123]: Только стандартные[123] переменные.	status

Выражение цвета. Выражение пересчитывается в конце каждого цикла синхронизации 514. Оно должно возвращать результат типа Цвета. Этот цвет будет использоваться для цветового выделения устройства на картах устройства 1315, панелях инструментов 912 и т.д. Чтобы посмотреть итоговый цвет, см. переменную Статус 1494 контекста Устройства.	color

Среда вычисления 114 выражения цвета:

Контекст по умолчанию 119	Контекст текущего устройства.
Таблица данных по умолчанию 120	Отсутствует.
Строка по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

Выражение широты. См. Отслеживание местоположения устройства 516 для получения более подробной информации.	latitude

Среда вычисления 114 выражения широты:

Контекст по умолчанию 119	Контекст текущего устройства.
Таблица данных по умолчанию 120	Отсутствует.
Строка по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

Выражение долготы. См. Отслеживание местоположения устройства 516 для получения более подробной информации.	longitude

Среда вычисления 114 выражения долготы:

Контекст по умолчанию 119	Контекст текущего устройства.
Таблица данных по умолчанию 120	Отсутствует.

Строка по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.
Период хранения истории местоположения. Определяет, как долго хранить историю изменений широты/долготы.	locationStoragePeriod
Активировать тревогу при отключении соединения. Флажок, который контролирует, будет ли формироваться тревога отключения устройства от сети 781, если устройство находится не в сети некоторое время.	offlineAlert
Виртуальная сеть устройств. Задает виртуальную сеть устройств 517, которой принадлежит устройство.	virtualNetwork

Все это доступно для просмотра через переменную genericProperties 1497.

10.1.8 Синхронизация

Синхронизация Device представляет собой периодический процесс, позволяющий AtomMind Server создавать и поддерживать серверный "образ" устройства посредством чтения метаданных Device и чтения/записи настроек Device.

Синхронизация позволяет следующее:

- Разрешает встроенным модулям, операторам и сторонним программам просматривать доступные настройки, операции и события устройств;
- Обеспечивает быстрый доступ и управление значениями настроек Device. Все изменения конфигурации устройства сохраняются в серверном кэше 502 и записываются на устройство максимально лучшим образом.

Действующий алгоритм синхронизации зависит от типа драйвера 518 Device, но в большинстве случаев он включает в себя три основных шага:

- Установка соединения между AtomMind Server и Device;
- Чтение метаданных Device, т.е. информации о доступных настройках, операциях и событиях и создание подходящих переменных, функций и событий в Контексте 1494 Device для доступа к ним;
- Синхронизация значений настроек между Device и серверным кэшем 502.

Просмотр статуса синхронизации

Статус синхронизации устройства отображается иконками с устройствами:

- Новые добавленные устройства, которые ещё не синхронизировались или синхронизируются в данный момент, представлены иконками с вопросом: 
- Полностью синхронизированные устройства, завершившие синхронизацию без проблем, представлены иконками с галочкой: 
- Устройства с изменениями серверной конфигурации, ожидающие записи в аппартаную часть, представлены иконками с часами: 
- Устройства с ошибками синхронизации представлены иконками с восклицательным знаком: 

ПРОСМОТР ПРОЦЕССА СИНХРОНИЗАЦИИ

Если включена настройка Расширенный статус устройства 517 учетной записи устройства или текущая синхронизация была явно запрошена системным оператором, иконка статуса 516 устройства покажет прогресс текущей синхронизации:

- Этап соединения представлен иконками с молнией: 

- Этап чтения метаданных представлен иконками с лупой: 
- Этап синхронизации настроек значения представлен иконками со стрелками: 

Активируйте **Расширенный статус устройства,** если Вам сложно определить текущий статус Вашего устройства.

Виды синхронизации

Существует три вида синхронизации:

ПОЛНАЯ СИНХРОНИЗАЦИЯ

Это "стандартный" вид синхронизации, и он состоит из трех шагов:

- Подключение (только если оно еще не было установлено)
- Получение метаданных устройства
- Чтение/запись настроек устройства

Полная синхронизация выполняется один раз за Период синхронизации 510 устройства.

ЧАСТИТЧНАЯ СИНХРОНИЗАЦИЯ.

Частичная синхронизация выполняется посредством чтения/записи настроек устройства, для которых установлен Пользовательский режим синхронизации 504. Данный вид синхронизации довольно быстрый, т.к. он не включает в себя подключение (если оно уже установлено), сбор метаданных, чтение/запись **всех** значений настроек. Только одна настройка считывается или записывается на устройство.

Частичная синхронизация происходит каждый раз по истечению **Пользовательского периода синхронизации** для определенной настройки.

СИНХРОНИЗАЦИЯ ПРИ ПОДКЛЮЧЕНИИ

Синхронизация при подключении заставляет сервер устанавливать соединение с устройством и пройти аутентификацию/авторизацию. Обычно никакие данные не считываются и не записываются на устройство.

Такой вид синхронизации помогает серверу определить статус подключения 516 устройства.

Данный вид синхронизации выполняется только при запуске сервера, и только если время после последней полной синхронизации не превышает Период синхронизации 510.

10.1.9 Статус устройства

Статус устройства можно посмотреть, используя действие Показать статус 111 контекста устройства 1494. Сюда входит следующая информация:

- **Статус** устройства и используемый для статуса **Цвет** (рассчитываются согласно настройкам аккаунта устройства 510 **Выражение статуса** и **Выражение цвета**)
- Тип **Драйвера устройства**, используемого устройством
- **Время последней синхронизации**
- Текущий **Статус соединения**
- **Статус синхронизации** последней синхронизации 514 и/или прогресс текущей
- **Заполнение очереди синхронизации,** показывающее процент использования очереди запросов на синхронизацию
- **Операции чтения и асинхронного обновления переменных устройства,** показывающее число значений, читаемых с устройства или отправляемых устройством в асинхронном режиме.
- **Операции записи переменных устройства,** то есть число значений, записанных на устройство.
- **Операции вызова функций устройства,** то есть число выполнений операций устройства.
- **События устройства**, то есть число асинхронных оповещений, созданных устройством.

Статусы подключения и синхронизации

Комбинация статуса текущего подключения и статуса синхронизации представлена в Состоянии контекста устройства 1495 (иконка контекста).

СТАТУС СОЕДИНЕНИЯ

Статус соединения указывает, способен ли AtomMind Server подключиться к устройству. Статусы подключения:

- **Неизвестно.** Подключение не проверялось с момента создания учетной записи устройства или запуска сервера.
- **Онлайн.** Соединение между AtomMind Server и устройством установлено.
- **Офлайн.** Нет установленного соединения между AtomMind Server и устройством. Проверьте историю события 75 для обнаружения причины невозможности соединения.
- **Приостановлено**. Устройство приостановлено из-за ошибки проверки настроек сервера или зависимости 512. Попытки подключения не выполняются.

СТАТУС СИНХРОНИЗАЦИИ

Статус синхронизации показывает:

- Итоговый результат последней синхронизации.
- Прогресс текущей синхронизации, если включена опция **Расширенный статус** в Общих свойствах устройства 510.

Статусы синхронизации:

- **Синхронизировано**.
- **Ожидание синхронизации.**
- **Ошибка синхронизации.**
- **Не синхронизировано или синхронизация в процессе.**

Расширенные статусы синхронизации:

- **Подключение**. AtomMind Server пытается подключиться к устройству.
- **Чтение метаданных**. AtomMind Server считывает информацию о настройках, операциях и событиях устройства.
- **Синхронизация настроек**. AtomMind Server синхронизирует значения настроек между устройством и серверным кэшем 502.

Статус синхронизации настроек

Эта таблица выдает статус синхронизации по каждой переменной настроек устройства. Она включает следующие данные по каждой переменной:

- Дата/время последней синхронизации.
- Время ввода/вывода по последней синхронизации, т.е. время, потраченное драйвером устройства для считывания/записи значения настроек с аппаратной части.
- Обновленный флаг на сервере, который true, если значение настроек было обновлено в кэше сервера и новое значение еще не записано в аппаратной части.
- Текстовое описание текущего статуса синхронизации настроек.

10.1.10 Отслеживание местоположения устройства

Учетные записи устройства AtomMind Server предоставляют общий способ получить, сохранить и обработать географическое положение физических устройств. Информация о местоположении может динамически обновляться путем опроса устройств, либо определяется вручную в учетной записи устройства. Учетная запись может также отслеживать историю положений.

Действительные положения устройств и история положений может отражаться на картах и спутниковых изображениях. Например, виджет Карта 1010 предлагает необычайно легкий способ отслеживать устойства на Картах Google.

Расчет местоположения устройства

У каждой учетной записи устройства есть две специальные настройки:

- Выражение Широты
- Выражение Долготы

Эти выражения можно установить при редактировании общих свойств устройства 510. Каждое выражение перерассчитывается в конце каждого цикла синхронизации. Оно должно вернуть число с плавающей точкой, представляющее текущие широту/долготу устройства.

Обычно эти выражения рассчитывают координаты, совершая некоторые математические операции с настройками устройства, содержащими информацию о широте и долготе. Например, довольно легко конвертировать хранилище значений в градусах, минутах и секундах в значение с плавающей точкой, используя выражение.

Чтобы вручную определить положение устройства, напишите Выражение Широты и Выражение Долготы, которые возвращают константы с плавающей точкой, например `12.4459`.

Обзор местоположения устройства

Большинство компонентов AtomMind, такие как виджет Карта, получают информацию о местоположению автоматически, как только определяются правильные Выражения Широты и Долготы. Числовые значения текущей долготы и широты доступны через переменную Местоположение 1502 контекста Устройства.

10.1.11 Производительность устройства

Поскольку устройства - это ключевые компоненты AtomMind Server, они часто имеют решающее влияние на производительность всего сервера.

Производительность каждого отдельного устройства - это сложение следующих факторов:

- Производительность базового драйвера устройства 518. Может быть особенно важна в случае с драйверами устройств, записанными пользователем.
- Количество настроек устройства 501 и их пользовательский период синхронизации 504 (или общий период синхронизации 510 учетной записи устройства).
- Период хранения истории 503 настроек устройства и уровень вместе с производительностью базы данных сервера 692.
- Количество и сложность статистических каналов, основанных на переменных устройства 507.
- Частота операций устройства 509, вызываемых другими системными модулями и внешними приложениями.
- Количество событий, полученных от устройства и сохраненных в базе данных сервера, и сложность цепочек их обработки (такие как обработка правил модели 813, активированная этими событиями).

10.1.12 Сети виртуальных устройств

Сети виртуальных устройств используются для контроля нескольких параллельных сессий устройств ввода/вывода по группам устройств или для всех устройств, управляемых AtomMindom. Это полезно для тонко настраиваемого мониторинга:

- Больших сетей, где запуск сервера мониторинга может вызвать большой трафик
- Радио или подобных сетей, где группа устройств делит с сервером соединение с низкой пропускной способностью

Сети настраиваются в глобальной конфигурации сервера 205. Каждая сеть определяется:

- Уникальным **Именем**, идентифицирующим сеть в пределах сервера
- Понятным человеку **Описанием**
- **Обрабатывающей способностью**, т.е. количеством параллельных транзакций ввода/вывода, разрешенных для всех устройств, принадлежащих сети
- **Задержка транзакции**. Пауза после каждой транзакции ввода/вывода перед запуском новой для устройств сети

Когда виртуальная сеть настроена, к ней возможно добавить устройства. Чтобы добавить устройства к виртуальной сети, выберите их в диалоговом окне устройств Общие свойства 510.

10.2 Драйверы устройств

Драйвер устройства представляет собой особый вид плагина[207], который определяет, как AtomMind Server взаимодействует с определенным типом аппаратных устройств. Драйвер определяется во время создания учетной записи устройства[497].

AtomMind имеет встроенную поддержку множества коммуникационных протоколов. Драйверы устройств, входящие в комплект поставки AtomMind Server, позволяют подключить многочисленные устройства, производимые тысячами разных вендоров.

Однако иногда бывает необходимо подключить новое устройство, использующее проприетарный неподдерживаемый протокол. В этом случае, можно реализовать новый драйвер устройства программно[1345] или в режиме low code[551]. Другой способ - использовать преобразователь протокола на стороне сервера, называемый Агент AtomMind.

Драйверы устройств AtomMind реализуются на технологии Java и являются платформенно-независимыми. Драйверы представляют из себя плагины для сервера, поэтому их инсталляция достигается простым копированием файла с последующим перезапуском сервера.

ТВЭЛ регулярно создает новые драйверы устройства AtomMind Server для различных типов аппаратных устройств и протоколов. Чтобы увидеть новейший список доступных драйверов устройств, посетите сайт AtomMind.

Стандартные драйверы устройств

Таблица содержит список всех драйверов стандартных коммуникационных протоколов, которые доступны для платформы AtomMind и производных продуктов[1788]:

Протокол	Драйвер	Описание
Протокол AtomMind[2119]	Агент AtomMind[523]	Связь с Агентами AtomMind, реализованными на разных языках и платформах (Tibbo BASIC, C/C++, .NET, Java).
Asterisk	Asterisk[522]	Мониторинг и управление системой компьютерной телефонии Asterisk путем отправки команд CLI и обработки ответов.
BACnet	BACnet[532]	Поддержка BACnet IP и BACnet MS/TP. Чтение/запись свойств объектов. Доступ к сервисам устройств и обработка оповещений.
CoAP	CoAP[538]	Веб-протокол передачи данных для использования в ограниченных узлах и сетях Интернета вещей.
CORBA	CORBA[530]	Выполнение вызовов CORBA через IP сеть со спецификацией входных параметров и обработкой данных ответа.
CWMP	CWMP[540]	Управление и мониторинг абонентского оборудования (CPE) в соответствии со спецификацией TR-069.
DHCP	Network Host / DHCP[611]	Мониторинг работоспособности DHCP-сервера.
DLMS/COSEM	DLMS/COSEM[541]	Получение текущих показаний приборов учета и их истории.
DNP3	DNP3[543]	Полная поддержка для уровня приложений DNP3: чтение/запись, выбор и управление, прямое управление, управление событиями и т.д.
DNS	Network Host / DNS[600]	Валидация содержания зоны DNS. Мониторинг работоспособности DNS-сервера.
Ethernet/IP	Ethernet/IP[549]	Поддержка открытого промышленного протокола Ethernet, CIP.
FTP	Network Host / FTP[605]	Мониторинг атрибутов удаленных файлов. Мониторинг работоспособности FTP-сервера.
GPS/GLONASS Data	Satellite Vehicle Tracker[651] или Flexible драйвер[551]	Получение произвольных отчетов от любых спутниковых датчиков и других устройств M2M через TCP или UDP. Обработка команд на основе бизнес-правил. Поддержка для различных моделей датчиков «из коробки».
HTTP/HTTPS	HTTP/HTTPS[569]	Загрузка содержимого веб-страниц в ядро системы. Мониторинг работоспособности веб-сервера.

ICMP	Network Host / Ping 596	Мониторинг доступности (ping) и трассировка сетевых маршрутов (traceroute).
IEC 60870-5-104	IEC-104 545	Поддержка протокола МЭК 60870-5-104 в обоих режимах, slave и master.
IEC 60870-5-104	IEC-104 Server 547	Поддержка протокола МЭК 60870-5-104 в режиме сервера.
IMAP	Network Host / IMAP 602	Мониторинг работоспособности IMAP-сервера.
IPMI	IPMI 567	Мониторинг и управление серверами и сетевыми устройствами по IPMI.
JMS	WebSphere MQ 660	Мониторинг IBM WebSphere MQ.
JMX	Java Management Extensions 571	Выполнение операций MBean. Обработка оповещений MBean.
LDAP	Network Host / LDAP 611	Загрузка результатов запросов в ядро системы. Мониторинг работоспособности LDAP-сервера.
LON/LonTalk	OPC 617	Сеть устройств LON и серверов LNS можно настроить через сервер OPC и драйвер устройства OPC. Доступные мосты с LON на OPC включают IPLONGATE, Martikon OPC Server для Echelon LNS, Martikon OPC Server для Echelon LonManager, ConneXSoft CXS iLink DA Server для Echelon Smart Server, Gesytec Easylon OPC Server, Newron System NLOPC MIP и другие.
LON/LonTalk	Web Services 646	Echelon SmartServer и программное обеспечение можно настроить через SOAP (Web Service) API и драйвер устройства SOAP.
Meter-Bus	M-Bus 581	Получение значений точечного экспозамера и их истории.
Modbus	Modbus 583	Поддержка Modbus/RTU, Modbus/ASCII, Modbus/TCP и Modbus/UDP. Операции чтения/записи регистров.
GSM/GPRS Modem Control	Modem 587 or Flexible Driver 551	Отправка и получение SMS, управление модемом и получение данных посредством выполнения AT-команд.
MQTT	MQTT 590	Сетевой протокол для обмена сообщениями между устройствами, реализующий модель издатель-подписчик. Работает поверх TCP/IP.
NMEA 0183	NMEA 615 or Flexible Driver 551	Загрузка данных NMEA в ядро системы. Отслеживание местонахождения устройств.
ODBC	Database 647	Через стандартный мост JDBC-ODBC, см. SQL.
OPC DA	OLE for Process Control 617	Поддержка OPC DA 2.0 через DCOM. Работа под Windows, Linux и Mac OS.
OPC DA/HDA/AE	Агент AtomMind 523 драйвер + AtomMind OPC Agent 1980	Поддержка OPC DA, AE и HDA. OPC-агент AtomMind – это отдельное ПО для Windows, которое работает с серверами AtomMind под Windows, Linux и Mac OS.
OPC UA	OPC Unified Architecture 623	Полная поддержка стека OPC UA.
POP3	Network Host / POP3 601	Мониторинг работоспособности POP3-сервера.
Radius	Network Host / Radius 612	Мониторинг работоспособности Radius-сервера.
SIP	SIP 625	Тестирование звонков VoIP и отслеживание метрик звонка.
SMB/CIFS	SMB/CIFS 630	Получение доступа и мониторинг файлов и папок по технологии Microsoft Windows Network (SBM/CIFS).
SMI-S	SMI-S 633	Управление дисковыми хранилищами, поддерживающими протокол SMI-S. Мониторинг свойств объектов, выполнение запросов и методов объектов,

		обработка событий.
SMPP	SMPP 635	Отправка SMS сообщений через шлюз SMPP.
SMTP	Network Host / SMTP 603	Мониторинг работоспособности SMTP-сервера.
SNMP	SNMP 637	Поддержка SNMP v1, v2c и v3. Операции чтения/записи, получение и отправка ловушек (traps). Каталог и редактор MIB-файлов.
SOAP	Web Services 646	Выполнение произвольных запросов Web Service по протоколу SOAP путем определения вводимых данных и обработки данных вывода.
SQL	Database 647	Поддержка всех JDBC/ODBC-совместимых СУБД. Выполнение динамически сгенерированных запросов SELECT/UPDATE/INSERT/DELETE. Загрузка результатов запросов в ядро системы. Мониторинг статуса сервера баз данных.
SSH	Network Host / SSH 609	Выполнение скриптов и приложений на удаленных компьютерах. Мониторинг работоспособности SSH-сервера.
Telnet	Network Host / SSH 609	Выполнение скриптов и приложений на удаленных компьютерах. Мониторинг работоспособности Telnet-сервера.
VMware SOAP API	VMware 657	Получение статусов гипервизора/VM и счетчиков производительности.
WMI	WMI 661	Мониторинг свойств объектов, выполнение WQL запросов и методов объектов, обработка событий.
XMPP	XMPP 683	Расширяемый протокол обмена сообщениями и информацией о присутствии на базе XML.
	Application/Script 529	Выполнение настраиваемых приложений/скриптов по запросу или расписанию. Получение и обработка их выходных данных.
	Avatar 527	Создание локального «Аватара» драйверов любых удалённых устройств для упрощения разработки решений и улучшения производительности сети.
	File 576	Мониторинг локальных файлов, проверка контрольных сумм, загрузка содержимого файлов в ядро системы.
	Flexible 551	Выступает как комплект для самостоятельной разработки драйверов, позволяющий инженерам реализовать поддержку проприетарных протоколов без написания кода на Java.
	Folder 577	Мониторинг локальных папок, загрузка списка файлов в ядро системы.
	Graph Database 566	Хранение топологий в различных графовых базах, поддерживаемых Apache TinkerPop, включая Neo4j. Доступ к вычислительным операциям над графами через язык Gremlin.
	Message Stream 580 или Flexible Driver 551	Мониторинг входящих данных по последовательному порту или TCP/UDP соединению.
	Virtual Device 654	Имитатор устройства, предоставляет переменные различных типов, генераторы волнообразных сигналов, тестовые операции и события.
	Web Transaction 659	Всесторонний мониторинг веб-приложений за счет выполнения скриптов, моделирующих действия пользователя любой сложности, и анализа полученных результатов по показателям доступности, правильности и производительности.

Справочник по драйверам

Подпункты данного раздела посвящены различным драйверам устройств, которые привязаны к ядру AtomMind а также различным вертикальным решениям на рынке.

Описание каждого драйвера включает следующее:

Раздел	Примечания
Информация о драйвере	Детали драйвера, такие как ID плагина драйвера и т.д.

Основные настройки	Общая конфигурация 521 плагина драйвера.
Настройки уровня пользователя	Конфигурация уровня пользователя 521 плагина драйвера.
Свойства аккаунта устройства	Настройки уровня устройства 521 плагина драйвера. Включают в себя настройки подключения, настроенные вручную определения источников устройства и .т.д.
Активы устройства	Описание метода, используемого драйвером для определения активов устройства 501.
Настройки устройства	Описание метода, используемого драйвером для создания переменных относительно настроек аппаратного устройства.
Операции устройства	Описание метода, используемого драйвером для создания функций/действий относительно операций аппаратного устройства.
События устройства	Описание метода, используемого драйвером для создания описания событий, относящихся к типу событий аппаратного устройства. Данный раздел также показывает, как и когда экземпляры события получены от аппаратного устройства и конвертированы в события AtomMind.
Управление подключением	Детали о том, как и когда драйвер переключает устройство между статусами 516 Онлайн и Офлайн.
Детали синхронизации	Любая информация о процессе синхронизации 514 Устройства, обусловленная особенностями драйвера.

Уровни настроек

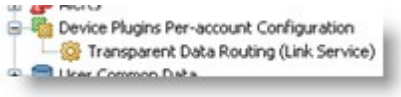
Каждый драйвер имеет до трех уровней настроек:

- **Основные настройки**. Влияют на поведение драйверов по умолчанию. Изменение основных настроек разрешено пользователем с необходимым уровнем доступа.
- **Настройки уровня пользователя**. Влияют на поведение драйвера, только если он относится к Устройству, которое принадлежит к определенной учетной записи пользователя 478. Когда драйвер взаимодействует с определенным Устройством, он использует *настройки уровня пользователя*, сохраненные в учетной записи пользователя Устройства .
- **Настройки устройства (свойства устройства)**. Данные настройки определяют, как драйвер устройства подключается к Устройству, взаимодействует с ним и обрабатывает данные Устройства.

Многие драйверы не используют все три уровня настроек.

Администрирование драйверов устройства

Два типа контекста используются для администрирования драйверов устройств. Первый представляет собой общий контекст конфигурации драйверов/плагинов 1505, который служит контейнером. Второй является контекстом конфигурация драйвера/плагина 1505, который хранит конфигурацию для одного драйвера.



Найти их можно в двух местах:

- Под корневым 1559 контекстом, где хранятся основные настройки.
- Под пользовательским 1608 контекстом, где хранятся настройки уровня пользователя.

Если определенный драйвер не содержит настройки какого-либо из данных уровней, соответствующий контекст конфигурации драйвера устройства не будет создан (т.е. не появится в Системном Дереве).

Настройки **Уровня устройства** для драйвера доступны через действие Редактировать свойства устройства 1494 в контексте Устройство.

10.2.1 Asterisk

Драйвер устройств 518 *Asterisk позволяет* AtomMind Server отслеживать состояние, производительность и работоспособность Asterisk VoIP сервера. Драйвер посылает команды Asterisk серверу и получает ответы, которые преобразуются в таблицы данных 49 AtomMind.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.asterisk

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

СВОЙСТВА ПОДКЛЮЧЕНИЯ

Настройки соединения определяют, как AtomMind Server взаимодействует с определенным сервером Asterisk. Данные настройки доступны через опцию изменить свойства аккаунта устройства 1494 Контекста Device. Доступны следующие свойства подключения:

Настройка	Описание
Адрес	IP адрес или имя хоста Asterisk сервера.
Порт	Порт, на котором работает Asterisk сервер.
Имя пользователя	Имя пользователя для авторизации.
Пароль	Пароль для авторизации.
Таймаут	Время ожидания выполнения операции Asterisk сервера. Как долго драйвер ожидает ответа сервера в миллисекундах.

Действия Asterisk

Это свойство содержит список команд, которые будут посылаться серверу Asterisk при каждом цикле синхронизации 514.

Имя	Тип	Описание
Команда отправить	Булевой	Включение/выключение отправки команды при синхронизации 514.
Действие Asterisk	Строка	Список команд доступных для отправки.
Пользовательская команда	Строка	Включен, когда выбрано "Пользовательская команда" в поле "Действие Asterisk". Тело комманды, которое будет отправлено серверу Asterisk.
Имя переменной	Строка	Включен, когда выбрано "Пользовательская команда" в поле "Действие Asterisk". Доступно имя результата "Пользовательской команды".

Список доступных команд для отправки:

Тип действия	Описание действия
AgentsAction	AgentsAction запрашивает состояние всех агентов.
AgiAction	Добавляет новую команду AGI, которая должна быть асинхронно выполнена AGI приложением. Это действие добавляет приложение в очередь определённого канала. Если канал находится вне асинхронного AGI, приложение вернет ошибку.

CoreShowChannelsAction	CoreShowChannelsAction запрашивает состояние всех активных каналов. Данное действие похоже на StatusAction, но с более детальной информацией о канале.
DbGetAction	Извлекает запись из базы данных Asterisk для данного ключа. Если запись найдена, сервером Asterisk отсылается DBGetResponseEvent, содержащий значение, иначе отсылается ManagerError, что означает, что ни одна запись не найдена.
IaxPeerListAction	Возвращает список всех определенных IAX пиров.
ParkedCallsAction	ParkedCallsAction запрашивает список всех текущих вызовов на удержании.
QueueStatusAction	QueueStatusAction запрашивает состояние всех определенных очередей, их агентов(членов очередей) и вызывающих абонентов.
QueueSummaryAction	QueueSummaryAction возвращает статистику для одной или всех очередей.
ShowDialplanAction	Возвращает список свойств, определенных в плане набора(dialplan).
SipPeersAction	Возвращает список всех определенных SIP пиров.
SipShowRegistryAction	Возвращает список деталей о SIP регистрациях.
StatusAction	StatusAction запрашивает состояние всех активных каналов. Вы также можете (в том числе Asterisk 1.6) передать имя канала, чтобы получить статус одного определенного канала.
VoicemailUsersListAction	Возвращает список всех пользователей голосовой почты.
ZapShowChannelsAction	ZapShowChannelsAction запрашивает список всех zap каналов.
Custom Command	CommandAction посылает команды интерфейса, работающего в режиме командной строки (CLI) серверу Asterisk. Для вывода списка поддерживаемых команд введите "help" в коммандной строке Asterisk. В ответ на CommandAction Вы получите CommandResponse, который содержит CLI вывод.

Активы Device

Драйвер не поддерживает активы.

Настройки Device

Этот драйвер создает отдельную переменную настройки Device для каждого действия, добавленного в таблицу дейстий Asterisk. Эта переменная обеспечивает результат действия.

Операции Device

Драйвер не проводит операции.

События Device

Драйвер не представляет события.

Детали синхронизации

Драйвер Asterisk подключается к Asterisk серверу, посылает действия, обеспечивает конфигурацию, обрабатывает ответы и выходит из системы сервера Asterisk.

10.2.2 Agent

Драйвер устройства AtomMind Agent является драйвером устройства 518, позволяющим AtomMind Server взаимодействовать с Agent. Технически Agent представляет собой BASIC приложение, которое создает и поддерживает дерево контекстов 41 при помощи узла для каждого подключенного к нему физического Device (элемента оборудования). В действительности Device также может быть реализовано с помощью

пользовательского кода, являющегося частью Agent. Переменные 61 контекста каждого Device относятся к настройкам Device. Каждый такой контекст обладает функциями 70, позволяющими выполнять некоторые операции с Device, и событиями 61, использующимися для управления состоянием Device и изменениями в режиме работы. AtomMind Server использует протокол передачи данных AtomMind 2119 для взаимодействия с Agent.

Этот драйвер отличается от драйвера устройства Локальный Агент 574. Последний устанавливает исходящее соединение в "режиме агента" вместо обслуживания входящих соединений агента.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.agent

Общие настройки

- **Количество портов для** Agent. Входящие подключения от агентов будут приниматься на данный порт.
- **Количество портов для безопасных соединений** Agent. Входящие безопасные соединения от Агентов будут приниматься на этот порт.
- **Таймаут команды**. Время ожидания команд, посланных при помощи протокола передачи данных Агентов.
- **Таймаут команды вещания**. Время ожидания команд, отправленных через широковещательную передачу во время обнаружения Агентов.
- **Автоматическая регистрация аккаунтов устройств для новых Агентов.** Позволяет Агентам передавать AtomMind Server пароли, что приводит к автоматической регистрации сервером учетных записей в случае попыток подключения новых Агентов.
- **Максимальная длина очереди событий**. Максимальное количество единичных событий Агента, которые должны быть сохранены, если обрабатываемое событие отложено ввиду некоторых причин. Если очередь событий переполняется, удаленный агент не сможет посылать события.

Настройки уровня пользователя

Не определены.

Свойства Device

- **Пароль**. Пароль Агента.

Активы Device

Наличие активов 501 определяется конкретной реализацией Agent.

Настройки Device

Драйвер устройства < %AGENT%> предоставляет удаленные переменные Agent, такие как локальные настройки Device.

Операции Device

Драйвер устройства Agent предоставляет удаленные функции Agent, такие как локальные операции Device.

События Device

Драйвер устройства Agent предоставляет удаленные события Agent, такие как локальные события Device.

Подключение

Данный драйвер приводит устройство в режим **Онлайн**, если:

- Агент подключен к серверу и произвел вход в систему
- Основные операции агента выполнены успешно.

СТАТИСТИКА СОЕДИНЕНИЙ АГЕНТА

Диалог статуса устройства 515 для драйвера устройства Agent показывает дополнительную таблицу Статистики соединения, которая показывает:

- **Период соединения**
- **Количество команд**, то есть количество команд, отправленных Агенту
- **Среднее время ответа** на все команды
- **Входящий трафик** (несжатый)
- **Исходящий трафик** (несжатый)
- **Количество неотвеченных команд**

МЕТРИКА ПРОИЗВОДИТЕЛЬНОСТИ

Свойство	Описание
Длина очереди события	Показывает, сколько событий на рассмотрения на отправку в Очереди событий.
Отклоненные события	Показывает, сколько событий было отклонено из-за переполненной Очереди событий.
Отключения	Показывает количество отключений.
Событий получено	Показывает общее количество полученных событий.
Обновлений получено	Показывает количество полученных событий обновления.

Детали синхронизации

Драйвер Агента выполняет синхронизацию 514 AtomMind Server и Device, проводимую Агентом по следующей схеме:

1. Считывает с Agent список доступных контекстов и их отношения "родитель-потомок".
2. Создает серверный контекст для каждого контекста Agent. Данный контекст позволяет пользователям AtomMind взаимодействовать с Агентом напрямую или через кэш переменных 502.
3. Проводит синхронизацию внутренних часов реального времени Agent со временем на сервере. Время устанавливается согласно временной зоне, назначенной для Device.
4. Считывают определения переменных, функций и событий с каждого контекста, доступного в Agent. Однако он не считывает *значения*, сохраненные в Таблицах Данных, а только их формат. Более подробную информацию об определении переменных смотрите здесь 61.
5. Находит общие таблицы 509, которые могут быть использованы в качестве "заданных значений" переменных Agent (таких как настройки Device).
6. Создает кэш 502 свойств (если Device подключается впервые) или обновляет его согласно определениям переменных, считанных ранее. Переменные, относящиеся к группе 61 **по умолчанию**, синхронизируются с сервером и могут быть изменены пользователями. Кэш свойств создается только в серверных контекстах, относящихся к контекстам Device в Agent. Переменные внутренних настроек Agent модифицируются сервером напрямую, без создания кэша.
7. Устанавливает опцию *Конфигурировать* 1495 Device в каждом контексте, относящемся к контексту Device в Agent.
8. Создает необходимое действие 87 для каждой функции контекста Device внутри группы **по умолчанию.** Данное действие позволяет пользователям вызывать функцию Agent, уточнять, указывать и просматривать ее параметры.
9. Устанавливает действие *События Мониторинга* в каждом контексте, относящемся к контексту Device в Agent, чтобы показать все события, поступающие от Agent и принадлежащие к группе **по умолчанию**.
10. Синхронизирует значения переменных между кэшем сервера и Agent для каждого контекста, соответствующего контексту Device в Agent.

Точная структура коммуникаций между Agent и AtomMind Server во время синхронизации описана в Коммуникация между 690 Agent и AtomMind Server 690.

Дополнительные действия контекста Device Агента

Контекст устройств 1492, предоставляемый Агентом, может выполнять различные дополнительные действия 87. Большинство из них определяется пользовательским BASIC приложением Агента. Например, если Агент работает с

некоторыми датчиками терминалов, подключенных к последовательным портам, возможно выполнение действия **Обнаружения устройств**, которое заставляет заново сканировать последовательные порты, находить подключенные устройства и обновлять список контекстов 1494 Device.

Существуют два дополнительных действия контекста Device, предоставляемых Агентом:

КОНФИГУРАЦИЯ АГЕНТА

Данное действие конфигурации 105 используется для конфигурации самого Агента, т.е. для установки параметров, определяющих, как Агент взаимодействует с подключенными к нему аппаратными устройствами, как он преобразовывает протокол данных устройств в Протокол передачи данных AtomMind 2119, и какие переменные, функции и события предоставляются агентом в контексты <%dt% 1494, относящиеся к данным устройствам.

ПЕРЕЗАГРУЗКА АГЕНТА

Данное действие перезагружает программируемый модуль, запускающий BASIC приложение агента. Технически это действие вызова функции 103, которое вызывает функцию **перезагрузки** из корневого контекста Агента.

Обнаружение Агента

Обнаружение Агента представляет собой процесс размещения всех Агентов аппаратного обеспечения в *сегменте локальной сети* и автоматического создания у*четных записей обнаруженных Агентов.*

Сегмент локальной сети означает, что между ПК и всеми остальными устройствами данного сегмента установлены только сетевые концентраторы (нет роутеров, мостов, брандмауэров и т.п.)

Аппаратные Агенты, расположенные "позади" роутера (т.е. в случае, когда для обнаружения Агента пакеты, отправленные из AtomMind Server, должны пройти через роутер) не могут быть обнаружены автоматически AtomMind Server, потому что широковещательные дэйтаграммы протокола UDP, используемые для их обнаружения в сети, не трансилруются роутером. Однако иногда все-таки возможно подключить такие Агенты к AtomMind Server, если они будут доступны через их IP адреса (иными словами, если имеется сетевой маршрут).

Каждый обнаруженный Агент идентифицируется MAC-адресом. Обнаружение определит все местные Агенты, даже если некоторые из них имеют одинаковый IP адрес или же неправильный или недоступный IP адрес. Для доступа AtomMind Server к Агентам в режиме обнаружения не требуется наличие корректного IP адреса.

Во время процесса обнаружения AtomMind Server создает новый контекст **Обнаруженного Агента** для каждого определяемого Агента. Все контексты обнаруженных Агентов, созданные во время предыдущего обнаружения, удаляются и создаются новые. Невозможно создать Учетную запись обнаруженного Агента вручную, это осуществляется только в процессе обнаружения.

Если автоматическое обнаружение Агентов не включено и не было запущено вручную, контекст обнаружения не будет содержать учетные записи Агентов.

Обнаружение может быть начато при выполнении одного их двух действий **Обнаружения:**

- Обнаружение Агентов
- Обнаружение Агентов и Автоподключение

Подключение внешних Агентов к AtomMind Server

Подключение внешних Агентов к AtomMind Server представляет собой процесс реконфигурации аппаратного Агента для немедленного его подключения к AtomMind Server после запуска. Существует несколько способов:

1. Подключение ранее обнаруженного 2096 Агента, используя его учетную запись 2112.
2. Автоматическое подключение всех обнаруженных Агентов к AtomMind Server без взаимодействия с пользователем.
3. Конфигурация вручную обнаруженных Агентов, подключенных неверно в результате выполнения выше перечисленных функций.
4. Подключение вручную Агента, который не был определен в процессе обнаружения, но может быть доступен через AtomMind Server.

ПОДКЛЮЧЕНИЕ РАНЕЕ ОБНАРУЖЕННОГО АГЕНТА К ATOMMIND SERVER

Данная процедура запускается действием **Подключить Агент к AtomMind Server** контекста Агента.

Пользователь должен определить 90 параметры до начала процесса подключения:

- **Пароль.** Необходим, если в настройках Агента оборудования установлен пароль. Данный пароль будет использован AtomMind Server для доступа к настройкам Агента аппаратных средств.

- **Имя пользователя.** Имя пользователя 478 AtomMind Server, владеющего учетной записью Агента, которое будет использоваться для аутентификации Агента оборудования при входе в AtomMind Server.
- **Имя устройства**. Имя вышеуказанной Учетной записи 497 Device. Настройка **Имя устройства** Агента оборудования будет выставлена согласно данному значению.
- **Принудительное подключение, даже если выполнена конфигурация Агента для AtomMind Server или имеется несовместимое программное обеспечение.** По умолчанию подключение не удается и появляется сообщение об ошибке в том случае, если Агент уже настроен для подключения к AtomMind Server или версия программного обеспечения устарела (т.е. не поддерживает автоматическую конфигурацию для AtomMind Server). Данная опция отключает проверку значений настроек Агента и его номера версии. Принудительное подключение пройдет успешно, но может быть, что Агент не сможет в действительности подключиться к AtomMind Server или осуществить вход. Если Агент не вошел в AtomMind Server по истечению нескольких секунд после завершения принудительного подключения, вручную измените параметры настроек Агента (в зависимости от вида устройства это можно сделать при помощи клавиатуры/жидкокристаллического дисплея или веб интерфейса).

АВТОМАТИЧЕСКОЕ ПОДКЛЮЧЕНИЕ ОБНАРУЖЕННОГО АГЕНТА К ATOMMIND SERVER

Если обнаружение было запущено, используя действие **Обнаружение агента и Автоподключение**, AtomMind Server автоматически выполняет процедуру подключения обнаруженного Агента к AtomMind Server 526 для каждого Агента, определенного во время процесса обнаружения 526. Это может быть полезным для быстрого подключения нескольких Агентов, встроенных в сегмент локальной сети.

Автоподключение не выполняется, если:

- Агент запрашивает пароль для доступа к настройкам, либо
- Агент уже настроен на подключение к AtomMind Server при запуске, либо
- Программное обеспечение Агента не поддерживает автоподключение.

10.2.3 Аватар

Драйвер устройства *Аватар* создает точную копию другого устройства, которая синхронизируется 514 с исходным устройством. Аватары чаще всего используются для создания локальных копий аккаунтов удаленных устройств, подключенных в локальному серверу через распределенную архитектуру 1332.

Как и любые другие устройства, аватары хранят отдельный моментальный снимок устройства 502 и собирают историю переменных/событий.

Аватары отличаются от Агентов 523 и Локальных агентов 574.

Управление устройствами Аватар

Драйвер Аватар сопровождается моделью 810 **Управление устройствами Аватар,** которая позволяет автоматически создавать, обновлять и удалять аватары для набора удаленных устройств, подключенных к AtomMind Server через распределенную архитектуру 1332. Эту модель можно создать, как любой другой ресурс 208.

Настойка модуля **Управление устройствами Аватар** (доступна через действие **Редактировать дополнительные свойства** контекста модели):

ОБЩИЕ НАСТРОЙКИ

Общие настройки модуля:

- **Модуль включен.** Контролирует работу модуля в целом.
- **Автоматическое удаление аватара.** Активирует автоматическое уничтожение аватаров после удаления их исходных устройств.
- **Форсировать синхронизацию.** Опция активирует единовременное создание аватаров для выбранных исходных устройств. Опция отключается автоматически после завершения процесса.

НАСТРОЙКА ИСХОДНЫХ СЕРВЕРОВ

Настройка удаленных серверов, для которых будут создаваться аватары:

- **Активный.** Настройка включает/отключает запись конфигурации
- **Имя распределенного соединения.** Имя Поставщика распределенной архитектуры 1332

- **Имя точки монтирования.** Имя точки монтирования вышеупомянутого Поставщика
- **Пользовательский префикс.** Префикс, который будет добавляться к именам удаленных устройств, чтобы не допустить путаницы с именами локальных устройств
- **Локальный пользователь.** Аккаунт пользователя 478, под которым будут созданы аватары
- **Тип устройства.** Список типов устройств, для которых будут созданы аватары

информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.avatar

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

свойства аккауна устройства

- **Исходное устройство**. Путь контекста устройства 1494, для которого создается аватар.
- **Ресурсные группы**. Группы объектов удаленного устройства, которые будут реплицированы аватаром. Таблица групп включает столбец **Активный,** который определяет, включена ли репликация определенной ресурсной группы, а также столбец **Имя группы,** определяющий имя реплицируемой группы. По умолчанию реплицируются объекты группы `remote`.

Активы Device

Активы 501 аватара реплицируются из исходного устройства.

Настройки Device

Настройки устройства аватара реплицируются из исходного устройства.

Операции Device

Функции устройства аватара реплицируются из исходного устройства.

События Device

События устройства аватара реплицируются из исходного устройства.

Подключение

Драйвер приводит устройство в режим **Онлайн**, если доступен контекст **Исходного устройства**.

Статус подключения аватара не привязан к статусу подключения самого исходного устройства.

Детали синхронизации

Драйвер Аватар синхронизирует 514 аватар с **Исходным устройством,** следуя алгоритму:

1. Чтение информации об активах, переменных, функциях и событиях Исходного устройства, доступных в Ресурсных группах
2. Чтение значений переменных Исходного устройства
3. Подписка аватара на избранные обновления записей переменных и события Исходного устройства

10.2.4 Внешнее приложение/Скрипт

Драйвер устройства 518 Внешнее приложение/скрипт позволяет AtomMind Servery выполнять любые приложения или скрипты, находящиеся на машине, где запущен сервер. Приложения могут выполняться по запросу или периодически. Их коды выхода, результаты выполнения и поток вывода ошибок доступны для анализа.

При вызове приложения или скрипта по запросу операторы могут также указать пользовательские аргументы входа для него.

Информация по драйверу

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.application

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

ПЕРЕМЕННЫЕ (ПЕРИОДИЧЕСКОЕ ВЫПОЛНЕНИЕ)

Это табличная настройка, определяющая, какие приложения должны выполняться периодически. Каждое приложение представлено переменной настройки устройства 501, а поля содержат код выхода, результат выполнения и ошибки, обнаруженные во время последнего выполнения.

Поле	Описание
Имя	Имя переменной настройки устройства, представляющей код выхода приложения и результат выполнения.
Описание	Описание этой переменной.
Команда	Путь для выполнения приложения или скрипта.
Аргументы	Список аргументов для передачи данной переменной/скрипту.
Таймаут	Максимальное время ожидания завершения выполнения приложения.
Кодировка	Кодировка, применяемая для обработки результата выполнения приложения и его ошибок.

ФУНКЦИИ (ВЫПОЛНЕНИЕ ПО ЗАПРОСУ)

Это табличная настройка, определяющая, какие приложения должны выполняться по запросу операторов. Каждое приложение представлено функцией работы устройства 509. Эти функции принимают аргументы командной строки приложения/скрипта как их вход. Коды выхода и результаты выполнения приложения/скрипта возвращаются этими функциями.

Поле	Описание
Имя	Имя функции настройки устройства, вызывающей приложение или скрипт.
Описание	Описание этой функции.
Команда	Путь для выполнения приложения или скрипта.
Таймаут	Максимальное время ожидания завершения выполнения приложения.
Кодировка	Кодировка, применяемая для обработки результата выполнения приложения и его ошибок.

Активы Device

Драйвер не поддерживает активы.

Настройки Device

Внешний драйвер приложения/скрипта устройства создает одну переменную настройки устройства для каждой записи в таблице **Переменные (Периодическое выполнение)**. У этой переменной есть следующие поля:

- **Код выхода**. Числовой код, возвращенный приложением.
- **Результат выполнения**. Результат выполнения приложения.
- **Ошибки**. Поток вывода ошибок приложения.

Операции Device

Внешний драйвер приложения/скрипта устройства создает одну функцию работы устройства для каждой записи в таблице **Функции (Выполнение по запросу)**. Эта функция принимает список аргументов командной строки для передачи приложению. У нее есть следующие поля:

- **Код выхода**. Числовой код, возвращенный приложением.
- **Результат выполнения**. Результат выполнения приложения.
- **Ошибки**. Поток вывода ошибок приложения.

События Device

Нет событий, представленных драйвером.

Подключение

Устройство приложения/скрипта всегда **Онлайн**.

Синхронизация

Во время синхронизации 514 драйвер выполняет все приложения и скрипты, определенные в таблице **Переменные (Периодическое выполнение)**.

10.2.5 CORBA

Драйвер устройства 518 *CORBA* позволяет AtomMind Server работать с любым приложением на базе CORBA. CORBA - это стандарт, определенный Object Management Group (OMG) для облегчения коммуникации систем, развернутых на различных платформах.

Драйвер имеет следующие возможности:

- Компиляция IDL файлов;
- Выполнение методов, описанных в IDL файлах.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.corba

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

СВОЙСТВА ПОДКЛЮЧЕНИЯ CORBA

Настройки соединения определяют как AtomMind Server взаимодействует с определенным приложением CORBA. Данные настройки доступны через опцию изменить свойства 1494 Device контекста Device. Доступны следующие свойства подключения:

Настройка	Описание
Тип соединения	Определяет тип соединения с удаленнным приложением на основе CORBA. Сейчас доступны три вида соединения: corbaloc, corbaname и IOR File.
Corbaloc	Это поле становится доступным, когда выбран тип соединения "Corbaloc".

	Синтаксис: corbaloc:<iiop>:<host>:<port>/<object key>. Параметр <iiop> может быть не указан. **Пример:** corbaloc::127.0.0.1:52486/NameService
Corbaname	Это поле становится доступным, когда выбран тип соединения "Corbaname". Синтаксис: corbaname::<host>:<port>/<object key>#<object name>. **Пример:** corbaname::127.0.0.1:52486/NameService#Server
IOR File	Это поле становится доступным, когда выбран тип соединения "IOR File". IOR является преобразованной в строку ссылкой, которая индетифицирует объект на удаленном сервере CORBA. IOR файл может быть предоставлен разработчиком приложения CORBA.

IDL ФАЙЛЫ

Это табличная настройка, определяющая, какие IDL файлы будут использоваться при работе с объектом CORBA. IDL является языком спецификаций, используемым для описания интерфейса компонентов ПО. IDL файлы описывают интерфейс независимым от языка способом, осуществляя коммуникацию между компонентами ПО, которые не могут общаться на одном языке – например, между компонентами, написанными на языке C++ и компонентами, написанными на языке Java.

Поле	Описание
IDL File	IDL файл описывает методы удаленных объектов и типы данных.

Некоторые IDL файлы могут зависеть друг от друга. В этом случае, если в таблице IDL некоторые из зависимых файлов не указаны, произойдет ошибка.

Активы Device

Драйвер не поддерживает активы.

Настройки Device

Драйвер не предоставляет настроек.

Операции Device

Количество и функциональность операций зависит от IDL файлов, определенных в настройках.

В процессе соединения драйвер проверяет и составляет набор CORBA IDL файлов.

В результате, будет создан набор функций Device, соответствующий методам удаленного объекта. Эти методы определяют функциональность объектов CORBA. Они позволяют пользователям отправлять запросы серверу.

Любой вызов функции Device связан с вызовом метода CORBA. AtomMind Server использует таблицу входных параметров функции для построения объектов CORBA. Итоговые CORBA объекты конвертируются в таблицу выходных параметров.

События Device

Драйвер не представляет события.

Подключение

Драйвер не устанавливает/не тестирует подключения. Любое CORBA устройство всегда находится **онлайн**, несмотря на возможные ошибки, которые могут произойти из-за неполадок сети.

Детали синхронизации

Данный драйвер бездействует во время периодической синхронизации.

10.2.6 BACnet

Драйвер устройства 518 BACnet позволяет AtomMind Servery взаимодействовать с устройствами, поддерживающими **BACnet/IP протокол.** BACnet широко используется для автоматизации зданий и систем контроля отопления, вентиляции, кондиционирования, освещения, доступа и систем пожарообнаружения и сигнализации.

Драйвер BACnet обеспечивает подключаемость оборудования, используя BACnet протокол поверх IP сетей (часто соотносимый с *BACnet/IP* или *Annex J*).

ВИДЫ ПОДДЕРЖИВАЕМЫХ ОБЪЕКТОВ

Приведенный ниже список содержит все виды BACnet объектов, которые могут быть прочитаны, записаны или обработаны агентом через BACnet драйвер.

- Analog Input
- Analog Output
- Analog Value
- Binary Input
- Binary Output
- Binary Value
- Calendar
- Command
- Device
- Event Enrollment
- File
- Group
- Loop
- Multi State Input
- Multi State Output
- Notification Class
- Program
- Schedule
- Averaging
- Multi State Value
- Trend log
- Life Safety Point
- Life Safety Zone
- Accumulator
- Pulse Converter
- Event Log
- Trend Log Multiple
- Load Control
- Structured View
- Access Door

ПОДДЕРЖИВАЕМЫЕ СЛУЖБЫ

Приведенный ниже список включает все службы BACnet, доступные при использовании драйвера.

Служба BACnet	**BACnet Interoperability Building Block (BIBB)**
Who-Is	DM-DDB-A
ReadProperty	DS-RP-A
ReadPropertyMultiple	DS-RPM-A
WriteProperty	DS-WP-A
SubscribeCOV	DS-COV-A
SubscribeCOVProperty	DS-COVP-A

BACnet Interoperability Building Block (BIBB) описывает службы, поддерживаемые BACnet устройством или приложением (*Annex K* спецификации BACnet).

ПОДДЕРЖКА СЕГМЕНТА

Драйвер устройства 518 BACnet обеспечивает сегментированные запросы и ответы, которые, в свою очередь, поддерживают размер окон между 1 и 127 байтами.

Информация о драйвере

ID плагина 518 com.tibbo.linkserver.plugin.device.bacnet

драйвера :

Общие настройки

Для входа в сеть BACnet AtomMind Server эмулирует BACnet устройство, которое взаимодействует с другими устройствами в сети. Данное устройство называется **Локальным Устройством**, выполняющим функцию драйвера.

Общие настройки используются для установки данного Локального Устройства.

Свойство	Описание
ID устройства	ID устройства используется для идентификации Локального устройства BACnet, оно может быть задано в диапазоне от 0 до 4194304. Только одно устройство может использовать данное ID в сети.
Широковещательный адрес	Маска широковещательного адреса для запроса "*Who-Is*".
Время ожидания подключения	Время (в миллисекундах), в течение которого драйвер ждет ответа "*I -Am*" после отправки запроса "*Who-Is*".
Порт	Определяет локальный UDP порт, к которому будет привязан драйвер для выполнения всех взаимодействий по каналу. Настройка по умолчанию - 47808 (0xBAC0). Обычно все BACnet/IP устройства сети Ethernet используют один и тот же порт.
Время ожидания	Данный параметр определяет время ожидания драйвером ответа от устройства до повторной отправки или продолжения запроса. Действующий диапазон - от 100 до 9999 миллисекунд.
Время ожидания сегмента	Определяет то же время ожидания, что описано выше, но только для сегментированных запросов.
Окно сегмента	Данный параметр определяет количество сегментов сообщения, которые могут бы посланы до возвращения сегмента сообщения о подтверждении принимающей стороной. Отправитель предлагает размер окна, а получатель уточняет действительный размер (который будет не больше предлагаемого размера). Таким образом, драйвер использует эту настройку в качестве предлагаемого размера окна для запросов и действительного размера окна для ответов от устройства. Большие размеры, скорее всего, увеличат быстродействие надежной сети, но, с другой стороны, меньшие размеры позволят выявить проблемы взаимодействия на раннем этапе и исправить их при возврате меньших сегментов. Действующий диапазон - от 1 до 127.
Повторное выполнение	Определяет количество повторных попыток драйвера получить подтверждение запроса перед прекращением.
Максимально допустимая APDU длина	Данный параметр определяет общую длину или количество байт в сегментах сообщений, принятых драйвером. Драйвер попытается прочитать максимальную APDU длину, назначенную конечным устройством при запуске, и будет использовать наименьшие из локальных или удаленных лимитов при отправке ответов. Настройки меньшей величины могут быть необходимы для приспособления ограничений оборудования между драйвером и конечным устройством. Варианты включают 50, 128, 206 (для формата LonTalk), 480 (для формата ARCNET), 1024 и 1476 (для формата ISO 8803-3). Наибольшее значение 1476 обычно является оптимальным.
Максимальное число свойств при составном сегментированном запросе	Ограничивает количество объектов, входящих в запросы Read Property Multiple. Фактическое число объектов, включенных в запрос, может варьироваться в зависимости от того, сколько объектов необходимо для чтения и записи в определенное время. Обычно чем выше значение, тем лучше производительность. Для больших запросов или ответов, однако, производительность может быть снижена сегментацией сообщения. К сожалению, нет установленных правил для определения оптимальной настройки. Для отладки какого-либо устройства пользователям необходимо экспериментировать с настройками. Устройства, не поддерживающие службы Read Property Multiple или Write Property Multiple, должны быть настроены на 1.
Максимальное число свойств при составном несегментированном запросе	То же, что описано выше, но для устройств, не поддерживающих передачу сегментации.

Настройки уровня пользователя

Не определены.

Свойства Device

НАСТРОЙКИ ПОДКЛЮЧЕНИЯ

Настройки подключения определяют взаимодействие AtomMind Server с определенным устройством BACnet. Доступны следующие свойства подключения:

Свойство	Описание
Метод адресации	Существует 3 способа адресации к устройству BACnet: • **По адресу и порту.** В данном случае номер экземпляра устройства определяется отправкой запроса "*Who-Is*" на заданный адрес/порт. • **По номеру экземпляра устройства.** В данном случае IP адрес и порт устройства с установленным номером экземпляра определяются отправкой широковещательного запроса "*Who-Is*" для диапазона IP, назначаемого широковещательным адресом общей настройки. • **Комбинированный.** Может быть использован для взаимодействия с устройствами, не поддерживающими идентификацию "*Who-Is*". В данном случае, адрес/порт и номер экземпляра нужно уточнить.
IP адрес и имя хоста	Адрес устройства BACnet, к которому необходимо подключиться.
Порт	IP порт устройства, к которому необходимо подключиться.
Номер экземпляра	Идентификатор удаленного устройства.

Активы устройства

Драйвер создает один корневой актив для каждого вида объекта BACnet, поддерживаемого устройством. Дочерние элементы каждого актива соотносятся с отдельными экземплярами данного объекта, обнаруженными в устройстве.

Настройки Device

Драйвер устройства 518 BACnet создает одну переменную настройки устройства на каждое обнаруженное свойство каждого объекта устройства BACnet. Переменные подразделяются не несколько групп. Свойства *Объекта Устройства* относятся к группе "Основных свойства". Все остальные настройки разделены на группы по *типу объекта*: Analog Input, Analog Output и т.д.

В каждой группе "Тип Объекта" создается новая подгруппа для каждого *Экземпляра объекта*, если он обладает свойствами соответствующего типа.

Операции Device

Контекст драйвера устройства 518 BACnet выполняет следующие функции и действия:

ПОДПИСКА НА УВЕДОМЛЕНИЯ ОБ ИЗМЕНЕНИЯХ ЗНАЧЕНИЙ (COV)

Данная функция позволяет создать/отменить подписку на уведомления об Изменениях Значений от объекта (COV). Механизм COV позволяет драйверу получать уведомления об изменениях свойств удаленных объектов BACnet.

Если объект предоставляет COV отчет, при изменении определенных свойств объекта подписанным клиентам отправляются COV уведомления.

После успешной подписки драйвер устройства 518 будет асинхронно обновлять кэш настроек устройства 502 при получении COV уведомления.

Существует определенные риски в использовании подписки на COV. Это проблемы, свойственные системам, основанным на событиях, например, неожиданные падения ссылок, потеря подписки при перезапуске устройства и т.д. Объекты критических данных должны периодически опрашиваться совместно с получением их обновлений через COV уведомления.

Формат параметров **ввода** функции имеет следующие поля:

Имя	Тип	Описание
subscribe	Boolean	Флажок, показывающий, следует ли выполнить подписку/отказ от подписки.

subscriberProcessIdentifier	Long	Цифровая "основа", необходимая для идентификации подписки внутри драйвера. Не используйте один и тот же идентификатор процесса для разных подписок. Этот же идентификатор следует использовать для отмены подписки на COV уведомления.
monitoredObjectIdentifier	DataTable	Хранит идентификатор объекта в принимающем устройстве, для которого требуется подписка. Здесь выделяются два поля: • **objectType** • **instanceNumber**
issueConfirmedNotifications	Boolean	Определяет, необходимо ли устройству выполнить *ConfirmedCOVNotifications или UnconfirmedCOVNotifications* при изменениях.
lifetime	Long	Требуемое время действия подписки. Значение, равное нулю, означает бесконечное время действия, не предусматривающее автоматическую отмену. Значение, не равное нулю, означает время действия до автоматической отмены подписки.

Функция не имеет параметров **выхода**.

Для более подробной информации о COV подписках смотрите "*Службы тревог и событий",* раздел *SubscribeCOV Service* спецификации BACnet.

СВОЙСТВА COV ПОДПИСКИ

Данная функция подобна описанной выше, но она позволяет подписаться на уведомления об изменениях свойств определенного объекта.

Формат параметров **ввода** функции имеет следующие поля:

Имя	**Тип**	**Описание**
subscribe	Boolean	Флажок, показывающий, следует ли выполнить подписку/отказ от подписки.
subscriberProcessIdentifier	Long	Цифровая "основа", необходимая для идентификации подписки в пределах драйвера. Не используйте один и тот же идентификатор процесса для разных подписок. Этот же идентификатор следует использовать для отмены подписки на COV уведомления.
monitoredObjectIdentifier	DataTable	Хранит идентификатор объекта в принимающем устройстве, для которого требуется подписка. Здесь выделяются два поля: • **objectType** • **instanceNumber**
issueConfirmedNotifications	Boolean	Определяет, необходимо ли устройству выполнить *ConfirmedCOVNotifications* или *UnconfirmedCOVNotifications* при изменениях.
lifetime	Long	Требуемое время действия подписки. Значение, равное нулю, означает бесконечное время действия, не предусматривающее автоматическую отмену. Значение, не равное нулю, означает время действия до автоматической отмены подписки.
monitoredPropertyIdentifier	Integer	Определяет свойство, изменение которого должно быть передано.

covIncrement	Float	Минимальное изменение контролируемого свойства, приводящее к запуску *COVNotification* через BACnet устройство. Данный параметр игнорируется, если тип BACnet контролируемого свойства не является *REAL*.

Функция не имеет параметров **выхода**.

СВОЙСТВО ЧТЕНИЯ

Данная функция используется для запроса значения одного свойства определенного объекта BACnet.

Функция имеет следующие параметры **ввода**:

Имя	Тип	Описание
objectIdentifier	DataTable	Определяет объект, свойство которого должно быть прочитано и возвращено в ответе. Выделяют два поля: • **objectType** • **instanceNumber**
propertyIdentifier	Integer	Определяет свойство, которое должно быть прочитано и возвращено данной службой.
propertyArrayIndex	Long	Если рассматриваемое свойство представляет собой массив, данный дополнительный параметр определяет индекс массива. Если он опущен (т.е. NULL), будет прочитан весь массив.

Вывод функции представляет собой значение рассматриваемого свойства. Его формат зависит от типа свойства.

Вы можете использовать функции *Read Property* и *Read Property Multiple*, чтобы прочитать **специализированные свойства**. Просто установите параметр propertyIdentifier на желаемый индекс свойства.

СВОЙСТВО ЧТЕНИЯ МНОЖЕСТВА

Данная функция используется для чтения значения одного или нескольких свойств одного или нескольких *объектов* BACnet.

Формат параметров **ввода** функции (может иметь несколько записей) содержит следующие поля:

Имя	Тип	Описание
objectIdentifier	DataTable	Определяет объект, свойства которого должны быть прочитаны и возвращены в ответе. Выделяют два поля: • **objectType** • **instanceNumber**
propertyReferences	DataTable	Список свойств, читаемых с объекта. Определяется двумя полями: • **propertyIdentifier** • **propertyArrayIndex** Могут быть использованы специальные идентификаторы свойств, такие как *ALL*, *REQUIRED* или *OPTIONAL.*

Формат параметров **вывода** функции:

Имя	Тип	Описание

objectIdentifier	DataTable	Идентификатор объекта, определяемый с помощью: • **objectType** • **instanceNumber**
results	DataTable	Читает результат свойств вышеуказанного объекта в виде таблицы с несколькими полями: • **error** (Boolean) • **propertyIdentifier** (Integer) • **propertyArrayIndex** (Integer) • **readResult** (Data Table) Поле **readResult** содержит значение свойства или текст ошибки, в случае ее возникновения.

СВОЙСТВА ЗАПИСИ

Функция используется для изменения значения одного свойства объекта BACnet.

Формат параметров **ввода** функции:

Имя	Тип	Описание
objectIdentifier	DataTable	Определяет объект, свойства которого должны прочитываться и возвращаться в ответе. Выделяют два поля: • **objectType** • **instanceNumber**
propertyIdentifier	Integer	Определяет свойство, которое должно быть прочитано и возвращено данной службой.
propertyArrayIndex	Long	Если рассматриваемое свойство представляет собой массив, данный дополнительный параметр определяет индекс массива.
valueType	String	Тип значения BACnet. Данный параметр необходим только в интерактивном режиме вызова функции. Он используется для установления формата параметра, описанного далее.
value	DataTable	Новое значение свойства.
priority	Integer	Целое число в диапазоне 1-16, которое указывает приоритет данной операции записи.  Для более подробной информации о свойствах смотрите "*Процедуры BACnet*" в разделе "*Установление приоритетов команд*" спецификации BACnet.

Функция не имеет параметров **выхода.**

События Device

Драйвер не представляет события.

Подключение

Драйвер приводит устройство в режим **Онлайн**, если:

- Локальное устройство BACnet создано и инициализировано успешно.
- Удаленное устройство BACnet самоопределяется во время обмена *"Who-is"/"I-am"* (если настройка подключения устройства **Метод Адресации** не установлена на **Комбинированный**).

- Удалось прочитать следующие свойства удаленного *объекта устройства: Максимально допустимая APDU длина, Сегментация поддерживается, Идентификатор поставщика (*если настройка подключения устройства **Метод Адресации** не установлена на **Комбинированный**).

Синхронизация

Синхронизация между AtomMind Server и Device BACnet происходит в несколько шагов:

- Чтение поддерживаемых удаленным устройством служб и создание необходимых операций устройства.
- Распознавание всех объектов, представленных в удаленном устройстве, посредством чтения свойства "*Список объектов" объекта устройства* удаленной единицы BACnet.
- Обнаружение свойств всех найденных объектов.

Драйвер BACnet сочетает два способа обнаружения свойств объекта:

- запрос свойств от самого *объекта*;
- чтение каждого свойства, определяемого в спецификации BACnet для данного *типа объекта.*

Первый способ позволяет обнаружить расширенные свойства объекта при наличии таковых, в то время как второй способ обнаруживает только стандартные свойства BACnet.

Собственные свойства могут быть обнаружены во время синхронизации, если устройство поддерживает специальный идентификатор свойств ALL в службе *ReadPropertyMultiple.*

В настройках всех устройств имеется доступ на чтение/запись, однако, само устройство может запретить доступ на их запись. В случае создания переменной, защищенной от записи, устройство выдаст ошибку *Доступ на запись запрещен*.

10.2.7 CoAP

Драйвер устройства 518 Constrained Application Protocol обеспечивает AtomMind Server моделью взаимодействия между конечными точками приложения по принципу запрос/ответ, поддерживает встроенное обнаружение сервисов и ресурсов, а также включает ключевые понятия Web, такие как URI и типы интернет-медиа. Этот драйвер очень похож на HTTP и используется для обмена данными между машинами (M2M).

Основные особенности драйвера CoAP:

- Web-протокол, используемый в M2M с ограниченными требованиями
- Обмен асинхронными сообщениями
- Низкие накладные расходы и очень простой синтаксический разбор
- Поддержка URI и типов содержимого
- Возможность прокси и кэширования

информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.coap

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

СВОЙСТВА ПОДКЛЮЧЕНИЯ

Настройки соединения определяют как AtomMind Server взаимодействует с сервером CoAP. Доступны следующие свойства подключения:

Свойство	Описание

Адрес	IP адрес или имя хоста CoAP сервера.
Протокол	CoAP или безопасный CoAP (CoAPS).
Порт	Порт CoAP сервера. ПО умолчанию: 5683.
URL	Путь к ресурсу расположенному на CoAP сервере.
Тип запроса	Поддерживается следующие методы CoAP запроса: **GET, POST, PUT** и **DELETE**.
Данные для выполнения запроса POST	Данные, которые будут представлены с запросом POST в виде строки.
Использовать клиентские сертификаты	При использовании протокола CoAPS, активирует применение клиентского сертификата при аутентификации на сервере.
Путь к хранилищу сертификатов	Путь к файлу хранилища ключей, содержащий сертификаты клиента. Это путь локальный файловой системы на машине с AtomMind Server.
Тип хранилища сертификатов	Тип файла хранилища сертификатов: JKS или PKCS12.
Пароль к хранилищу сертификатов	Пароль, дешифрующий файл хранилища ключей (опционально).
Alias	Идентификатор сертификата ключа
Использовать доверенный сертификат	При использовании протокола CoAPS, активирует применение доверенного сертификата.
Путь к хранилищу доверенных сертификатов	Путь к файлу хранилища ключей, содержащий доверенные сертификаты. Это путь локальный файловой системы на машине с AtomMind Server.
Тип хранилища сертификатов	Тип файла хранилища доверенных сертификатов: JKS или PKCS12.
Пароль к хранилищу доверенных сертификатов	Пароль, дешифрующий файл хранилища доверенных ключей (опционально).
Alias	Идентификатор доверенного сертификата ключа
Таймаут	Время ожидания выполнения операции CoAP сервера

активы Device

Драйвер не поддерживает активы.

настройки Device

Драйвер устройства CoAP создает одну переменную Device. Переменная включает следующие поля:

Свойство	**Описание**
Успешно	Указывает на успешные соединения драйвера CoAP.
Время ответа, миллисекунд	Время ответа сервера.
Код ответа	Код ответа CoAP.
Ответ	Текст ответа CoAP.
Ошибка	Сообщение об ошибке, или NULL, если запрос выполнен успешно.

операции Device

ВЫПОЛНИТЬ COAP ЗАПРОС

Эта операция отправляет необработанный CoAP запрос GET, DELETE, PUT, или POST на устройство и возвращает выходные данные. Входные данные включают URL, метод запросов, данные для выполнения запроса POST и дополнительные CoAP заголовки. Возвращаемые значения - флаг об успешном выполнении, код ответа CoAP, заголовки, текст вернувшейся страницы и сообщение об ошибке.

события Device

Драйвер не представляет события.

подключение

Этот драйвер приводит устройство в статус **онлайн,** если соединение было успешно установлено.

Детали синхронизации

CoAP монитор подключается к IP хосту, отправляет CoAP запрос, созданный с использованием предоставленной конфигурации (порт, URL, тип и данные запроса, CoAP заголовки, агент, таймаут), и анализирует ответ.

Если в настройках включена аутентификация, и требуется Аутентификация базового доступа для доступа к веб-странице, драйвер также отправляет запрос на авторизацию.

10.2.8 CWMP

Драйвер устройства CPE WAN Management Protocol (CWMP) используется для централизованной настройки и управления оборудованием на территории клиента, которое соответствует спецификации TR-069. В этом случае, сервер AtomMind выступает как сервер автоматического конфигурирования.

Информация о драйвере

ID плагина драйвера: com.tibbo.linkserver.plugin.device.cwmp

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

СВОЙСТВА ПОДКЛЮЧЕНИЯ

Настройки соединения определяют как AtomMind Server взаимодействует с определенным клиентским оборудованием. Данные настройки доступны через опцию Изменить свойства Device контекста Device. Доступны следующие свойства подключения:

Свойство	Описание
Идентификатор клиентского оборудования	Идентификатор клиентского оборудования для идентифицирования при входящем соединении с AtomMind Server.
Адрес установки соединения	Имя хоста или IP адрес для установления HTTP соединения с оборудованием клиента.
Порт установки соединения	Номер порта для установления HTTP соединения с оборудованием клиента.
Путь в URL установки соединения	HTTP URL путь для установки HTTP соединения с оборудованием клиента.

активы Device

Драйвер не поддерживает активы.

настройки Device

Драйвер устройства CWMP создает одну переменную настройки Device для каждого обнаруженного параметра клиентского оборудования.

операции Device

Драйвер не представляет операции

события Device

Драйвер не представляет события.

подключение

Этот драйвер приводит устройство в статус **онлайн,** если исходящее HTTP соединение с клиентским оборудованием было успешно.

Детали синхронизации

Синхронизация между AtomMind Server и устройством CWMP включает следующие шаги:

- Получение метаданных о доступных свойствах оборудования клиента
- Чтение значений свойств оборудования

10.2.9 DLMS/COSEM

Драйвер устройства 518 DLMS/COSEM позволяет AtomMind Server общаться с умными счётчиками, поддерживающими протокол DLMS/COSEM. Считывание показаний счетчиков доступны как переменные настроек устройств 501.

Драйвер DLMS/COSEM поддерживает как последовательную (Serial), так и IP (TCP) связь со счетчиками.

КОНФИГУРАЦИЯ ATOMMIND SERVER ДЛЯ ПОСЛЕДОВАТЕЛЬНОЙ ПЕРЕДАЧИ ДАННЫХ

Обратитесь к разделу Включение последовательного порта 1448, если у вас проблемы с подключением к последовательным устройствам DLMS/COSEM.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.dlms-cosem

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

НАСТРОЙКИ ПОДКЛЮЧЕНИЯ

Настройки подключения определяют, как AtomMind Server взаимодействует с определенным устройством DLMS/COSEM. Данные настройки доступны через опцию изменить свойства 1494 Device контекста Device. Доступны следующие свойства подключения:

Property	Description
Режим	Выбор DLMS/COSEM TCP, DLMS/COSEM UDP и DLMS/COSEM Serial.
IP адрес или имя хоста	Адрес устройства DLMS/COSEM (для DLMS/COSEM TCP/UDP).

Port	Порт устройства DLMS/COSEM (номер IP порта, заданный по умолчанию на 4059 для DLMS TCP/UDP; имя серийного порта для DLMS Serial).
Скорость передачи информации (Baud Rate)	Скорость передачи информации (для DLMS/COSEM Serial).
Контроль входящего потока	Тип контроля входящего потока: None, CTS/RTS, или XON/XOFF (для DLMS/COSEM Serial).
Контроль исходящего потока	Тип контроля исходящего потока: None, CTS/RTS, или XON/XOFF (для DLMS/COSEM Serial).
Биты данных (Data Bits)	Биты последовательных данных (для DLMS/COSEM Serial).
Стоп-биты (Stop Bits)	Стоп-биты последовательных данных (для DLMS/COSEM Serial).
Четность (Parity)	Четность (для DLMS/COSEM Serial).
Таймаут	Время ожидания выполнения команды (30 секунд по умолчанию).
Производитель	Производитель.
Механизм аутентификации	Авторизация (открытый доступ, доступ по паролю, доступ с аутентификацией) и ввод пароля если нужен.
Пароль	Пароль для аутентификации.
Обращение к объекту через его логическое имя	Обратиться к объекту COSEM и считать/записать определенный атрибут или выполнить определенный метод, можно двумя способами, либо через его логическое имя (LN referencing), либо через короткое имя (SN referencing).
Адрес клиента	Идентификатор клиента.
Логический адрес	Логический адрес.
Физический адрес	Физический адрес.

Активы Device

Драйвер создает один корневой актив для каждого вида объекта DLMS/COSEM, поддерживаемого устройством. Дочерние элементы каждого актива соотносятся с отдельными экземплярами данного объекта, обнаруженными в устройстве.

Настройки Device

Драйвер не предоставляет настроек.

Операции Device

Драйвер не проводит операции.

События Device

Драйвер не представляет события.

Подключение

Драйвер переводит устройство в режим **онлайн** если:

- Последовательный порт был успешно открыт (для устройств DLMS/COSEM Serial)
- TCP подключение к устройству было успешно установлено (для устройств DLMS/COSEM TCP)

Синхронизация

Синхронизация между AtomMind Server и устройством DLMS/COSEM включает в себя следующие шаги:

- Чтение метаданных доступных показаний счетчика.
- Чтение полученных значений показаний счетчика M-Bus и хранение этих значений в кэше настроек.

10.2.10 DNP3

Драйвер устройства DNP3 осуществляет получение данных через распределённый сетевой протокол вер. 3 (DNP3).

Информация о драйвере

ID плагина драйвера: com.tibbo.linkserver.plugin.device.dnp3

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

НАСТРОЙКИ СОЕДИНЕНИЯ

Свойство	Описание
Тип соединения	Тип соединения: TCP или Serial.
Задержка повторной попытки	Период между повторами команд.
Адрес	IP адрес или имя хоста устройства (для TCP соединения).
Порт	Последовательный порт или TCP порт устройства.
Скорость передачи информации (Baud Rate)	Скорость передачи информации последовательного соединения.
Биты данных (Data Bits)	Биты данных последовательного соединения.
Стоп-биты (Stop Bits)	Стоп-биты последовательного соединения.
Четность (Parity)	Четность последовательного соединения.
Контроль потока	Контроль потока последовательного соединения.
Размер очереди записи VTO	Настройка конфигурации мастер-уровня.
Использовать нестандартную VTO функцию	Настройка конфигурации мастер-уровня.
Позволить синхронизацию времени	Настройка конфигурации мастер-уровня.
Включить незатребованные сообщения	Настройка конфигурации мастер-уровня.
Включить незатребованные сообщения класса 0	Настройка конфигурации мастер-уровня.
Включить незатребованные сообщения класса 1	Настройка конфигурации мастер-уровня.
Включить незатребованные сообщения класса 2	Настройка конфигурации мастер-уровня.
Включить незатребованные сообщения класса 3	Настройка конфигурации мастер-уровня.

Интенсивность опроса целостности	Настройка конфигурации мастер-уровня.
Процент повторного выполнения задач	Настройка конфигурации мастер-уровня.
Таймаут ответа	Настройка конфигурации уровня приложений.
Количество повторных попыток	Настройка конфигурации уровня приложений.
Максимальный анализируемый размер фрагмента	Настройка конфигурации уровня приложений.
Использовать подтверждения канального уровня	Настройка конфигурации канального уровня.
Количество повторных попыток	Настройка конфигурации канального уровня.
Локальный адрес	Настройка конфигурации канального уровня.
Удаленный адрес	Настройка конфигурации канального уровня.
Таймаут ответа для подтвержденных запросов	Настройка конфигурации канального уровня.

Активы Device

Драйвер не поддерживает активы.

Настройки Device

ИЗМЕРЕНИЯ

Таблица измерений содержит список всех значений, полученных от DNP3 устройства. Каждое измерение определено:

- Индексом
- Значением
- Типом
- Датой/временем
- Качеством

Операции Device

- Отослать команду Control Relay Output Block
- Отослать команду 16-bit Integer Analog Output
- Отослать команду 32-bit Integer Analog Output
- Отослать команду 32-bit Float Analog Output
- Отослать команду 64-bit Double Analog Output

События Device

Драйвер не представляет события.

Подключение

Драйвер переходит в режим **Онлайн**, если подключение к DNP3 устройству было успешно установлено.

Синхронизация

Синхронизация между AtomMind Server и DNP3 устройством включает в себя чтение таблицы измерений.

10.2.11 IEC-104

Драйвер устройства 518 IEC-104 позволяет AtomMind Server взаимодействовать с устройствами, поддерживающими **протокол IEC 60870-5-104**. Эти устройства могут быть соединены с системой и, как происходит со всеми другими типами устройств, их данные преобразуются в объединенную форму для того, чтобы они могли быть доступны из разных объектов AtomMind. См. статью Devices для получения подробной информации о "нормализованном" представлении устройства в AtomMind.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.iec-104

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

СВОЙСТВА СОЕДИНЕНИЯ

Настройки соединения определяют, как AtomMind Server взаимодействует с определенным устройством IEC-104. Данные настройки доступны через опцию изменить свойства 1494 Device контекста Device. Доступны следующие свойства соединения:

Свойство	Описание
IP-адрес или имя хоста	Адрес устройства IEC-104.
Порт	Порт устройства.
Таймаут соединения (t0)	Максимальное время ожидания (в милисекундах) сообщения STARDT CON после отправки сообщения STARTDT ACT. По стандарту этот таймаут называется t0.
Общий адрес ASDU	Таблица с адресами целевой станции или широковещательный адрес. Если длина поля общего адреса 1 байт, то адреса от 1 до 254 используются как адрес отдельной станции (адрес станции), а 255 используется для широковещательной адресации. Если длина поля общего адреса 2 байта, то адреса от 1 до 65534 используются как адрес отдельной станции (адрес станции), а 65535 используется для широковещательной адресации.
Время подтверждения (t1)	Максимальное время, при котором подтверждение не получено (для I-Frames или Test-Frames) до отключения соединения. По стандарту этот таймаут называется t1 (по умолчанию 15с, минимальное значение - 1с, максимальное - 255с).
Время подтверждения полученных сообщений (t2)	Максимальное время перед подтверждением полученных сообщений, которые еще не были подтверждены, используя S формата APDU. По стандарту этот таймаут называется t2 (по умолчанию 10с, минимальное значение - 1с, максимальное - 255с).
Максимальное свободное время (t3)	Максимальное время, когда соединение может быть свободным перед отправкой тестового кадра. По стандарту этот таймаут называется t3 (по умолчанию 20с, минимальное значение - 1с, максимальное - 172800с (48ч)).
Счетчик контроля передачи (k)	Максимальное количество неподтвержденных отправленных APDU формата I. По стандарту этот параметр называется k. По умолчанию 12, минимальное значение - 1, максимальное - 32767.
Счетчик контроля получения (w)	Количество неподтвержденных полученных APDU формата I перед тем, как соединение автоматически будет посылать формат S от APDU, чтобы подтвердить их. По стандарту этот параметр называется w. По умолчанию 8, минимальное значение - 1, максимальное - 32767.

ОБЪЕКТЫ УСТРОЙСТВА

Это свойство содержит список объектов устройства IEC-104 (тэги), которые доступны и управляются AtomMind. Когда добавлено новое устройство IEC-104, один или более объектов должны быть настроены, чтобы сделать данные устройства доступными для системы. Каждый объект IEC-104 представляется единственной переменной 61 контекста 1494 Device .

Настройка	Описание
Активен	Этот флажок показывает активность параметра.
Имя	Имя объекта.
Описание	Описание объекта.
Адрес объекта	Адрес тэга IEC-104 в десятичной форме.
Общий адрес ASDU	Адрес целевой станции или широковещательный адрес. Если длина поля общего адреса 1 байт, то адреса от 1 до 254 используются как адрес отдельной станции (адрес станции), а 255 используется для широковещательной адресации. Если длина поля общего адреса 2 байта, то адреса от 1 до 65534 используются как адрес отдельной станции (адрес станции), а 65535 используется для широковещательной адресации.

Активы Device

Драйвер не поддерживает активы.

Настройки Device

Драйвер устройства IEC-104 создает одну переменную настройки Device на каждый регистр устройства.

Операции Device

ОДНОПОЗИЦИОННАЯ КОМАНДА

Отправляет однопозиционную команду (C_SC_NA_1).

ДВУХПОЗИЦИОННАЯ КОМАНДА

Отправляет двухпозционную команду (C_DC_NA_1).

КОМАНДА УСТАВКИ, НОРМАЛИЗОВАННОЕ ЗНАЧЕНИЕ

Отправляет команду уставки, нормализованное значение (C_SE_NA_1).

КОМАНДА УСТАВКИ, МАСШТАБИРОВАННОЕ ЗНАЧЕНИЕ

Отправляет команду уставки, масштабированное значение (C_SE_NB_1).

КОМАНДА УСТАВКИ, КОРОТКИЙ ФОРМАТ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ

Отправляет команду уставки, короткий формат с плавающей запятой (C_SE_NC_1).

КОМАНДА ПОШАГОВОГО РЕГУЛИРОВАНИЯ

Отправляет команду пошагового регулирования (C_RC_NA_1).

СТРОКА ИЗ 32 БИТОВ

Отправляет строку из 32 битов (C_BO_NA_1).

КОМАНДА СИНХРОНИЗАЦИИ ВРЕМЕНИ

Отправляет команду синхронизации времени (C_CS_NA_1).

КОМАНДА ОПРОСА

Отправляет команду опроса (C_IC_NA_1).

КОМАНДА ЧТЕНИЯ ДИРЕКТОРИИ

Отправляет команду чтения директории (F_SC_NA_1).

КОМАНДА ЧТЕНИЯ ФАЙЛА

Отправляет команду чтения файла.

КОМАНДА ЗАПИСИ ФАЙЛА

Отправляет команду записи файла.

События Device

Драйвер не предоставляет события.

Управление соединением

Устройство будет **Online**, если драйвер сделает следующее:

- Соединение TCP с Device было успешно установлено

Детали синхронизации

Синхронизация между AtomMind Server и устройством IEC-104 включает в себя следующие шаги:

- Создание кэша настроек 502 по списку объектов устройства. Каждая переменная используется для доступа единственного объекта устройства IEC-104.
- Чтение значений регистра IEC-104 и сохранение этих значений в кэше настроек.

10.2.12 IEC-104 Server

Драйвер устройства 518 IEC-104 Server.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.iec-104-server

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

СВОЙСТВА СОЕДИНЕНИЯ

Настройки соединения определяют, как IEC-104 Server связывается с удаленными устройствами клиента IEC-104:

Свойство	Описание
Порт	Номер порта для соединения.
Общий адрес ASDU	Адрес исходной станции.
Таймаут соединения (t0)	Максимальное время ожидания (в милисекундах) сообщения STARDT CON после отправки сообщения STARTDT ACT. По стандарту этот таймаут называется t0.
Время подтверждения (t1)	Максимальное время, при котором подтверждение не получено (для I-Frames или Test-Frames) до отключения соединения. По стандарту этот таймаут называется t1 (по умолчанию 15с, минимальное значение - 1с, максимальное - 255с).

Время подтверждения полученных сообщений (t2)	Максимальное время перед подтверждением полученных сообщений, которые еще не были подтверждены, используя S формата APDU. По стандарту этот таймаут называется t2 (по умолчанию 10с, минимальное значение - 1с, максимальное - 255с).
Максимальное свободное время (t3)	Максимальное время, когда соединение может быть свободным перед отправкой тестового кадра. По стандарту этот таймаут называется t3 (по умолчанию 20с, минимальное значение - 1с, максимальное - 172800с (48ч)).
Счетчик контроля передачи (k)	Максимальное количество неподтвержденных отправленных APDU формата I. По стандарту этот параметр называется k. По умолчанию 12, минимальное значение - 1, максимальное - 32767
Счетчик контроля получения (w)	Количество неподтвержденных полученных APDU формата I перед тем, как соединение автоматически будет посылать формат S от APDU, чтобы подтвердить их. По стандарту этот параметр называется w. По умолчанию 8, минимальное значение - 1, максимальное - 32767.
Выражение преобразования кода качества	Выражение преобразует значение качества AtomMind Server в значение качества IEC.

ОБЪЕКТЫ УСТРОЙСТВА

Это свойство содержит список объектов устройства IEC-104 (тэги), которые доступны и управляются AtomMind. Когда добавлено новое устройство IEC-104 Server, один или более объектов должны быть настроены, чтобы сделать данные устройства доступными для системы. Каждый объект IEC-104 Server представляется единственной переменной 61 контекста 1494 Device .

Настройка	**Описание**
Адрес объекта	Адрес тэга IEC-104 в десятичной форме. Совпадает с именем переменной.
Описание	Описание объекта.
Тип	Тип объекта. Возможные значения: • M_SP_NA_1 (1) • M_ME_TE_1 (35) • M_DP_NA_1 (3) • M_ME_TF_1 (36) • M_ME_NB_1 (11) • M_ME_NC_1 (13) • M_SP_TB_1 (30) • M_DP_TB_1 (31)
Таймаут опроса	Таймаут периодической отправки данных.
Группа	Когда контролируемый пункт получит команду запроса, он ответит всеми значениями точек данных; если это групповая команда запроса, пункт ответит значениями точек данных, принадлежащих группе.

Активы Device

Драйвер не поддерживает активы.

Настройки Device

Драйвер устройства IEC-104 Server создает одну переменную настройки Device на каждый регистр устройства.

Операции Device

- **Команда тестирования**. Отправляя команду тестирования (ASDU 107) IEC подчиненному, IEC мастер может проверить, что соединение с уровнем приложения подчиненного работает корректно.

События Device

Драйвер не предоставляет события.

Управление соединением

Устройство будет **Online**, если драйвер сделает следующее:

- Прослушивание порта TCP успешно.

Детали синхронизации

Драйвер не предоставляет операции.

10.2.13 Ethernet/IP

Драйвер устройства 518 Ethernet/IP позволяет AtomMind Server обмениваться данными с устройствами, поддерживающими протокол Ethernet/IP. Эти устройства могут быть соединены с системой и, как происходит со всеми другими типами устройств, их данные переобразованы в объединенную форму для того, чтобы они могли быть доступны из разных объектов AtomMind. См. статью Device для получения подробной информации о "нормализованном" представлении устройств в AtomMind.

Драйвер Ethernet/IP использует несвязанные явные сообщения (**UCMM**) для обслуживания класса на устройстве поля.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.ethernet-ip

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

НАСТРОЙКИ СОЕДИНЕНИЯ

Настройки соединения определяют, как AtomMind Server взаимодействует с определенным устройством Ethernet/IP. Данные настройки доступны через опцию изменить свойства 1494 Device контекста Device. Доступны следующие свойства соединения:

Свойство	Описание
IP-адрес или имя хоста	Адрес устройства Ethernet/IP.
Порт	Порт устройства.
Слот	Слот устройства.

ОБЪЕКТЫ УСТРОЙСТВА

Это свойство содержит список объектов устройства Ethernet/IP (тэги), которые доступны и управляются AtomMind. Когда добавлено новое устройство Ethernet/IP, один или более объектов должны быть настроены, чтобы сделать данные устройства доступными для системы. Каждый объект Ethernet/IP представляется единственной переменной 61 контекста 1494 Device .

Настройка	Описание
Имя	Имя объекта.
Описание	Описание объекта.
Формат объекта	Определяет формат объекта:

Свойство	Описание

	<table><tr><td>Имя</td><td>Имя поля.</td></tr><tr><td>Описание</td><td>Текстовое описание поля.</td></tr><tr><td>Тип</td><td>Определяет, как интерпретировать значение одного или более прилегающих объектов. См. раздел преобразование типов 549.</td></tr></table>
Доступный для записи	Флаг, указывающий на то, что переменная перезаписываемая.
Класс	Это класс CIP, которому будет отправляться сообщение. Номер класса можно найти в документации устройства поля.
Экземпляр	Это экземпляр класса CIP, которому будет отправлено сообщение. Номер экземпляра класса можно найти в документации устройства. Экземляр с номером ноль получит доступ к сервисам класса, но не особый экземпляр класса.
Атрибут	Это атрибут класса CIP, на котором будет выполняться действие. Некоторые службы запрашивают атрибут, а некоторые нет.

Активы Device

Активы не поддерживаются драйвером.

Настройки Device

Драйвер устройства Ethernet/IP создает одну переменную настройку Device на каждую регистрацию устройства.

ПРЕОБРАЗОВАНИЕ ТИПОВ

Таблица, данная ниже, показывает, как объекты Ethernet/IP преобразуются в переменные контекста Device.

Тип	**Количество байт на каждое значение**	**Формат переменной AtomMind Server**
BOOL	1	Доступен для чтения/записи, 1 колонка булевого типа
SINT	1	Доступен для чтения/записи, 1 колонка целочисленного типа
INT	2	Доступен для чтения/записи, 1 колонка целочисленного типа
DINT	4	Доступен для чтения/записи, 1 колонка целочисленного типа
SHORT_STRING	1 + n (первый байт показывает длину)	Доступен для чтения/записи, 1 колонка типа строка
STRING	2 + n (первый байт показывает длину)	Доступен для чтения/записи, 1 колонка типа строка

Операции Device

Драйвер не проводит операции.

События Device

Драйвер не предоставляет события.

Управление соединением

Устройство будет **Online**, если драйвер сделает следующее:

- Соединение TCP с Device было успешно установлено

Детали синхронизации

Синхронизация между AtomMind Server и устройством Ethernet/IP включает в себя следующие шаги:

- Создание кэша настроек 502 по списку объектов устройства. Каждая переменная используется для доступа единственного объекта устройства Ethernet/IP.

- Чтение значений регистра Ethernet/IP и сохранение этих значений в кэше настроек.

10.2.14 Flexible Драйвер

Flexible драйверы устройств выступают в роли "комплекта создания драйверов". Это позволяет подключать устройства, работающие по проприетарным протоколам, при помощи глубокой настройки системы, не прибегая к разработке определенного драйвера устройства[518] на языке программирования Java.

применимость Flexible драйвера

Flexible драйвер подходит для подключения нового устройства или источника данных, которые отвечают следующим критериям:

- Коммуникация осуществляется через TCP, UDP или последовательный порт локальной машины
- Устройство либо принимает соединения от AtomMind Server, либо устанавливает соединения с сервером самостоятельно
- AtomMind Server выступает как клиент в коммуникациях (т.е. отправляет команды и получает ответы и/или получает асинхронные сообщения с устройства), а устройство выступает в роли сервера
- Протокол устройства не использует "транзакции", т.е. цепочки отдельных сообщений, действующих как отдельный консолидированный блок данных протокола (хотя в отдельных случаях поддержка транзакций все же реализуется)
- Протокол устройства не предполагает парсинг потока входных данных в отдельные блоки данных протокола в зависимости от времени задержки (как иногда происходит в протоколах последовательной передачи данных)

Flexible драйвер работает в следующих случаях:

- Протокол устройства - ASCII (текстовый) или бинарный
- В основе протокола:
  - синхронные сообщения от устройства
  - команды сервера с последующими ответами устройства
  - одновременно синхронные команды/ответы и асинхронные сообщения от устройства (комбинированный режим)
- Ответы устройства могут быть соотнесены с командами сервера двумя способами:
  - устройство может отвечать на команды в той же последовательности, в которой их посылает сервер
  - команды сервера и ответы устройства могут иметь ID (уникальные для каждой сессии и соотнесенные друг с другом). В этом случае устройство может отвечать на команды сервера в произвольном порядке

Flexible драйвер устройств - расширенный инструмент, который требует хорошего понимания единой модели данных[41] платформы, понятий таблицы данных[49], переменные[61], функции[70], события[73], язык выражений[112], устройства[497], а также общей архитектуры драйверов устройств[518].

информация о драйвере

ID плагина[518] драйвера: com.tibbo.linkserver.plugin.device.flexible

общие настройки

Общие настройки драйвера включают Таблицу настроек, которая регулирует взаимодействие с устройствами, устанавливающими входящие соединения с AtomMind Server:

Свойство	Описание
Description	Описание поставщика/модели/типа/версии устройства, обрабатываемого данной записью.
Protocol	Протокол устройства: TCP или UDP.
Port	TCP/UDP порт для приема входящих подключений.

Input Splitter Mode	Если выбрана опция Попытка разделить на каждом байте, драйвер "подает" буфер входных данных для выполнения Выражения разделителя входного потока на каждом байте полученных от устройства данных. Если выбрана опция Попытка разделить в конце блока, Выражение разделителя входного потока вычисляется только однажды, после получения блока байтов от устройства с использованием единственной операции чтения сокета Java/OC. Опция Попытка разделить в конце блока может повысить производительность, т.к. количество попыток анализа буфера входящих данных значительно сокращается.  Важно понимать, что входящие блоки данных, полученные из входного сокета, не связаны напрямую с блоками данных протокола (PDU). Сетевая и операционная системы (как на стороне устройства, таки и на стороне AtomMind Server) могут случайным образом разбивать или объединять блоки данных, отправленных самим устройством. Нельзя гарантировать, что каждый PDU, отправленный устройством в одной операции, будет получен драйвером в одном блоке данных.
Input Stream Splitter Expression	Используется для извлечения команд из буфера входящих данных. Возвращает количество первых байтов в буфере, которые образуют полностью определенный блок протокола, либо null, если буфер не содержит полных блоков протокола.
Encode Expression	Осуществляет кодировку исходящей команды перед отправкой на сокет или последовательный порт. Возвращает строку с закодированной командой. Может использоваться для добавления префиксов, CRC или любого другого способа кодирования команд. Должно ссылаться на оригинальную команду через переменную среды `command`.
Decode Expression	Декодирует входящие команды после извлечения из потока входящих данных с помощью Выражения разделитель входного потока. Может удалять префиксы и суффиксы команд, проверять CRC или проверять целостность команды любым другим способом. Должно возвращать преобразованную команду в виде строки, либо NULL, если команда неверная и должна быть отброшена.
Device ID Detection Method Expression	Это выражение должно возвращать числовое значение, идентифицирующее метод, который будет использоваться для определения ID устройства: • Если выражение возвращает **0,** ID устройства будет обнаруживаться **синхронно** путем отправки специальной команды по определению присвоенного устройству ID в соответствии с настройками Выражения чтения ID устройства и Выражения результата чтения ID устройства • Если выражение возвращает **1**, ID устройства будет обнаруживаться **асинхронно** из первого сообщения от устройства, которое будет проанализировано с помощью Выражения результата чтения ID устройства
Read Device ID Expression	Это выражение должно возвращать команду для отправки на устройство, чтобы получить ID этого устройства.
Device ID Processing Expression	Выражение используется, чтобы анализировать ответ устройства на Выражение чтения ID устройства и возвращать ID устройства. Ответ доступен в виде строки, содержащейся в переменной окружения `command`. Выражение может ссылаться на него через ссылку `{env/command}`.
Encoding	Кодировка строк, используемая драйвером.

настройки уровня пользователя

Не определены.

свойства аккаунта устройства

СВОЙСТВА СОЕДИНЕНИЯ

Свойства соединения определяют, как AtomMind Server общается с устройством. Доступ в этим свойствам можно получить через действие Редактировать свойства аккаунта устройства[1494] контекста Device. Доступны следующие свойства соединения:

Свойство	**Описание**
Connection	Тип соединения: Исходящее (драйвер сам устанавливает соединение) или Входящее (одно из входящих соединений устройства будет относиться к аккаунту этого

	устройства).
Device ID	ID аккаунта данного устройства, используемый для соотнесения с входящим соединением устройства.
Mode	Способ связи устройств: через TCP, UDP или последовательный порт. Используется только для исходящих соединений устройства.
IP address or Host Name	Адрес устройства (для связи через TCP и UDP). Используется только для исходящих соединений устройства.
Port	Порт устройства для связи через TCP и UDP. Используется только для исходящих соединений устройства.
Baud Rate	Скорость передачи данных (для связи через последовательный порт).
Incoming Flow Control	Тип управления входящим потоком: нет, CTS/RTS или XON/XOFF (для связи через последовательный порт).
Outgoing Flow Control	Тип управления исходящим потоком: нет, CTS/RTS или XON/XOFF (для связи через последовательный порт).
Data Bits	Биты данных (для связи через последовательный порт).
Stop Bits	Стоповые биты (для связи через последовательный порт).
Parity	Четность (для связи через последовательный порт).
Timeout	Таймаут команды (по умолчанию 30 секунд). Если за это время не получен ответ, соответствующий отправленной сервером команде, команда считается "неотвеченной", и ее выполнение завершается ошибкой.

ОБМЕН КОМАНДАМИ

Эта таблица настраивает базовые параметры обмена блоками протокольных данных с устройством.

Приведенные ниже настройки обмена командами будут применяться только для Исходящих соединений устройства.

В случае Входящего соединения, будут использоваться настройки обмена командами, указанные в той записи общих настроек драйвера, которая использовалась для принятия Входящего соединения.

Свойство	**Описание**
Input Stream Splitter Expression	Используется для извлечения команд из буфера входящих данных. Возвращает количество первых байтов в буфере, которые образуют полностью определенный блок протокола, либо null, если буфер не содержит полных блоков протокола.
Input Splitter Mode	Если выбрана опция Попытка разделить на каждом байте, драйвер "подает" буфер входных данных для выполнения Выражения разделителя входного потока на каждом байте полученных от устройства данных. Если выбрана опция Попытка разделить в конце блока, Выражение разделителя входного потока вычисляется только однажды, после получения блока байтов от устройства с использованием единственной операции чтения сокета Java/OC. Опция Попытка разделить в конце блока может повысить производительность, т.к. количество попыток анализа буфера входящих данных значительно сокращается.  Важно понимать, что входящие блоки данных, полученные из входного сокета, не связаны напрямую с блоками данных протокола (PDU). Сетевая и операционная системы (как на стороне устройства, таки и на стороне AtomMind Server) могут случайным образом разбивать или объединять блоки данных, отправленных самим устройством. Нельзя гарантировать, что каждый PDU, отправленный устройством в одной операции, будет получен драйвером в одном блоке данных.
Encode Expression	Осуществляет кодировку исходящей команды перед отправкой на сокет или последовательный порт. Возвращает строку с закодированной командой. Может использоваться для добавления префиксов, CRC или любого другого способа кодирования команд. Должно ссылаться на оригинальную команду через переменную среды `command`.
Decode Expression	Декодирует входящие команды после извлечения из потока входящих данных с помощью Выражения разделитель входного потока. Может удалять префиксы и

	суффиксы команд, проверять CRC или проверять целостность команды любым другим способом. Должно возвращать преобразованную команду в виде строки, либо NULL, если команда неверная и должна быть отброшена.
Encoding	Кодировка строк, используемая драйвером.

ОПЕРАЦИИ

Таблица содержит различные выражения, которые определяют базовые операции для Flexible драйвера.

Свойство	Описание
Asynchronous Command Detector Expression	Проверяет содержимое декодированной команды и возвращает TRUE, если это асинхронное сообщение от устройства (событие, обновление значения и пр.), либо FALSE, если это синхронный ответ на предварительно отправленную сервером команду.
Event/Variable Update Qualifier Expression	Проверяет содержимое асинхронного сообщения от устройства и возвращает 0, если это событие, и 1, если это инициированное устройством обновление значения переменной.
Command ID Expression	Используется, чтобы узнать уникальный ID команды в рамках текущей сессии. Это можно сделать либо путем генерирования ID (произвольно, последовательно и пр.), либо проверкой данных Команды, доступных через переменную среды `command`. ID, полученный в результате этого выражения, будет позднее соотнесен с ID, полученными в результате Выражения ID ответа, чтобы найти ответ на определенную команду.  Если в коммуникационном протоколе не используются ID команды/ответа, настройте данное выражение и **Выражение ID ответа** на возврат какого-либо статического значения, например, `"id"`. Это соотнесет первый полученный ответ с первой отправленной командой, и т.д.
Reply ID Expression	Анализирует содержимое синхронного ответа устройства и возвращает его ID как строку. Этот ID будет сопоставлен с ID отправленной ранее сервером команды, который был сгенерирован и получен в результате Выражения ID команды.  Если в коммуникационном протоколе не используются ID команды/ответа, настройте данное выражение и **Выражение ID команды** на возврат какого-либо статического значения, например, `"id"`. Это соотнесет первый полученный ответ с первой отправленной командой, и т.д.
Variable Name Expression	Используется для извлечения имени измененной переменной AtomMind Server из команды, которая была классифицирована как обновление значения переменной устройством.
Variable Timestamp Expression	Используется для извлечения даты/времени изменения переменной устройства из команды, которая была классифицирована как обновление значения переменной устройством.
Variable Quality Expression	Используется для извлечения свойства нового значения из команды, которая была классифицирована как обновление значения переменной устройством.
Variable Value Expression	Используется для извлечения значения новой переменной из команды, которая была классифицирована как обновление значения переменной устройством. Должно возвращать таблицу данных.
Event Name Expression	Используется для извлечения имени (типа) события AtomMind Server из команды, которая была классифицирована как асинхронное оповещение от устройства.
Event Timestamp Expression	Используется для извлечения даты/времени события устройства из команды, которая была классифицирована как асинхронное оповещение от устройства.
Event Level Expression	Используется для извлечения уровня 75 события AtomMind Server из команды, которая была классифицирована как асинхронное оповещение от устройства. Должно возвращать целочисленное значение.
Event Data Expression	Используется для извлечения данных, относящихся к событию, из команды, которая была классифицирована как асинхронное оповещение от устройства. Должно возвращать таблицу данных.
Variable Definitions Expression	Выражение должно получать список доступных переменных от устройства и возвращать его как таблицу данных. Обычно ссылается на какой-либо набор правил скрипта 879 или модели 810, который, в свою очередь, вызывает функцию **Отправлять**

	команду для получения частей метаданных устройства через специфичные для устройства команды. Каждый ряд в результирующей таблице данного выражения будет затем преобразован в определение отдельной переменной, как описано в разделе **Работа с динамическими метаданными**.
Function Definitions Expression	Выражение должно получать список доступных функций от устройства и возвращать его как таблицу данных. Обычно ссылается на какой-либо набор правил скрипта 879 или модели 810, который, в свою очередь, вызывает функцию **Отправлять команду** для получения частей метаданных устройства через специфичные для устройства команды. Каждый ряд в результирующей таблице данного выражения будет затем преобразован в определение отдельной функции, как описано в разделе **Работа с динамическими метаданными**.
Event Definitions Expression	Выражение должно получать список доступных типов событий от устройства и возвращать его как таблицу данных. Обычно ссылается на какой-либо набор правил скрипта 879 или модели 810, который, в свою очередь, вызывает функцию **Отправлять команду** для получения частей метаданных устройства через специфичные для устройства команды. Каждый ряд в результирующей таблице данного выражения будет затем преобразован в определение отдельного события, как описано в разделе **Работа с динамическими метаданными**.
Start Synchronization Expression	Выражение вычисляется в начале каждого цикла синхронизации 514 устройств. Может выполнять часть инициализации контекста драйвера и/или устройства. Результат выражения игнорируется.
Finish Synchronization Expression	Выражение вычисляется в конце каждого цикла синхронизации 514 устройств. Может выполнять часть очистки контекста драйвера и/или устройства. Результат выражения игнорируется.
Connect Expression	Выражение вычисляется после удачного подключения устройства. Обычно используется для реализации входа устройства в систему и других действий по аутентификации/авторизации через вызов функции **Отправлять команду** для обмена данными с устройством. Результат выражения игнорируется..
Disconnect Expression	Выражение вычисляется после удачного отключения устройства. Результат выражения игнорируется.

СТАТИЧЕСКИЕ ПЕРЕМЕННЫЕ

Таблица определяет, какие переменные устройств будут доступны всегда, в дополнение в полученным от устройства во время чтения динамических метаданных.

Каждая переменная определяется следующими параметрами:

Параметр	Описание
Name	Имя переменной.
Description	Удобочитаемого для человека описание переменной
Format	Формат переменной. Более подробно см. Формат 50.
Readable	Флаг, показывающий, что переменная читается.
Writable	Флаг, показывающий, что переменная записывается.
Help	Подробное описание переменной.
Group	Группа переменной либо NULL, если переменная не входит ни в одну группу.

Read Request Expression	Выражение должно возвращать данные, которые буду отправлены на устройство для запроса текущего значения переменной. Должно возвращать строку. К этой строке сначала применяют **Выражение кодирования,** а результат затем переводится в массив байтов и отправляется на устройство. Преобразование из строки в массив байтов использует кодировку UTF-8.
Read Result Processing Expression	Выражение преобразует данные, вернувшиеся от устройства (полученные через сокет или последовательный порт) в таблицу данных, представляющую значение переменной устройства. Полученные от устройства данные доступны как строка в переменной среды `command`. Выражение может ссылаться на нее через `{env/command}`. Эта строка получается в результате **Выражения раскодирования**, которое преобразует необработанные данные, полученные от устройства. Таблица данных, представляющая значение переменной, обычно создается при помощи функции языка выражений `table()`. Функция `table()` должна использовать формат переменной, который можно получить с помощью функции языка выражений `variableFormat()`.
Write Request Expression	Выражение должно возвращать данные, которые будут отправлены на устройство для добавления нового значения переменной, измененной на стороне сервера. Фактически, эти данные содержат значение обновленной переменной, кодированные в соответствии с протоколом устройства. Обновленное значение переменной доступно как таблица данным по умолчанию во время вычисления выражения. Выражение должно возвращать строку. К этой строке сначала применяют **Выражение кодирования,** а результат затем переводится в массив байтов и отправляется на устройство. Преобразование из строки в массив байтов использует кодировку UTF-8.
Write Result Processing Expression	Выражение анализирует данные, вернувшиеся с устройства на запрос записи переменных (полученные через сокет или последовательный порт). Цель выражения - понять успешность операции записи. Выражение должно возвращать текст об ошибке записи в строковом формате, либо NULL, если операция завершилась успешно. Полученный от устройства ответ доступен как строка в переменной среды `command`. Выражение может ссылаться на нее через `{env/command}`. Эта строка получается в результате **Выражения раскодирования**, которое преобразует необработанные данные, полученные от устройства.

СТАТИЧЕСКИЕ ФУНКЦИИ

Таблица определяет, какие функции устройств будут доступны всегда, в дополнение в полученным от устройства во время чтения динамических метаданных.

Каждая функция определяется следующими параметрами:

Параметр	Описание
Name	Имя функции.
Description	Удобочитаемого для человека описание функции.
Input Format	Входной формат функции. Подробнее см. Формат 50.
Output Format	Выходной формат функции. Подробнее см. Формат 50.
Help	Подробное описание функции.
Group	Группа функции либо NULL, если функция не входит ни в одну группу.

Execution Request Expression	Выражение должно возвращать данные, которые будут отправлены на устройство, чтобы выполнить операцию устройства. Фактически, эти данные содержат входные параметры операции (функции), кодированные в соответствии с протоколом устройства. Входные параметры операции (функции) доступны как таблица данных по умолчанию во время вычисления выражения. Выражение должно возвращать строку. К этой строке сначала применяют **Выражение кодирования,** а результат затем переводится в массив байтов и отправляется на устройство. Преобразование из строки в массив байтов использует кодировку UTF-8.
Execution Result Processing Expression	Выражение преобразует выходные данные операции устройства (полученные через сокет или последовательный порт) в таблицу данных, представляющую выходные данные функции устройства на стороне сервера. Полученные от устройства данные доступны как строка в переменной среды `command`. Выражение может ссылаться на нее через `{env/command}`. Эта строка получается в результате **Выражения раскодирования**, которое преобразует необработанные данные, полученные от устройства. Таблица данных, представляющая выходные данные функции, обычно создается при помощи функции языка выражений `table()`. Функция `table()` должна использовать формат функции контекста, который можно получить с помощью функции языка выражений `functionOutputFormat()`.

СТАТИЧЕСКИЕ СОБЫТИЯ

Таблица определяет, какие события устройства будут доступны всегда, в дополнение в полученным от устройства во время чтения динамических метаданных.

Каждое событие определяется следующими параметрами:

Параметр	Описание
Name	Имя события
Description	Удобочитаемого для человека описание события.
Format	Формат события. Подробнее см. Формат 50.
Help	Подробное описание события.
Level	Уровень события по умолчанию.
Group	Группа события, либо NULL, если событие не входит ни в одну группу.

ДИНАМИЧЕСКИЕ ПЕРЕМЕННЫЕ

Таблица определяет выражения, которые будут использоваться для извлечения параметров определения переменной из рядов результирующей таблицы **Выражения определения переменных**. Таким образом, она используется для преобразования каждого ряда этой таблицы в отдельное определение переменной.

Параметр	Описание
Name Expression	Выражение возвращает имя переменной.
Description Expression	Выражение возвращает удобочитаемое для человека описание переменной.
Format Expression	Выражение возвращает строковое представление формата переменной, либо null, если ее формат динамический. Более подробно см. раздел Кодирование формата таблицы 2124. Строка формата может использовать видимые или невидимые разделители 2130.

Readable Expression	Выражение возвращает флаг, указывающий, что переменная читается.
Writable Expression	Выражение возвращает флаг, указывающий, что переменная пишется.
Help Expression	Выражение возвращает подробное описание переменной.
Group Expression	Выражение возвращает группу переменной либо NULL, если переменная не входит ни в одну группу.
Read Request Expression Constructor	**Это выражение должно возвращать текст другого выражения.** Это другое, результирующее выражение, должно возвращать данные, которые будут отправлены на устройство для запроса текущего значения переменной. Выражение должно возвращать строку. К этой строке сначала применяют **Выражение кодирования,** а результат затем переводится в массив байтов и отправляется на устройство. Преобразование из строки в массив байтов использует кодировку UTF-8.
Read Result Processing Expression Constructor	**Это выражение должно возвращать текст другого выражения.** Это другое, результирующее выражение преобразует вернувшиеся от устройства данные (полученные через сокет или последовательный порт) в таблицу данных, представляющую значение переменной устройства. Полученные от устройства данные доступны как строка в переменной среды `command`. Выражение может ссылаться на нее через `{env/command}`. Эта строка получается в результате **Выражения раскодирования**, которое преобразует необработанные данные, полученные от устройства. Таблица данных, представляющая значение переменной, обычно создается при помощи функции языка выражений `table()`. Функция `table()` должна использовать формат, который можно получить с помощью функции языка выражений `variableFormat()`.
Write Request Expression Constructor	**Это выражение должно возвращать текст другого выражения.** Это другое, результирующее выражение должно возвращать данные, которые будут отправлены на устройство для добавления нового значения переменной, измененной на стороне сервера. Фактически, эти данные содержат значение обновленной переменной, кодированное в соответствии с протоколом устройства. Обновленное значение переменной доступно как таблица данных по умолчанию во время вычисления выражения. Выражение должно возвращать строку. К этой строке сначала применяют **Выражение кодирования,** а результат затем переводится в массив байтов и отправляется на устройство. Преобразование из строки в массив байтов использует кодировку UTF-8.
Write Result Processing Expression Constructor	**Это выражение должно возвращать текст другого выражения.** Это другое, результирующее выражение анализирует вернувшиеся с устройства данные на запрос записи переменной (полученные через сокет или последовательный порт). Цель выражения - понять успешность операции записи. Выражение должно возвращать текст об ошибке записи в строковом формате, либо NULL, если операция завершилась успешно. Полученный от устройства ответ доступен как строка в переменной среды `command`. Выражение может ссылаться на нее через `{env/command}`. Эта строка получается в результате **Выражения раскодирования**, которое преобразует необработанные данные, полученные от устройства.

ДИНАМИЧЕСКИЕ ФУНКЦИИ

Таблица определяет выражения, которые будут использоваться для извлечения параметров определения функции из рядов таблицы, полученной в результате **Выражение определения функций.** Таким образом, она используется для преобразования каждого ряда этой таблицы в отдельное определение функции.

Параметр	Описание
Name Expression	Выражение возвращает имя функции.
Description Expression	Выражение возвращает удобочитаемое для человека описание функции.
Input Format Expression	Выражение возвращает строковое представление входного формата функции, либо null, если ее входной формат динамический. Более подробно см. раздел Кодирование формата таблицы 2124. Строка формата может использовать видимые или невидимые разделители 2130.
Output Format Expression	Выражение возвращает строковое представление выходного формата функции, либо null, если ее выходной формат динамический. Более подробно см. раздел Кодирование формата таблицы 2124. Строка формата может использовать видимые или невидимые разделители 2130.
Help Expression	Выражение возвращает подробное описание функции.
Group Expression	Выражение возвращает группу функции, либо NULL, если переменная не входит ни в одну группу
Execution Request Expression Constructor	**Это выражение должно возвращать текст другого выражения.** Это другое, результирующее выражение, должно возвращать данные, которые будут отправлены на устройство для выполнения операции устройства. фактически, эти данные содержат входные параметры операции (функции), кодированные в соответствии с протоколом устройства. Входные параметры операции (функции) доступны как таблица данных по умолчанию во время вычисления выражения. Выражение должно возвращать строку. К этой строке сначала применяют **Выражение кодирования,** а результат затем переводится в массив байтов и отправляется на устройство. Преобразование из строки в массив байтов использует кодировку UTF-8.
Execution Result Processing Expression Constructor	**Это выражение должно возвращать текст другого выражения.** Это другое, результирующее выражение преобразует выходные данные операции устройства (полученные через сокет или последовательный порт) в таблицу данных, представляющую результат функции устройства на стороне сервера. Полученный от устройства ответ доступен как строка в переменной среды `command`. Выражение может ссылаться на нее через `{env/command}`. Эта строка получается в результате **Выражения раскодирования**, которое преобразует необработанные данные, полученные от устройства. Таблица данных, представляющая выходные данные функции, обычно создается при помощи функции языка выражений `table()`. Функция `table()` должна использовать формат функции контекста, который можно получить с помощью функции языка выражений `functionOutputFormat()`.

ДИНАМИЧЕСКИЕ СОБЫТИЯ

Таблица определяет выражения, которые будут использоваться для извлечения параметров определений событий из рядов таблицы, полученной в результате **Выражение определения событий**. Таким образом, она используется для преобразования каждого ряда этой таблицы в отдельное определение события.

Параметр	Описание
Name Expression	Выражение возвращает имя события.

Description Expression	Выражение возвращает удобочитаемое для человека описание события.
Format Expression	Выражение возвращает строковое представление формата события, либо null, если его формат динамический. Более подробно см. раздел Кодирование формата таблицы 2124. Строка формата может использовать видимые или невидимые разделители 2130.
Help Expression	Выражение возвращает подробное описание события.
Level Expression	Выражение возвращает уровень события по умолчанию.
Group Expression	Выражение возвращает группу события, либо NULL, если событие не входит ни в одну группу

активы Device

Драйвер не поддерживает активы.

настройки Device

Аккаунт Flexible устройства предлагает комбинацию из двух наборов переменных устройств:

- Переменные, определяемые вручную в таблице **Статические переменные**
- Переменные, чьи определения динамически получаются с устройства в соответствии с правилами, описанными в разделе **Работа с динамическими метаданными**

операции Device

Аккаунт Flexible устройства предлагает комбинацию из двух наборов функций устройств:

- Функции, определяемые вручную в таблице **Статические функции**
- Функции, чьи определения динамически получаются с устройства в соответствии с правилами, описанными в разделе **Работа с динамическими метаданными**

события Device

Аккаунт Flexible устройства предлагает комбинацию из двух наборов событий устройств:

- События, определяемые вручную в таблице **Статические события**
- События, чьи определения динамически получаются с устройства в соответствии с правилами, описанными в разделе **Работа с динамическими метаданными**

подключение

Драйвер переводит устройство в режим **онлайн**, если:

- TCP соединение с устройством успешно установлено (для режима TCP)
- Последовательный порт успешно открыт (для последовательного режима)

Синхронизация

Подробно о синхронизации можно прочитать в описании flexible драйвера устройств.

10.2.14.1 Исходящие и входящие соединения с устройством

Flexible драйвер может работать как с устройствами, принимающими входящие соединения от AtomMind Server, так и с теми, которые сами подключаются к серверу. Однако, работа в этих двух режимах значительно отличается.

ИСХОДЯЩИЕ СОЕДИНЕНИЯ

В этом случае в аккаунте устройства указываются все необходимые коммуникационные параметры устройства (адрес, порт). AtomMind Server подключается к устройству и начинает обмен командами.

ВХОДЯЩИЕ СОЕДИНЕНИЯ

Если устройства самостоятельно подключаются к серверу (как большинство мобильных устройств с SIM-картами), коммуникация с каждым устройством определенного поставщика, типа, модели или версии осуществляется при помощи специальной записи в таблице общих настроек драйвера. Эта запись позволяет серверу "слушать" определенный TCP или UDP порт и принимать входящие соединения от устройства.

После того, как соединение устройства принято, драйвер начинает ограниченный обмен командами для того, чтобы:

- обнаружить **ID устройства** (строка с уникальным для данного устройства значением, например, номером IMEI)
- найти соответствующий контекст аккаунта устройства 1494 с совпадающим ID устройства (в настройках аккаунта устройства)
- соотнести входящее соединение с этим устройством

Когда входящее соединение связано с аккаунтом определенного устройства, дальнейшая работа не отличается от режима с исходящим соединением.

10.2.14.2 Работа с потоками данных

Обычно Flexible драйвер работает с двумя потоками данных:

- Исходящий поток данных (данные, исходящие через TCP соединение, последовательность отправленных UDP датаграмм или данные, отправленные через порт последовательного подключения)
- Входящий поток данных (данные, входящие через TCP соединение, последовательность входящих UDP датаграмм или данные, полученные через порт последовательного подключения)

Оба потока инициализируются на стадии подключения устройства цикла синхронизации. Это предполагает установление TCP соединения или открытие последовательного порта.

После успешного завершения стадии подключения, драйвер выполняет два параллельных действия:

- Запись данных в исходящий поток об:
  - Операциях чтения/записи переменных устройства
  - Операциях выполнения функций
  - Других стадиях цикла синхронизации (чтение метаданных, синхронизация старта/остановки и пр.)
- Чтение данных из входящего потока, расщепление их на протокольные блоки и соотнесение этих блоков с:
  - Ответами на ранее отправленные команды (команды чтения/записи переменных, команды выполнения функций, команды, отправленные на других стадиях синхронизации, например, чтение метаданных)
  - Асинхронными уведомлениями устройств, которые преобразуются в события 73 устройства

10.2.14.3 Настройка драйвера

В разделе перечислены основные обязательные шаги для настройки коммуникации с новым типом устройства.

1. Определите статические метаданные устройства или правила чтения динамических метаданных
   a. Если устройство использует статические метаданные (т.е. нельзя опросить устройство для определения доступных переменных, функций и событий), настройте **Статические переменные**, **Статические функции** и **Статические события** в профиле устройства.
   b. Если устройство предоставляет метаданные динамически (информацию о доступных настройках, операциях, и типах оповещений), настройте необходимые выражения в группе **Динамические метаданные** раздела **Операции** в профиле устройства.
2. Настройте коммуникации с устройством, установив выражения в группах **Обмен командами** и **Операции с данными** раздела **Операции** в профиле устройства.
3. Если ваше устройство отправляет асинхронные уведомления, настройте их преобразование в AtomMind события с помощью добавления выражений в группе **Обработка асинхронных событий** раздела **Операции** в профиле устройства.
4. Если ваше устройство отправляет асинхронные сообщения при изменении настроек на стороне устройства, настройте их преобразование в AtomMind обновления переменных устройства с помощью добавления

выражений в группе **Обработка асинхронного обновления переменной** раздела **Операции** в профиле устройства.

10.2.14.4 Обработка входящего потока данных

В разделе объясняется, как Flexible драйвер обрабатывает поток входящих данных.

Драйвер постоянно получает байты данных через TCP соединение, последовательный порт или через UDP-датаграммы. Все эти байты добавляются в конец так называемого *Буфера входящих данных*.

Просмотреть текущее содержимое буфера можно в свойстве **Статистика буфера входящих данных,** доступном через действие Показать статус 111 в контексте Flexible драйвера.

Каждый раз при получении и добавлении нового байта/байтов данных в буфер, драйвер вычисляет **Выражение разделитель входного потока**. Это выражение должно определить, содержит ли входящий буфер полностью определенный протокольный блок. Если это так, выражение должно вернуть длину протокольного блока (т.е. количество байтов в нем). В противном случае, если цельный блок получен не полностью, выражение должно вернуть 0 (нуль).

Выражение разделитель входного потока часто анализирует:

- Первые байты входного буфера, которые содержат длину текущего протокольного блока. В этом случае, выражение проверяет байты определенной "длины команды" и возвращает их значение, если в буфере содержится такое же, либо большее количество данных.
- Некоторые символы разделителей команд или, если точнее, первый случай их употребления в буфере входящих данных. Если такие символы обнаружены, выражение возвращает их индекс в буфере, поскольку он соответствует длине команды.

Когда **Выражение разделитель входного потока** возвращает ненулевое значение (например, *N*), драйвер удаляет первые *N* байтов из входного буфера и отправляет их на дальнейшую обработку. На этом этапе байты преобразуются в строку с использованием UTF-8 кода, и в дальнейшем именуются *Командой*.

После того как Команда полностью определена, драйвер вычисляет **Выражение раскодирования**. Это выражение должно преобразовать (раскодировать) Команду и вернуть преобразованные данные Команды в виде строки.

Если Команду невозможно успешно раскодировать (например, если данные команды несогласованные), **Выражение раскодирования** вернет NULL.

Обычно **Выражение раскодирования** делает следующее:

- Обрезка данных команды путем удаления несодержательных префиксов и суффиксов команды
- Проверка достоверности данных команды, например, вычисление и проверка CRC через вызов внешнего скрипта 879

Выражение раскодирования и все остальные выражения в группах **Обмен командами** и **Операции с данными** имеют доступ к данным Команды через переменную среды `command` изнутри выражения. Используйте ссылку `{env/command}`, чтобы получить строковое представление байтов Команды.

После раскодирования Команды драйверу нужно решить, является ли данная Команда:

- синхронным ответом на одну из ранее отправленных команд,
- либо асинхронным сообщением, представляющим событие устройства или асинхронное обновление значения какой-либо переменной устройства.

Чтобы это определить, драйвер вычисляет **Выражение детектор асинхронных команд**. Данное выражение должно проанализировать данные Команды и вернуть:

- TRUE, если Команда - это асинхронное событие или обновление переменной. Дальнейшая обработка идет по алгоритму, описанному в разделе **Обработка асинхронных команд**.
- FALSE в иных случаях, т.е. если Команда - это ответ на одну из ранее отправленных сервером команд. Дальнейшая обработка идет по алгоритму, описанному в разделе **Обработка синхронных ответов**.

10.2.14.5 Обработка асинхронных команд

Данная последовательность действий выполняется, когда полученная от устройства Команда распознается как асинхронная.

Сначала драйвер вычисляет **Выражение классификатора события/обновления переменной,** чтобы определить, является ли Команда Событием (сгенерированным устройством уведомлением) или Обновлением

переменной (доставкой изменения значения на стороне устройства, которая содержит новое значение). Квалифицированное выражение должно вернуть:

- 0, если Команда является Событием
- 1, если Команда является Обновлением переменной

Обработка асинхронных событий

Если Команда считается событием, вычисляются выражения в группе **Обработка асинхронных событий:**

- **Выражение имени события** должно возвращать имя AtomMind-события 73.
- **Выражение временной метки события** должно возвращать время появления события в устройстве. По умолчанию данное выражение установлено на `now()`, т.е. временная метка события будет установлена на конкретное время, когда она создается на сервере.
- **Выражение уровня события** должно возвращать уровень 75 %AG%>-события 73. Если возвращает NULL, будет использован уровень по умолчанию, определенный таблицей **Статические события**.
- **Выражение данных события** должно создавать и возвращать Таблицу со специфичными для события данными. Обычно создание Таблицы, представляющей данные события, осуществляется при помощи функции `table()` языка выражений. Функция `table()` должна использовать формат события, который можно извлечь функцией `eventFormat()` языка выражений.

После вычисления описанных выше выражений драйвер инициирует новое AtomMind Server-событие 73, представляющее событие устройства 510. Затем данное событие хранится и обрабатывается обычным способом.

ОБработка асинхронных обновлений пременных

Если Команда считается обновлением переменной, вычисляются выражения в группе **Обработка асинхронного обновления переменной**:

- **Выражение имени переменной** должно возвращать имя измененной AtomMind-переменной 61.
- **Выражение временной метки переменной** должно возвращать время, когда произошло изменение на стороне устройства. По умолчанию данное выражение установлено на `now()`, т.е. временная метка обновления переменной будет установлена на конкретное время при обработке на сервере.
- **Выражение качества переменной** должно возвращать качество нового значения переменной либо NULL, если метрика качества должна быть не определена.
- **Выражение значения переменной** должно создавать и возвращать Таблицу данных, представляющую новое значение переменной. Обычно создание Таблицы, представляющей значение переменной, осуществляется при помощи функции `table()` языка выражений. Функция `table()` должна использовать формат переменной, который можно извлечь функцией `variableFormat()` языка выражений.

После вычисления описанных выше выражений драйвер регистрирует новое обновление AtomMind Server - переменной устройства 501, которое потом хранится и обрабатывается обычным способом.

10.2.14.6 Обработка синхронных ответов

Данная последовательность действий выполняется, когда полученная от устройства Команда распознается как ответ на предварительно отправленную сервером команду.

Сначала нужно найти команду сервера, ответ на которую представлен текущей Командой устройства.

Для этого драйвер вычисляет **Выражение ID ответа**, которое должно вернуть строку с уникальным ID ответа.

После того, как ID ответа устройства известен, драйвер находит команду сервера с таким же ID и:

- если команда сервера была отправлена во время чтения переменной, процесс чтения возобновляется, и **Выражение обработки результата чтения** вычисляется для преобразования Команды в значение переменной.
- если команда сервера была отправлена во время записи переменной, процесс записи возобновляется, и **Выражение обработки результата записи** вычисляется, чтобы определить, подтверждает ли устройство успешность записи.
- если команда сервера была отправлена во время выполнения функции, процесс выполнения возобновляется, и **Выражения обработки результата выполнения** вычисляется для преобразования Команды в выходное значение функции.

Если коммуникационный протокол устройства использует числовые ID команд, эти ID должны быть преобразованы в строки при помощи **Выражения ID команды** и **Выражения ID ответа**.

10.2.14.7 Работа с метаданными

Метаданные 500 устройства представляют собой определения переменных (настройки), функций (операции) и событий (оповещения), предоставляемых устройством. Flexible драйвер поддерживает два типа метаданных устройств:

- **Статические метаданные** предполагают, что определения доступных переменных/функций/событий зафиксированы в профиле устройства на стороне сервера
- **Динамические метаданные** предполагают, что определения переменных/функций/событий извлекаются из устройства при синхронизации 514

Настройка Flexible драйвера отличается в случае со статическими и динамическими метаданными.

Работа со статическими метаданными

Настроить Flexible драйвер для работы с фиксированными наборами переменных/функций/событий относительно легко. Необходимо заполнить таблицы **Статические переменные**, **Статические функции** и **Статические события**, описанные в разделе **Свойства аккаунта устройства**. Эти таблицы определяют параметры и форматы сущностей устройства, а также правила чтения/записи переменных и выполнения функций.

Работа с динамическими метаданными

Чтобы настроить Flexible драйвер для чтения определений переменных/функций/событий с устройства, выполните следующее:

- Задайте **Выражение определения переменных**, **Выражение определения функций** и **Выражение определения событий** в группе **Динамические метаданные** раздела **Операции** в профиле устройства. Каждое из этих выражений должно возвращать Таблицу данных с перечнем всех переменных/функций/событий, извлеченных с устройства. Каждый ряд в этих таблицах должен содержать определение одной переменной/функции/события.
- Обработка таблиц, полученных в результате указанных выше выражений, будет происходить построчно. Для каждого ряда драйвер будет вычислять выражения из групп **Динамические переменные**, **Динамические функции** и **Динамические события** соответственно. После вычисления всех выражений драйвер будет использовать их результаты для создания определения переменной/функции/события и добавления его к метаданным.

Для каждого ряда в таблице определений переменной, драйвер будет вычислять все выражения в группе **Динамические переменные**:

- **Выражение имени** должно возвращать имя переменной
- **Выражение описания** должно возвращать удобочитаемое для человека описание переменной
- **Выражение чтения** должно возвращать TRUE, если переменная читается, и FALSE в другом случае
- **Выражение записи** должно возвращать TRUE, если переменная записывается, и FALSE в другом случае
- **Выражение формата** должно возвращать формат переменной, кодированный в строку
- **Выражение группы** должно возвращать группу переменной, либо NULL, если переменная не входит ни в одну группу
- **Выражение подсказки** должно возвращать подробное описание переменной
- **Выражение запроса чтения** должно возвращать текст другого выражения, которое будет использоваться для подготовки запроса чтения переменной с дальнейшей отправкой на устройство
- **Выражение разбора запроса чтения** должно возвращать текст другого выражения, которое будет использоваться для преобразования ответа устройства в новое значение переменной
- **Выражение запроса записи** должно возвращать текст другого выражения, которое будет использоваться для подготовки запроса записи переменной с дальнейшей отправкой на устройство
- **Выражение разбора запроса записи** должно возвращать текст другого выражения, которое будет использоваться для проверки успешности операции записи

10.2.14.8 Ручной обмен командами

Контекст Flexible драйвера предлагает функцию **Отправить команду** для отправки необработанных команд на устройство и получения ответов от него. Функция используется для чтения метаданных устройств, проведения аутентификации устройств и пр.

Имя функции: sendCommand

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*

Входные записи: 1

Входной Формат 50:

Имя	Тип	Описание
command	String	Необработанные данные команды для отправки на устройство. Будут преобразованы в массив байтов при помощи кодировки UTF-8.
id	String	Уникальный ID команды. Используется для поиска ответа устройства на данную определенную команду. ID входящих команд вычисляются при помощи **Выражения ID ответа**.

Выходные записи: 1

Выходной Формат 50:

Имя	Тип	Описание
command	String	Строка с необработанным ответом устройства, конвертированная из массива байтов при помощи кодировки UTF-8.

10.2.14.9 Отладка подключений устройств

Flexible драйвер инициирует специальное событие **Ошибка драйвера Flexible,** когда какое-нибудь выражение по обработке данных заканчивается ошибкой. Такие события используются для диагностики проблем во время настройки драйвера для работы с новым типом устройств.

Имя события flexibleDriverError

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*

Время хранения: Не хранится

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Комментарии
description	String	Тип выражения с ошибкой вычисления.
expression	String	Текст выражения с ошибкой вычисления.
error	String	Текст ошибки.
defaultTable	Data Table	Таблица по умолчанию во время вычисления выражения.
defaultContext	String	Контекст по умолчанию во время вычисления выражения.

environment	Data Table	Переменные среды, доступные во время вычисления выражения.
stackTrace	Data Table	Трассировка внутреннего стека ошибки. Дает команде поддержки AtomMind возможность детальной диагностики.

10.2.15 Графовая база данных

Драйвер устройства 518 Графовая база данных позволяет AtomMind Server хранить составляющие топологии узлы, ребра и их свойства. Он предназначен для хранения данных, не ограничивая их заранее определенной моделью. В AtomMind драйвер GraphDB устанавливает соответствие между наборами экземпляров класса и отношениями между ними.

Конструкционно, графовые базы данных позволяют легко и быстро извлекать сложные иерархические структуры, которые сложно смоделировать в реляционных системах.

Первый класс драйвера БД - `Neo4j`. Он способен поддерживать сверхбольшие неизменяемые графы и предоставляет специальный язык запросов по графам. Этот язык используется для загрузки пользовательских частей топологии для анализа в оперативной памяти или визуализации.

Другой класс драйвера БД - `TinkerGraph`. Это легкая база данных графов в памяти, которая служит эталонной реализацией для модели графов свойств.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.graphdb

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства аккаунта Device

НАСТРОЙКИ СОЕДИНЕНИЯ

Настройки соединения определяют, как AtomMind Server взаимодействует с сервером GraphDB. Доступны следующие свойства соединения:

Свойство	Описание
Встроенная СУБД	Определяет, должен ли использоваться экземпляр графовой БД, встроенный в AtomMind Server.
Адрес СУБД	IP адрес или имя хоста сервера внешней графовой БД.
Порт СУБД	Номер порта сервера внешней графовой БД.
Имя пользователя	Логин внешней графовой БД.
Пароль	Пароль от внешней графовой БД.
Класс драйвера (только если не совместим с JDBC4)	Выберите либо **TinkerGraph**, либо **Neo4j**.
Дополнительные свойства	Таблица с дополнительными свойствами соединения БД. В таблице два столбца: `Property` и `Value`. Для соединения с графовыми БД используется Apache TinkerPop. Список поддерживаемых настроек можно найти в документации по TinkerPop.
Минимальный размер пула запросов	Минимальный размер соединений, поддерживаемый пулом запросов.

Максимальный размер пула запросов	Максимальный размер соединений, поддерживаемый пулом запросов.
Запрос на добавление пула	Приращение, по которому расширяется объем пула запросов.

Активы Device

Активы не поддерживаются драйвером.

Настройки Device

Драйвер устройства GraphDB создает одну динамическую переменную Device, которая называется **Статистика базы данных**. Переменная включает следующие поля:

Поле	Описание
Имя СУБД	Показывает имя БД.
Версия СУБД	Показывает версию БД.
Драйвер СУБД	Имя драйвера вашей указанной БД.
Версия драйвера СУБД	Показывает версию драйвера БД.
Количество соединений	Количество соединений, установленных с БД.
Количество занятых соединений	Количество занятых соединений с БД.

Операции Device

ВЫПОЛНИТЬ ЗАПРОС

Данное действие позволяет выполнять произвольные запросы к БД. Поддерживаются запросы "выбрать" и "обновить".

события Device

Драйвер не предоставляет события.

подключение

Данный драйвер приводит устройство в режим **Онлайн**, если соединение было установлено успешно.

10.2.16 IPMI

Драйвер устройства 518 IPMI позволяет AtomMind Server взаимодействовать с устройствами, поддерживающими **IPMI Протокол**.Данные устройства могут быть подключены к системе и, подобно всем другим типам устройств, их данные преобразовываются в специальную форму, так что доступ к ним возможен от разных экземпляров AtomMind,. Обратитесь к разделу Devices для получения более детальной информации о "нормализованном" представлении устройств в AtomMind.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.ipmi

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

НАСТРОЙКИ СОЕДИНЕНИЯ

Настройки соединения определяют, как AtomMind Server взаимодействует с определенным IPMI устройством. Данные настройки доступны через опцию изменить свойства Device контекста Device. Доступны следующие свойства подключения:

Свойство	Описание
IP адрес или имя хоста	Address of IPMI device.
Пользователь	Имя пользователя.
Пароль	Пароль.
Уровень привилегий	Уровень привилегий.
Повторы	Количество повторов.
Таймаут	Таймаут команд (по умолчанию 1с).

НАСТРОЙКИ ЖУРНАЛА СОБЫТИЙ

Свойство	Описание
Режим	Значения: все события, последние события и диапазон.
Id первого события	Id первого полученного события для режима 'Диапозон'.
Id последнего события	Id последнего полученного события для режима 'Диапозон'.
Последние события	Максимум последних полученных событий для режима 'Последние события'.

Активы Device

Драйвер не поддерживает активы.

Настройки Device

Драйвер устройств IPMI создает три переменные настроек Device:

Имя переменной	Описание переменной	Комментарии
fru	FRU	Содержит следующие атрибуты: • **ID** (`id`) • **Имя свойства** (`propertyName`) • **Значение свойства** (`propertyValue`)
sensors	Сенсоры	Содержит следующие атрибуты: • **ID** (`id`) • **Статус** (`status`) • **Имя** (`name`) • **Тип** (`type`) • **Чтение** (`reading`) • **Unit** (`unit`) • **Нижнее не критичное значение** (`lowerNonCriticalThreshold`)

		• **Верхнее не критичное значение** (upperNonCriticalThreshold) • **Нижнее критичное значение** (lowerCriticalThreshold) • **Верхнее критичное значение** (upperCriticalThreshold)
event log	Журнал событий	• **ID** (id) • **Тип** (type) • **Дата** (date) • **Тип сенсора** (sensorType) • **Подтверждение события** (eventAssertion) • **Событие** (event)

Операции Device

Контекст драйвера устройства 518 IPMI предоставляет следующие функции для управления шасси:

- Включение питания
- Выключение питания
- Полная перезагрузка.

События Device

Драйвер не поддерживает события.

Подключение

Драйвер переводит устройство в режим **онлайн** если:

- TCP подключение к устройству было успешно установлено.

10.2.17 HTTP/HTTPS

Драйвер устройства 518 HTTP/HTTPS позволяет AtomMind Server обмениваться данными с внешними веб серверами через протокол HTTP или HTTPS. Это позволяет формировать различные необработанные запросы (GET, POST, и т.д.) и внедрять ответы (как заголовки, так и содержимое) в ядро платформы для дальнейшей обработки.

Веб-сервер является компьютерной программой, которая доставляет контент, например веб-страницы, в глобальную сеть интернет, используя Протокол передачи гипертекста (HTTP).

HTTP/HTTPS драйвер также контролирует любой из веб-серверов (Apache, IIS и т.д.), проверяя, принимает ли контролируемое устройство HTTP(S) запросы и выдает ли правильные страницы в ответе. Драйвер поддерживает GET и POST запросы, авторизацию и другие опции для HTTP(S) запросов.

Информация о драйвере

ID плагина 518 драйвера : com.tibbo.linkserver.plugin.device.http

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

СВОЙСТВА ПОДКЛЮЧЕНИЯ

Настройки подключения определяют, как AtomMind Server взаимодействует с определенным HTTP сервером. Данные настройки доступны через опцию Изменить свойства устройства[1494] контекста Device. Доступны следующие свойства подключения:

Настройка	**Описание**
Адрес	IP адрес или имя хоста HTTP сервера.
Протокол	HTTP или HTTP Secure (HTTPS).
Версия протокола	Версия 1.1 или 2.0.
Порт	Порт, на котором работает HTTP сервер.
URL	Путь к управляемому ресурсу, расположенному на определенном HTTP сервере.
Метод запроса	В настоящее время поддерживаются следующие методы HTTP запросов: **GET, POST, PUT** и **DELETE**
Данные для выполнения запроса POST	Данные, которые будут представлены с POST запросом в виде строки.
Данные для выполнения запроса PUT	Данные, которые будут представлены с PUT запросом в виде строки.
Кодирование	Устанавливает кодировку содержимого *Данных для выполнения запроса POST* и *Ответа.*
Включить авторизацию	Включает авторизацию (если необходимо).
Имя пользователя	Имя пользователя для авторизации.
Пароль	Пароль для авторизации.
Домен	Домен для NTLM авторизации.
Рабочая станция	Рабочая станция для NTLM авторизации.
Прокси	Адрес HTTP/HTTPS/SOCKS прокси для использования.
Использование сертификата клиента	Позволяет использовать сертификат клиента для аутентификации на сервере, если используется протокол HTTPS.
Файл хранилища ключей	Путь к файлу хранилища ключей, содержащий сертификаты клиента. Это локальный путь файловой системы на машине, где работает AtomMind Server.
Тип хранилища ключей	Тип файла хранилища ключей: JKS или PKCS12.
Пароль хранилища ключей	Пароль, дешифрующий файл хранилища ключей (опционально).
Дополнительные HTTP заголовки	Дополнительная информация в составе заголовка запроса. Представляет собой таблицу с последующими полями: • Имя заголовка • Значение заголовка
Агент	Строка агента пользователя.
Таймаут	Время ожидания выполнения операции HTTP сервера.
Отключить расширение SNI	Отключает расширение Указание Имени Сервера (SNI) протокола TLS.

Активы Device

Драйвер не поддерживает активы.

Настройки Device

HTTP драйвер устройства создает только ту переменную настройки Device, которая показывает результат HTTP запроса. Данная переменная включает следующие поля, анализируемые при помощи различных AtomMind модулей:

Свойство	Описание
Успешно	Указывает, что веб сервер успешно прочитал страницу.
Время отклика, миллисекунд	Время ответа сервера.
Код состояния	Код HTTP ответа.
Заголовки	Таблица заголовков HTTP ответа.
Ответ	HTTP ответ, т.е. содержание веб-страницы.
Ошибка	Текст ошибки или NULL, если ответ был успешным (даже если код HTTP ответа не равен 200/OK).
Бинарные данные состояния	Бинарная версия ответа.

Операции Device

- **Выполнить HTTP запрос**. Эта операция отправляет HTTP запрос GET или POST устройству и возвращает выход. Вводные значения включают URL, метод запроса, данные POST и дополнительные заголовки HTTP. Возвращенные значения - флажок успешного выполнения, время ответа, ответный код HTTP, заголовки, текст возвращенной страницы и текст ошибки.

События Device

Драйвер не представляет события.

Подключение

HTTP устройство находится в режиме **онлайн**, если не было обнаружено ошибок во время подключения и выполнения HTTP запроса, а также если был получен правильный HTTP ответ.

Синхронизация

HTTP монитор подключается к IP хосту, отправляет IP запрос, построенный с помощью специальной конфигурации (порт, URL, тип и данные запроса, HTTP заголовки, агент, время ожидания) и анализирует ответ.

Если в настройках включена аутентификация и необходима базовая аутентификация доступа для доступа к веб-странице, драйвер тоже посылает запрос на аутентификацию.

10.2.18 JMX (Java расширения для сетевого управления)Mana

JMX драйвер устройства 518 позволяет AtomMind Servery взаимодействовать с Java-приложениями и сервером приложений, используя протокол Java расширений для сетевого управления (JMX).

Драйвер выполняет следующие функции:

- Обнаружение MBeans, предоставленных MBean сервером (JMX хост).
- Создание настройки контекста 573 Device для каждого доступного атрибута MBean.
- Создание операции контекста 573 Device для каждой операции MBean.
- Создание определения событий контекста 573 Device для каждого типа уведомлений MBean.
- Выполнение запросов MBean.

Запросы MBean

Запросы MBean повзоляют выбрать несколько MBeans одинакового формата/типа. Каждый запрос выводит результат в таблице, которая в своих колонках показывает все свойства выбранных MBeans, по одному экземпляру MBean в каждой строке.

СИНТАКСИС ЗАПРОСА MBEAN

Текст запроса состоит из двух частей, *домена* и *ключевых свойств*, разделенных символом двоеточия (:).

Домен - это строка символов, не включающая двоеточие (:). Она может включать символ подстановки звездочка (*) или знак вопроса (?). Звездочка соответствует любой последовательности из нуля или более символов, а вопросительный знак соответствует любому одному символу. Если домен является пустым, то он будет заменен в некоторых случаях доменом по умолчанию MBean-сервера.

Ключевые свойства - это неупорядоченный запятой набор ключей и соответствующие значения. Каждый ключ - это непустая строка символов, которая может не содержать символы запятая (,), равенство (=), двоеточие, звездочку или вопросительный знак. Тот же ключ может не произойти дважды в данном запросе.

Список *ключевых свойств* может содержать один элемент звездочка (*), что означает, что MBeans, выбранный запросом, может содержать любое количество других (неуказанных) свойств.

Пример: `java.lang:type=GarbageCollector,*`

Это запрос MBean представляет список всех MBeans в домене `java.lang`, у которых есть свойство `type`, эти значения равны `GarbageCollector`. Такие MBeans могут иметь любое количество других свойств с любыми значениями.

Каждое значение, связанное с ключом, является строкой символов как без кавычек, так и в кавычках.

- Значение без кавычек - это возможная пустая строка символов, которые могут не содержать символов запятая, равно, двоеточие, кавычки, звездочка или знак вопроса.
- Значение в кавычках состоит из кавычек ("), затем возможной пустой строки символов с последующими закрывающими кавычками. Внутри строки символов косая черта влево (\) имеет особое значение. Она должна сопровождаться одним из следующих символов:
  - Вторая косая черта влево. Вторая косая черта влево не имеет особого смысла, и два символа представляют единую косую черту влево.
  - Символ 'n'. Два символа представляют собой перевод строки ('\n').
  - Кавычки. Два символа представляют собой кавычки и не разрывают указанное значение. Закрывающие кавычки в конце должны присутствовать для оформления допустимости значения.
  - Знак вопроса (?) или звездочка (*). Эти два символа представляют собой знак вопроса или звездочку соответственно.

Ещё примеры запросов MBean:

`*:type=Foo,name=Bar` соответствует именам домена, который имеет определенный набор ключей `type=Foo,name=Bar`.

`d:type=Foo,name=Bar,*` соответствуют именам домена `d`, который имеет ключи `type=Foo,name=Bar`, плюс ноль и другие ключи.

`*:type=Foo,name=Bar,*` соответствуют именам доменов, которые имею ключи `type=Foo,name=Bar`,плюс ноль и другие ключи.

`d:type=F?o,name=Bar` соответствует, например, `d:type=Foo,name=Bar` и `d:type=Fro,name=Bar`.

`d:type=F*o,name=Bar` соответствует, например, `d:type=Fo,name=Bar` и `d:type=Frodo,name=Bar`.

`d:type=Foo,name="B*"` соответствует, например, `d:type=Foo,name="Bling"`. Знаки подстановки распознаются даже внутри кавычек и, как другие особые символы, могут быть отделены при помощи `\`.

Кавычки, знак вопроса или звездочка могут не появиться внутри значения в кавычках, кроме как непосредственно после нечетного числа последовательных косых черт влево.

Пробелы не имеют особого значения в запросе MBean. Например, строка домена `key1 = value1 , key2 = value2` представляет запрос с двумя ключами. Название каждого ключа содержит шесть символов, из которых первый и последний - пробелы. Значение, соответствующее ключу "key1" также начинается и заканчивается пробелом.

Никакая часть запроса не может содержать символ новой строки ('\n'), будь то домен, ключ или значение, в кавычках или без. Символ новой строки может быть представлен значением в кавычках с последовательностью \n.

Информация о драйвере

ID плагина драйвера : com.tibbo.linkserver.plugin.device.jmx

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

СВОЙСТВА ПОДКЛЮЧЕНИЯ

Свойства подключения определяют, как AtomMind Server взаимодействует с JMX сервером. Данные настройки доступны, используя опцию Изменить свойства 1494 Device контекста Device. Доступны следующие свойства подключения:

Свойство	Описание
Хост	IP адрес имени хоста, к которому производится подключение.
Порт	Порт подключения.
Имя пользователя	Имя пользователя для аутентификации.
Пароль	Пароль для аутентификации.
URL сервиса пользователя	Пользовательский URL сервиса JMX. Сервис URL уже включает хост и порт, поэтому настройки Хост и Порт не используются, если он указан.
Игнорировать атрибуты с ошибками чтения	Если включено, переменные настроек не будут создаваться для атрибутов MBean, сообщающих об ошибке чтения во время анализа метаданных MBean (например, первая попытка чтения атрибута).

ЗАПРОСЫ MBEAN

Это свойство содержит таюлицу запросы MBean 571:

Свойство	Описание
Имя	Имя переменной настройки Device, содержащей результаты запроса.
Описание	Описание переменной настройки Device, содержащей результаты запроса.
Запрос	Текст запроса.

Активы Device

Драйвер создает один корневой актив для каждого элемента MBean, предоставляемого MBean сервером.

Список MBeans, доступных на хосте JMX, может меняться в любое время. Чтобы запускать просмотр серверного списка MBeans (активы) при каждой синхронизации, поменяйте настройку Режим чтения метаданных 511 учетной записи устройства JMX на **Читать все**.

Настройки Device

JMX драйвер устройства создает одну переменную настройки Device для каждого **атрибута** каждого обнаруженного MBean. Данные переменные группируются по именам MBean и/или описаниям для упрощения навигации в базе данных.

Также одна переменная настройки создается при каждом запросе, определенном в таблице запросов MBean. Эта табличная переменная предлагает список MBeans, отобранных запросом.

Операции Device

JMX драйвер устройства создает одну функцию контекста Device и действие для каждой **операции** каждого обнаруженного MBean. Данные действия группируются по именам MBean и/или описаниям для упрощения навигации в базе данных.

События Device

JMX драйвер создает одно описание события устройства для каждого **типа уведомлений** каждого обнаруженного MBean. Драйвер автоматически добавляет приемник уведомлений для каждого уведомления и при его получении создает событие AtomMind Server.

Подключение

Драйвер переводит устройство в режим **онлайн** если:

- TCP подключение к JMX хосту было успешно установлено.
- Аутентификация и авторизация JMX RMI подключения выполнены успешно.

Синхронизация

Синхронизация между AtomMind Server и JMX хостом включает следующие действия:

- Чтение списка доступных MBeans.
- Чтение информации об атрибуте, предоставляемом каждым MBean.
- Чтение информации об операции, предоставляемой каждым MBean.
- Чтение информации об уведомлениях, предоставляемых каждым MBean.
- Значения чтения/записи всех атрибутов.
- Выполнение всех запросов MBean и чтение атрибутов MBeans, соответствующих этим запросам.
- Подписка на уведомления.

10.2.19 Локальный агент

Драйвер устройства 518 **Локальный агент** превращает AtomMind Server в Agent. Он собирает элементы (переменные/функции/события) с других локальных устройств 497 и/или ресурсов систем и предоставляет эти элементы для других AtomMind Server, подключаясь к ним как Agent.

На стороне другого сервера (получатель данных), коммуникация осуществляется драйвером устройства 523 Agent, который является "пиром" драйвера локального агента.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.agentconnector

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

НАСТРОЙКИ СОЕДИНЕНИЯ

Настройки соединения определяют, как локальный агент взаимодействует с удаленным AtomMind Server:

Свойство	Описание
IP адрес или имя хоста	Адрес удаленного AtomMind Server.
Порт	Номер порта подключения. Для незащищенных соединений, должен соответствовать общей настройке **Номер порта для подключения Агентов** драйвера устройства Agent на удаленном сервере. Для защищенных соединений, должен соответствовать общей настройке **Номер порта для подключения Агентов с использованием** драйвера устройства Agent на удаленном сервере.
Использовать защищенное соединение.	Использовать защищенное SSL соединение к указанным выше адресу и порту.
Использовать сжатие данных	Использовать ZLIB сжатие для обмена данными между Agent и AtomMind Server.

Владелец	Имя пользователя 478, владеющего удаленным аккаунтом устройства Agent.
Имя	Имя удаленного аккаунта устройства Agent.
Пароль	Пароль, определенный в удаленном аккаунте устройства Agent.
Асинхронно отправить обновления	Любые обновления настроек будут доставлены сразу же, нет необходимости ждать следующего цикла синхронизации.
События устройства буфера	В случае перезагрузки AtomMind Server, неудачного соединения или похожих похожих проблем доставки, события могут не удаваться. Буферизация событий будет накапливать недоставленные события и отправит их заново, как только соединение восстановится.
Срок истечения недоставленных событий	Период, после которого недоставленное событие будет стерто из Буфера событий, поэтому не будет отправлено при следующем соединении.
Потенциал очереди рассматриваемых событий	Когда соответствующий показатель событий выше, чем Device может отправить, события отправляются в в специальную Очередь рассматриваемых событий. Ее максимальная длина ограничена количеством.
Максимальное число потоков обработки команд	Максимальное число потоков обработки команд. Параметр используется для ограничения пула выполнения команд Agent.
Отправлять Keepalive сообщения	Отправляются ли Keepalive сообщения.
Таймаут команды	Максимальное время ожидания ответа на команду по протоколу AtomMind.

ЭКСПОРТИРОВАННЫЕ ЭЛЕМЕННТЫ

Данная таблица определяет, какие элементы (переменные/функции/события) будут экспортированы на удаленный AtomMind Server.

Свойство	**Описание**
Маска контекста	Маска контекста, чьи элементы будут предоставлены удаленным аккаунтом устройства Agent.
Группа	Стандартная группа элементов (переменные/функции/события), которые будут предоставлены удаленным аккаунтом устройства Agent. Если вам необходимо экспортировать переменные, добавленные относительной моделью, значение Группы должно следовать формату: `custom|[model_name]|[variable group].`

Активы Device

Драйвер не поддерживает активы.

Настройки Device

Настроечные переменные устройства локального агента предоставлены удаленному серверу. Тем не менее, их локальное редактирование не имеет смысла, так как они попросту отображают переменные экспортируемых локальных контекстов.

Операции Device

Функции устройства локального агента - это функции, предоставленные удаленному серверу. Однако, их локальные звонки не имеют смысла, так как они попросту отображают функции экспортируемых локальных контекстов.

События Device

События устройства локального агента - это события, предоставленные удаленному серверу. Однако, их локальная подписка и обработка не имеют смысла, так как они попросту отображают события экспортируемых локальных контекстов.

Подключение

Драйвер переводит устройство в режим **онлайн** если:

- Установлено TCP подключение к удаленному серверу
- Удаленный сервер успешно авторизовал локального агента, проверив существование удаленного пира Agent, валидность его аккаунта и пароля

Драйвер устройства поддерживает авторегистрацию Agent. Удаленный сервер автоматически создаст аккаунт для этого агента при первом подключении, если включена настройка **Автоматическая регистрация аккаунтов устройства для новых агентов** в общих настройках драйвера устройства Agent на удаленном сервере.

Статус Device

Дополниетльная переменная статуса предоставляется драйвером:

МЕТРИКИ ПРОИЗВОДИТЕЛЬНОСТИ

Свойство	Описание
Количество событий на рассмотрении	Отображает, сколько событий находятся на рассмотрении для отправки в Очереди событий.

Детали синхронизации

Драйвер локального агента не считывает что-либо с удаленного сервера. Вместо этого он отсылает запрошенные значения переменных удаленному серверу, пересылает их обновления, обслуживает входящие звонки предоставленных функций, и пересылает предоставленные события.

10.2.20 Локальный файл

Драйвер устройства 518 **Локальный файл** обеспечивает контроль файлов, расположенных на сервере AtomMind.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.file

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

Мониторинг локального файла может быть сконфигурирован при помощи следующих свойств:

Имя поля	Описание поля
Путь	Определяет местоположение файла (полный путь).
Считывать содержимое	Включает/отключает чтение содержимого файла.
Пошаговое чтение	Переключается с чтения всего файла на пошаговое. Пошаговое чтение идеально подходит для анализа журнала регистрации. Если оно включено, сервер запоминает размер предыдущего файла и при следующем цикле синхронизации читает данные, начиная с последней точки до конца файла. Если размер файла не вырос с предыдущего цикла, чтение не происходит. Если размер файла уменьшился, чтение возобновляется с начала файла (это подходит для управления ротацией журнала регистрации).
Максимальный размер чтения	Максимальное количество байт, которое считывает сервер, если включено пошаговое чтение.

Размер чтения незавершенных заданий	Количество дополнительных байт для включения в пошаговое чтение.
Разрешить редактирование	Позволяет или запрещает изменение содержания файла.
Вычислять контрольную сумму	Включает/отключает вычисление контрольной суммы файла.
Шифровка файла	Определяет шифрование содержимого файла.

Активы Device

Драйвер не поддерживает активы.

Настройки Device

Драйвер устройства локального файла создает две переменные настроек Device:

Имя переменной	Описание переменной	Комментарии
attributes	Атрибуты файла	Содержит атрибуты файла: • **Время последнего изменения** (modificationTime) • **Размер** (size) • **Контрольная сумма**, рассчитываемая по алгоритму Adler-32 (checksum)
contents	Содержание файла	Содержание файла, доступное только для чтения (если включена опция Прочитать содержимое 576) или редактирования (если опция Разрешить редактирование 577 тоже включена).

Операции Device

Драйвер не проводит операции.

События Device

Драйвер не представляет события.

Подключение

Если все операции ввода/вывода файла завершились успешно, локальный файл Device находится в режиме "онлайн".

Если не указан ни один файл в пути или при доступе к файлу произошла ошибка, локальный файл Device считается находящимся в режиме "офлайн".

Синхронизация

Так как локальный файл является драйвером устройства 518, он выполняет "синхронизацию 514" с AtomMind Server, как и любой другой драйвер. Во время синхронизации происходит получение атрибутов и содержания файла. Также, если разрешена правка и пользователь изменил содержание файла в копии сервера, новое содержание записывается в файл.

10.2.21 Локальная папка

Драйвер устройства 518 **Локальная папка** обеспечивает контроль всех папок, расположенных на сервере AtomMind.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.folder

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

Локальная папка проверяет наличие папки, собирает данные об атрибутах папки, загружает и исследует содержание папки (файлы и подпапки). Содержание папки может анализироваться рекурсивно (включая подпапки) или не рекурсивно. Помимо этого, подлежащие обработке объекты фильтруются при помощи знака подстановки или регулярного выражения. Конфигурация осуществляется при использовании следующих свойств:

Имя поля	Описание поля
Путь	Определяет местонахождение папки, т.е. полный путь.
Считывать содержимое	Включает/отключает чтение содержания папки.
Рекурсивно	Если отключено, будет анализироваться содержание только определенной папки. Иначе все ее подпапки буду обработаны рекурсивно.
Следовать символьным ссылкам	Если включено, драйвер будет разрешать и следовать символьным ссылкам при сканировании содержимого папки.
Включаемые файлы	Определяет фильтр для объектов папки. Данная опция повзоляет выбирать, обрабатывать все файлы и папки или же только те, которые удовлетворяют выбранному знаку подстановки или регулярному выражению.

Настройки Device

Драйвер устройства Локальная папка создает две переменные настройки Device:

Имя переменной	Описание переменной	Комментарии
Атрибуты папки	Атрибуты	содержит следующие атрибуты: • **Время последнего изменения** (`modificationTime`) • **Размер**, т.е. общий размер всех файлов, выбранных согласно опциям "Рекурсивно" и "Включение файлов" (`size`) • **Количество файлов**, т.е. общее количество файлов, выбранных согласно опциям "Рекурсивно" и "Включение файлов" (**fileCount**)
Содержание папки	Содержание	Список объектов папки (файлы и подпапки) с данными полями: • **Тип** • **Относительный путь** • **Размер**

Операции Device

ЧТЕНИЕ ФАЙЛОВ

Данная операция используется для чтения содержания файлов. Она принимает список масок файла. Возвращает все файлы, если нет заданных масок. Результатом функции является Таблица данных 49 с двумя полями:

- **Имя файла**. Строковое имя файла.
- **Считывать содержимое**. Содержимое Блока данных 52 файла.

События Device

Драйвер не представляет события.

Подключение

Если все операции ввода/вывода файла завершились успешно, локальная папка Device находится в режиме **Онлайн**.

Если не указан ни один файл в пути или при доступе к папке произошла ошибка, локальный файл Device считается находящимся в режиме **Офлайн**.

Синхронизация

Устройства Локальной папки синхронизируются 514 с AtomMind Server, как и любое другое Device. Во время синхронизации происходит получение атрибутов и содержания папки.

10.2.22 Локальная система

ЛОКАЛЬНАЯ СИСТЕМА

Драйвер устройства 518 **Локальная система** обеспечивает мониторинг параметров системы, расположенных на сервере AtomMind.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.localsystem

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

Не определены.

Активы Device

Драйвер не поддерживает активы.

Настройки Device

Драйвер устройства локальной системы создает три переменные настроек Device:

Имя переменной	Описание переменной	Комментарии
cpu	Процессор	Содержит следующие атрибуты: • **Допустимая нагрузка** (`cpySystemLoad`) • **Число ядер** (`cpuCount`)
disks	Диски	Содержит следующие атрибуты: • **Корень** (`diskRoot`) • **Всего места** (`diskTotalSpace`)

		• **Свободное место** (freeTotalSpace) • **Доступное место** (diskUsableSpace)
memory	Память	Содержит следующие атрибуты: • **Всего физической** (memoryTotalPhysicalSize) • **Осталось физической** (memoryFreePhysicalSize) • **Всего свопа** (swapTotalSpaceSize) • **Осталось свопа** (swapFreeSpaceSize) • **Виртуальная память** (virtualMemoryCommittedSize)

Для 32-разрядных операционных систем значения общей и свободной памяти могут значительно отличаться от ожидаемых. Это связано с тем, что процесс, выполняющийся на такой системе, может использовать максимум 2 Гб памяти.

Операции Device

Драйвер не проводит операции.

События Device

Драйвер не представляет события.

Синхронизация

Devices локальной системы синхронизированы 514 с AtomMind Server как и любой другой Device.

10.2.23 Message Stream

MESSAGE STREAM

Драйвер устройства 518 Message Stream позволяет AtomMind Server взаимодействовать с устройствами, которые отсылают строковые данные, используя TCP, UDP порты или последовательные порты. Как только AtomMind Server подключается к устройству, он начинает прослушивать входящий поток. Этот поток разделен на сообщения специальным разделителем сообщений. Каждый раз, когда доступно новое письмо, оно передается в выражение разбора, где может быть определено любое поведение.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.message-stream

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

НАСТРОЙКИ ПОДКЛЮЧЕНИЯ

Настройки соединения определяют как AtomMind Server взаимодействует с отдельным устройством. Данные настройки доступны через опцию изменить свойства 1494 Device контекста Device. Доступны следующие свойства подключения:

Свойство	Описание
Режим	Выбор TCP, UDP или Последовательного режима.
IP адрес или имя хоста	Адрес устройства (для режима TCP/UDP).
Порт	Порт прослушивания устройства (для режима TCP/UDP) или имя последовательного порта, к которому подключено устройство(для последовального режима).
Скорость передачи информации	Скорость передачи информации (для последовального режима).
Контроль входящего потока	Тип контроля входящего потока: None, CTS/RTS, или XON/XOFF (для последовального режима).
Контроль исходящего потока	Тип контроля исходящего потока: None, CTS/RTS, или XON/XOFF (для последовального режима).
Биты данных	Последовательные биты данных (для последовального режима).
Стоп-биты	Последовательные стоп-биты (для последовального режима).
Четность	Последовательная четность (для последовального режима).
Таймаут	Таймаут команд (по умолчанию 5 секунд).
Разделитель сообщений	Разделитель входящих сообщений
Выражение обработки	Выражение выполнения для входящего сообщения

Активы Device

Драйвер не поддерживает активы.

Настройки Device

Драйвер устройства Message Stream создает две Device переменных настроек:

Имя переменной	Описание переменной	Комментарии
message	Сообщение	Содержит последнее принятое сообщение (без разделителя)
messageStatistics	Статистика сообщений	Содержит следующие атрибуты: • **Количество сообщений** (messageCount) • **Время последнего принятого сообщения** (lastMessageReceivedTime)

Операции Device

Драйвер не проводит операции.

События Device

Драйвер не представляет события.

Подключение

Драйвер переводит устройство в режим **онлайн**, если:

- Было успешно установлено TCP, UDP или последовательное соединение с устройством.

10.2.24 Meter Bus (M-Bus)

Драйвер устройства 518 Meter Bus (M-Bus) позволяет AtomMind Server общаться с умными счётчиками, поддерживающими протокол Meter Bus. Считывание показаний счетчиков и чтение метаданных получены и доступны как переменные настроек устройств 501.

Драйвер Meter Bus поддерживает как последовательную (Serial), так и IP (TCP) связь со счетчиками.

КОНФИГУРАЦИЯ ATOMMIND SERVER ДЛЯ ПОСЛЕДОВАТЕЛЬНОЙ ПЕРЕДАЧИ ДАННЫХ

Обратитесь к разделу Включение последовательного порта[1448], если у вас проблемы с подключением к последовательным устройствам Meter Bus.

Информация о драйвере

ID плагина[5181] драйвера: com.tibbo.linkserver.plugin.device.meterbus

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

НАСТРОЙКИ ПОДКЛЮЧЕНИЯ

Настройки подключения определяют, как AtomMind Server взаимодействует с определенным устройством M-Bus. Данные настройки доступны через опцию изменить свойства[1494] Device контекста Device. Доступны следующие свойства подключения:

Свойство	Описание
Тип подключения	Выбор M-Bus Serial и M-Bus TCP.
IP адрес или имя хоста	Адрес устройства M-Bus (для M-Bus TCP).
Порт	Порт устройства M-Bus (IP номер порта для M-Bus TCP; Имя последовательного порта для M-Bus Serial).
Скорость передачи информации	Скорость передачи информации (для M-Bus Serial).
Биты данных	Последовательные биты данных (для M-Bus Serial).
Стоп-биты	Последовательные стоп-биты (для M-Bus Serial).
Четность	Последовательная четность (для M-Bus Serial).
Адрес счетчика	Адрес счетчика на M-bus.
Использовать вторичный адрес	Определяет, должен ли быть использован вторичный адрес.
Производитель	Производитель счетчика (для вторичного адреса).
Версия	Версия счетчика (для вторичного адреса).
Среднее	Среднее (для вторичного адреса).
Исторические данные процесса	Определяет, должен ли сервер получать и обрабатывать исторические значения, собранные счетчиком. По умолчанию, обрабатываются только текущие (полученные) значения.

Активы Device

Драйвер не поддерживает активы.

Настройки Device

Драйвер устройства M-Bus создает две Device переменных настроек на каждое показание счетчика.

Первая переменная содержит чтение метаданных:

- Код поля данных
- Поле функции (поле управления)
- Номер хранилища
- Тариф

- Тип VIF
- Единица измерения
- Узел
- Описание
- SI префикс
- Экспонента

Вторая переменная отражает фактическое значение счетчика показаний. Значения представлены как строки, так как протокол M-Bus позволяет устройствам выводить нечисловые значения в редких случаях.

Операции Device

Драйвер не проводит операции.

События Device

Драйвер не представляет события.

Подключение

Драйвер переводит устройство в режим **онлайн** если:

- Последовательный порт был успешно открыт (для устройств M-Bus Serial)
- TCP подключение к устройству было успешно установлено (для устройств M-Bus TCP)

Синхронизация

Синхронизация между AtomMind Server и устройством M-Bus включает в себя следующие шаги:

- Чтение метаданных доступных показаний счетчика.
- Чтение и обработка исторических значений показаний если включено **Исторические данные процесса** в настройках устройства.
- Чтение полученных значений показаний счетчика M-Bus и хранение этих значений в кэше настроек.

10.2.25 Modbus

Modbus драйвер устройства 518 позволяет AtomMind Server взаимодействовать с устройствами, используя **Modbus протокол.** Данные устройства могут быть подключены к системе и, подобно всем другим типам устройств, их данные преобразовываются в специальную форму, так что доступ к ним возможен от разных экземпляров AtomMind. Смотрите статью Device для более подробной информации о "нормализованном" представлении устройств в AtomMind.

Modbus драйвер поддерживает сетевые версии (Modbus TCP и Modbus UDP) и серийные версии (Modbus RTU, Modbus ASCII и Modbus BIN) протокола. Серийные устройства Modbus должны быть подключены к COM портам компьютера, запускающего AtomMind Server.

Если множественные устройства Modbus находятся на serial bus (например, на RS-485 bus), все устройства будут соединены с AtomMind путем единственного аккаунта устройства. Регистры, которые принадлежат к разным устройствам, должны быть добавлены к общей таблице **регистров** (см. ниже) с различными значениями **Unit ID**.

Охват устройств Modbus путем сети IP

Есть два основных пути соединения последовательного устройства Modbus к AtomMind по сети IP:

ИСПОЛЬЗОВАНИЕ SERIAL-OVER-IP КОНВЕРТЕРА

В этом случае конвертер изменяет физический протокол (от Serial на Ethernet или Wi-Fi), но не изменяет протокол на уровне приложения. Поэтому данные Modbus ASCII или Modbus RTU передаются по сети IP "как есть". На стороне сервера виртуальный последовательный драйвер порта необходим, чтобы "поймать" трафик IP и предоставить его с помощью классического последовательного порта. На стороне AtomMind Server аккаунт устройства должен быть настроен для того, чтобы работать в режиме Modbus Serial, взаимодействуя путем верхнего виртуального последовательного порта.

ТВЭЛ предлагает широкий дапазон конвертеров Serial-over-IP (такие как DS1206 програмируемые контроллеры запущенной Serial-over-IP микропрограммы), также как и Virtual Serial Port Driver для ОС Windows и Linux.

Сторонние конвертеры, работающие в паре со сторонними виртуальными последовательными портами, также могут быть использованы для преобразований serial-over-IP.

ИСПОЛЬЗОВАНИЕ КОНВЕРТЕРА MODBUS SERIAL В MODBUS TCP

В этом случае конвертер изменяет да протокола: физический (из Serial в Ethernet или Wi-Fi) и на уровне приложения (Modbus ASCII или RTU в Modbus TCP). Таким образом, данные Modbus TCP передаются по сети IP. На стороне сервера драйвер устройства Modbus AtomMind должен быть настроен для работы в режиме Modbus TCP и прямо соединяться с устройством (или несколькими устройствами по bus) с помощью сети IP.

Любой из программируемых контроллеров ТВЭЛ (такие как DS1206), который может запустить приложение конвертер Modbus, выполняющий прикладной и физический протокол преобразования.

Сторонние конвертеры также могут быть использованы для преобразования Modbus Serial в Modbus TCP.

Конфигурация AtomMind Server для серийных коммуникаций

Смотрите включение серийных коммуникаций 1448, если возникли проблемы при подключении к серийным устройствам Modbus.

Информация о драйвере

ID плагина 5181 драйвера: com.tibbo.linkserver.plugin.device.modbus

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

НАСТРОЙКИ ПОДКЛЮЧЕНИЯ

Настройки подключения определяют, как AtomMind Server взаимодействует с определенным устройством Modbus. Данные настройки доступны через опцию Изменить свойства 1494 Device контекста Device. Доступны следующие свойства подключения:

Свойство	Описание
Версия Modbus	Выбор Modbus TCP, Modbus UDP и Modbus Serial.
Идентификатор устройства (Unit ID)	Идентификатор устройства Modbus. Идентификация (адрес) удаленного подчиненного узла, подключенного к серийной линии или другим информационным каналам(например, RS485).
IP адрес или имя хоста	Адрес устройства Modbus (для Modbus TCP/UDP).
Порт	Порт устройства Modbus (номер IP порта, заданный по умолчанию на 502 для Modbus TCP/UDP; имя серийного порта для Modbus Serial).
Кодирование	Выбор RTU, ASCII или BIN (для Modbus Serial).
Скорость передачи информации (Baud Rate)	Скорость передачи информации (для Modbus Serial).
Контроль входящего потока	Тип контроля входящего потока: None, CTS/RTS, или XON/XOFF (для Modbus Serial).
Контроль исходящего потока	Тип контроля исходящего потока: None, CTS/RTS, или XON/XOFF (для Modbus Serial).
Биты данных (Data Bits)	Биты последовательных данных (для Modbus Serial).
Стоп-биты (Stop Bits)	Стоп-биты последовательных данных (для Modbus Serial).
Четность (Parity)	Четность (для Modbus Serial).
Таймаут	Время ожидания выполнения команды (5 секунд по умолчанию).

Повторы	Количество повторов для каждой команды.
Повторное подключение для каждой команды	Заставляет каждую транзакцию Modbus TCP выполняться при отдельном подключении.
Режим записи	Определяет, какие команды протокола Modbus используются для установки значений регистра: • **Авто**. Используются команды Записать один регистр флага и Записать один регистр, если свойство **Размер** в **Регистре устройства** равно одному. В ином случае используются команды Записать несколько регистром флага и Записать несколько регистров. • **Один регистр на команду (Записать Один)**. Для всех записей будут использоваться команды Записать один регистр флага и Записать один регистр. Несколькими командами будут производиться записи **Регистры устройств** с **Размером** больше, чем один. • **Несколько регистров на команду (Записать несколько)**. Даже для одного регистра будут использоваться команды Записать несколько регистров флага и Записать несколько регистров (даже записи **Регистры устройств** с **Размером**, равным одному).

РЕГИСТРЫ УСТРОЙСТВА

Данное свойство содержит список регистров устройства Modbus, которые доступны для и управляются AtomMindom. как только добавлено новое устройство, должен быть сконфигурирован один и более регистр, чтобы данные устройства были доступны системе. Каждый регистр Modbus представлен одной переменной 61 контекста 1494 Device.

Устройства Modbus не создают **метаданных**, поэтому AtomMind Server не может знать о наличии доступных регистров Modbus определенного устройства. Так что необходимо вручную конфигурировать регистры.

Ниже приведен список свойств каждого регистра Modbus:

Свойство	**Описание**
Имя	Имя регистра. Будет использоваться в качестве переменной контекста 1494 Device, необходимой для доступа к регистру, поэтому она состоит только из букв, однозначных чисел и подчеркивания.
Описание	Текстовое описание регистра. Будет использовано в качестве описания переменной Контекста Device.
Тип	Тип регистра Modbus. Возможные значения:**Coil, Discrete Input, Input Register** и **Holding Register**.
Адрес регистра (десятичный)	Адрес регистра Modbus в десятичном формате. Этот адрес не абсолютный адрес регистра Modbus. Он указывает на смещение регистра от начала адресного пространства для регистров выбранного **Типа**. Например, Адресу (смещению) 1 для регистра типа **Регистр хранения** соответствует: • Адрес протокола 40000 • Адрес модели данных 40001
Размер	Количество однотипных регистров, читаемых одной операцией ввода/вывода Modbus и хранимых в переменной контекста AtomMind Server. Может быть полезно прочитать несколько регистров за раз в следующих случаях: • Если они логически представляют массив • Если они логически представляют строку • Если они логически представляют различные части сложных данных, которые должны читаться при помощи одной элементарной операции

	Действительное количество регистров, читаемых при помощи одной операции ввода/вывода - N * M, где • N - количество регистров, необходимых для хранения одного значения AtomMind Server, зависящего от **Типа** и **Формата** • M - это значение параметра **Размер** Например: • если **Тип** - это Входной регистр, **Формат -** 8 байт, плавающий, а **Размер -** 4, сервер читает 32 регистра и представляет их в виде массива в 4 элемента (Таблица данных 49 в одну колонку) с числами с плавающей точкой • если **Тип -** Регистр хранения, **Формат** - Текст переменного размера, а **Размер** - 64, сервер читает 64 регистра в виде Таблицы Данных в одну ячейку, содержащую строчное значение. Эта настройка имеет значение по умолчанию 1. Значение по умолчанию не должно меняться в большинстве случаев.
Unit ID	Modbus Unit ID. Идентификация (адрес) удаленного ведомого, соединенного на последовательной линии или на других шинах (например, RS485).

Вы можете импортировать список регистров из файла (например, файл CSV), используя функцию Импорт 382 компонента Редактор таблицы данных 382

Если вы хотите подключить несколько схожих устройств Modbus к AtomMind, вы можете заполнить таблицу регистров устройства только один раз, затем скопировать ее на другие устройства, используя действие репликации 110.

Активы Device

Драйвер не поддерживает активы.

Настройки Device

Драйвер устройства Modbus создает одну переменную настройки Device для каждого регистра.

ТИП КОНВЕРТАЦИИ

Данная таблица показывает, как Регистры Modbus конвертируются в переменные контекста Device. Заметьте, что количество строк в каждой переменной зависит от значения параметра **Размер**. По умолчанию все переменные имеют одну строку, т.е. скалярны.

Тип регистра	**Формат**	**Формат переменной AtomMind Server**
Дискретный выход (Coil)	*отсутствует*	Читаемый/записываемый, 1 колонка типа Логическое
Дискретный вход (Discrete Input)	*отсутствует*	Только для чтения, 1 колонка типа Логическое
Выходной регистр (Holding Register)	2-байтный Int Unsigned	Читаемый/записываемый, 1 колонка типа Целое
	2-байтный Int Signed	Читаемый/записываемый, 1 колонка типа Целое
	4-байтный Int Unsigned	Читаемый/записываемый, 1 колонка типа Целое
	4-байтный Int Signed	Читаемый/записываемый, 1 колонка типа Целое
	4-байтный Int Unsigned Swapped	Читаемый/записываемый, 1 колонка типа Целое
	4-байтный Int Signed Swapped	Читаемый/записываемый, 1 колонка типа Целое
	4-байтный Float	Читаемый/записываемый, 1 колонка типа Плавающее

	4-байтный Float Swapped	Читаемый/записываемый, 1 колонка типа Плавающее
	8-байтный Int Signed	Читаемый/записываемый, 1 колонка типа Длинное
	8-байтный Int Signed Swapped	Читаемый/записываемый, 1 колонка типа Длинное
	8-байтный Float	Читаемый/записываемый, 1 колонка типа Двойное
	8-байтный Float Swapped	Читаемый/записываемый, 1 колонка типа Двойное
	2-байтный BCD	Читаемый/записываемый, 1 колонка типа Целое
	4-байтный BCD	Читаемый/записываемый, 1 колонка типа Целое
	Символьный	Читаемый/записываемый, 1 колонка типа Строка. Сервер читает количество регистров, указанных в параметре **Размер,** и представляет их как строку, которая всегда имеет длину в 64 символа.
	Строковый	Читаемый/записываемый, 1 колонка типа Строка. Сервер читает количество регистров, указанных в параметре **Размер,** и представляет их как строку. Эта строка разрывается первым регистром со значением ноль.
Входной регистр (Input Register)	Любое	То же, что и для Выходного регистра с одинаковым **Форматом**, но только для чтения.

Операции Device

Драйвер не выполняет операции.

События Device

Драйвер не представляет события.

Подключение

Драйвер переводит устройство в режим **онлайн** если:

- Последовательный порт успешно открыт (для устройств Modbus)
- Установлено TCP соединение с устройством (для Modbus TCP устройств)
- Всегда (для Modbus UDP устройств)

Синхронизация

Синхронизация между AtomMind Server и устройством Modbus включает следующие шаги:

- Создание кэша настроек 502 согласно списку регистров устройства. Каждая переменная используется для доступа к одному регистру устройства Modbus.
- Чтение значений регистра Modbus и их хранение в кэше настроек.

10.2.26 Модем

Драйвер устройства 518 типа "Модем" позволяет AtomMind Server контролировать GSM модем или любой другой модем, основанный на AT наборе команд. Устройства модема часто используются для отправки SMS сообщений в качестве ответов на тревоги 779.

КОНФИГУРАЦИЯ ATOMMIND SERVER ДЛЯ СЕРИЙНЫХ КОММУНИКАЦИЙ

В большинстве случаев модем подключается к последовательному порту оборудования AtomMind Server. Смотрите ссылку Включение последовательного соединения 1448 для получении доступа к устройству через последовательный порт.

Информация о драйвере

ID плагина драйвера: com.tibbo.linkserver.plugin.device.modem

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

НАСТРОЙКИ ПОДКЛЮЧЕНИЯ

Настройки подключения определяют, как AtomMind Server взаимодействует с модемом. Данные настройки доступны через опцию Изменить свойства Device контекста Device. Доступны следующие свойства подключения:

Свойство	Описание
Режим	Выбор TCP, UDP и Serial.
IP адрес или имя хоста	Адрес модема (для TCP/UDP режима).
Порт	Слушающий порт модема (для TCP/UDP режима) или имя серийного порта, к которому подключен модем (для последовательного режима).
Скорость передачи данных (Baud Rate)	Скорость передачи информации (для последовательного режима).
Управление входящим потоком	Тип контроля входящего потока: None, CTS/RTS или XON/XOFF (для последовательного режима).
Управление исходящим потоком	Тип контроля исходящего потока: None, CTS/RTS или XON/XOFF (для последовательного режима).
Биты данных (Data Bits)	Биты последовательных данных (для последовательного режима).
Стоповые биты (Stop Bits)	Стоп-биты последовательных данных (для последовательного режима).
Четность (Parity)	Четность (для последовательного режима).
Таймаут	Время ожидания выполнения команды (для последовательного режима).

КОМАНДЫ

Данное свойство включает список команд, периодически передаваемых модему во время каждого цикла синхронизации. Последний ответ на каждую команду будет доступен в виде переменной контекста Device.

Команды заносятся в таблицу в случае переполнения наиболее часто используемых команд статуса модема.

Ниже приведен список свойств каждой команды модема:

Свойство	Описание
Имя	Имя команды. Будет использовано для названия переменной контекста Device, используемой для получения ответа, поэтому оно может состоять только из букв, однозначных чисел и подчеркивания.
Описание	Текстовое описание команды. Будет использовано в качестве описания переменной контекста Device.
Команда	Текст отправляемой команды (для большинства команд должен присутствовать префикс "AT").

Ожидать стандартного ответа	Если включено, ответ модема -- OK. Если свойство отключено, получение ответа прекращается, как только получен символ "новая строка".

Если вы хотите подключить несколько схожих устройств типа "модем" к AtomMind, вы можете заполнить таблицу команд только один раз, затем скопировать ее на другие устройства, используя действие репликации 110.

Настройки Device

Драйвер устройства типа "модем" создает одну переменную настройки Device для каждой команды из таблицы команд. Переменная содержит текст ответа, полученного от модема.

Операции Device

ОТПРАВКА SMS

Данная операция используется для отправки SMS сообщений через подключенный GSM модем. Сначала пользователю предлагается ввести номер получателя, текст SMS и, по желанию, включить режим Юникод.

Добавьте действие **Послать SMS** в список действий автоматического исправления ошибок 800 для отправки SMS уведомлений при возникновении тревоги.

ВЫПОЛНЕНИЕ ПОЛЬЗОВАТЕЛЬСКОЙ КОМАНДЫ

Позволяет отправлять пользовательскую команду модему.

События Device

SMS

Запускается, когда AtomMind Server получает входящее SMS сообщение.

Имя события sms

Права доступа: Доступно на уровне 486 прав доступа *Наблюдатель*

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Заметки
отправитель	строка	Номер отправителя.
текст	строка	Текст сообщения.

Подключение

Драйвер переводит устройство в режим **онлайн**, если было установлено соединение с модемом, и если был получен ответ OK на две команды: **ATE0** и **AT**.

Синхронизация

Синхронизация между AtomMind Server и устройством модема включает в себя следующие шаги:

- Создание кэша настроек 502 согласно списку команд. Каждая переменная используется для получения ответа на определенную команду модема.
- Выполнение команд модема и их хранение в кэше настроек.

10.2.27 MQTT

Драйвер устройства 518 MQTT позволяет AtomMind Server подключаться к брокеру MQTT и подписываться на темы (topics). Полученные сообщения представлены как события устройства.

MQTT – протокол обмена сообщениями стандарта ISO типа "издатель/подписчик" , используемый в добавок к протоколу TCP/IP.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.mqtt

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

НАСТРОЙКИ ПОДКЛЮЧЕНИЯ

Настройки подключения определяют, как AtomMind Server взаимодействует с MQTT брокером. Данные настройки доступны через опцию Изменить свойства 1494 Device контекста Device. Доступны следующие свойства подключения:

Настройка	Описание
Адрес брокера	Адрес брокера MQTT. Он определен как URI. Поддерживаются два типа подключения: `tcp://` для подключения TCP и `ssl://` для подключения TCP, защищенного SSL/TLS. **Пример 1:** `tcp://localhost:1883` **Пример 2:** `ssl://localhost:8883` Если порт не определен, MQTT будет использовать `1883` для TCP, и `8883` для TCP, защищенного SSL/TLS.
Темы	Таблица **тем** и параметров **Качества услуг (QoS** (Quality of Service)). Темы поддерживают **подстановочные знаки**. Параметр QoS может быть выставлен на 0, 1 или 2. • **QoS 0** показывает, что сообщение должно быть доставлено не более одного раза (ноль или один раз). • **QoS 1** показывает, что сообщение должно быть доставлено не менее одного раза (один или более раз). • **QoS 2** показывает, что сообщение должно быть доставлено только один раз.
Тип сообщения	Тип сообщений. Может быть текстовым сообщением или сообщением с данными.
Имя пользователя	Имя пользователя, которое используется для подключения к брокеру MQTT.
Пароль	Пароль, который используется для подключения к брокеру MQTT.
ID клиента	Идентификатор каждого клиента, устанавливающего соединение с сервером. ID клиента должен быть уникальным для всех клиентов. Используется сервером для хранения данных, которые связаны с клиентом, поэтому важно, чтобы этот параметр оставался неизменным, когда запрашивается соединение с сервером в течение подписок или достоверных сообщений. По умолчанию ID клиента генерируется автоматически.

Clean Session	Устанавливает, должен ли клиент и сервер запоминать состояние при перезапуске и переподключении. Если уставлено на **false,** и клиент и сервер будут сохранять состояние при перезагрузке клиента, сервера и переподключении. Доставка сообщений будет надежной и отвечать указанному QoS, даже при перезапуске клиента, сервера или соединения. Если уставлено на **true,** клиент и сервер не сохранят состояние при перезагрузке клиента, сервера или подключения.
Интервал Keep Alive	Это значение, измеряемое в секундах, определяет максимумальный временной интервал между отправленными или полученными сообщениями. Это позволяет клиенту определить, что сервер больше недоступен, без необходимости ждать таймаута TCP/IP. Клиент обеспечит то, что по крайней мере одно сообщение находится в сети в пределах каждого интервала keep alive. При отсутствии сообщения с данными в пределах временного периода, клиент посылает очень маленькое "пинг" сообщение, которое будет подтверждено сервером. Значение **0** отменяет обработку параметра в клиенте. Значение по умолчанию – **60** секунд.
Таймаут подключения	Устанавливает значение таймаута подключения. Это значение, измеряемое в секундах, определяет максимальный временной интервал, в течение которого клиент будет ожидать установления подключения к серверу MQTT. Таймаут по умолчанию – **30 секунд**. Значение **0** отменяет обработку таймаута. Это означает, что клиент будет ждать до тех пор, пока подключение не установится или будет отклонено.
Версия MQTT	Устанавливает версию MQTT. Действие по умолчанию для подключения с версией 3.1.1, и возвращение к версии 3.1, если предыдущая отказала в работе. Версия 3.1.1 или 3.1 может быть выбрана специфически, без возвращения к предыдущей версии.
Максимальное количество сообщений в процессе передачи	Устанавливает максимальное количество сообщений, которые могут одновременно передаваться по сетевому потоку. Увеличение данного значения займет больше памяти, но может увеличить пропускную способность. Значение по умолчанию - 10.
Использовать SSL/TLS	Позволяет использовать TCP подключение, защищенное SSL/TLS.
Протокол	Имя используемого криптографического протокола (**SSL**, **SSLv2**, **SSLv3**, **TLS**, **TLSv1**, **TLSv1.1** или **TLSv1.2**).
Тип хранилища	Тип файла хранилища ключей: **JKS**, **JCEKS** или **PKCS12**.
Файл хранилища	Путь к файлу хранилища, содержащему сертификаты клиента. Это путь к локальной файловой системе на машине, на которой работает AtomMind Server.
Путь к CA сертификату	Путь к файлу с CA сертификатом.
Путь к сертификату клиента	Путь к файлу с сертификатом клиента.
Пароль хранилища	Пароль для декодирования файла хранилища ключей.
Пароль менеджера ключей	Пароль для восстановления ключей в хранилище.
Включить буфер	Активирует использование офлайнового буфера сообщений для неподтвержденных сообщений.
Размер буфера	Максимальное число сообщений, которые могут храниться в буфере.
Удалять наиболее старые сообщения	Определяет, будут ли удаляться наиболее старые сообщения, когда количество сообщений в буфере достигает максимума. Если это значение false, буду игнорироваться сообщения сверх установленного максимума.

Активы Device

Драйвер не поддерживает активы.

Настройки Device

Драйвер устройства 518 MQTT создает одну переменную настроек Device:

Имя переменной	Описание переменной	Комментарии
messageCount	Количество сообщений	Количество полученных сообщений. Значение обновляется после каждой синхронизации. При переподключении количество становится снова 0.

Операции Device

ОПУБЛИКОВАТЬ ТЕКСТОВОЕ СООБЩЕНИЕ

Отправить текстовое сообщение брокеру.

У формата функции параметров **входа** есть следующие поля:

Имя	Тип	Описание
Topic	Строка	Тема, относительно которой будет опубликовано сообщение.
QoS	Целочисленное	Количество уровня сервиса. Валидные значения 0, 1 или 2.
Retained	Булевое	Должны сохраняться или нет сообщения брокером.
Text Message	Строка	Сообщение.

У функций нет параметров **выхода**.

ОПУБЛИКОВАТЬ СООБЩЕНИЕ ДАННЫХ

Отправить сообщение данных брокеру.

У формата функции параметров **входа** есть следующие поля:

Имя	Тип	Описание
Topic	Строка	Тема, относительно которой будет опубликовано сообщение.
QoS	Целочисленное	Количество уровня сервиса. Валидные значения 0, 1 или 2.
Retained	Булевое	Должны сохраняться или нет сообщения брокером.
Data Message	Блок данных	Сообщение данных.

У функций нет параметров **выхода**.

События Device

СООБЩЕНИЕ

Запускается, когда AtomMind Server получает входящее сообщение MQTT.

Имя события message

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Заметки
topic	Строка	Тема сообщения.
message	Таблица данных	Формат таблицы данных зависит от типа сообщения. • Формат для текстовых сообщений: **Имя поля** / **Тип поля** textMessage / Строка • Формат для текстовых сообщений:

		Имя поля / **Тип поля** dataMessage / Блок данных

Подключение

Данный драйвер переводит устройство в режим **онлайн**, если было установлено соединение с брокером MQTT.

10.2.28 Хост сети

Драйвер устройства **"хост (узел) сети"** разрешает AtomMind Server доступ к одному хосту IP сети для контроля его статуса и выполняемых на нем различных служб/приложений. Следующая таблица представляет доступные службы, общие настройки, свойства конфигурации и результаты мониторинга (представленные в виде настроек устройства):

Служба мониторинга	Свойства конфигурации	Результаты мониторинга	Операции
Пинг 596	Настройки пинга 597	Результаты пинга 597	
SNMP 598	Настройки опроса SNMP 640	Результаты SNMP 642	Операции SNMP 638
Типичный TCP 598	Настройки типичной службы TCP 599	Результаты типичной службы TCP 599	
Типичный UDP 599	Настройки типичной службы UDP 600	Результаты типичной службы UDP 600	
HTTP 600	Настройки HTTP	Результаты HTTP	Операции HTTP 638
DNS 600	Настройки DNS 600	Результаты DNS 601	
POP3 601	Настройки POP3 601	Результаты POP3 602	
IMAP 602	Настройки сервера IMAP 602	Результаты IMAP 603	
SMTP 603	Настройки сервера SMTP 603	Результаты SMTP 603	
Двусторонняя передача E-mail 604	Настройки прохождения электронной почты 604	Результаты прохождения электронной почты 604	
FTP 605	Настройки сервера FTP 607	Результаты FTP 607	
Трассировка 607	Настройки трассировки 608	Результаты трассировки 609	
SSH 609	Настройки SSH 610	Результаты SSH 610	Операции SSH 638
DHCP 611	Настройки DHCP 611	Результаты DHCP 611	
LDAP 611	Настройки LDAP 612	Результаты LDAP 612	
Radius 612	Настройки Radius 613	Результаты Radius 613	
WMI 661	Настройки WMI 661	Смотрите главу WMI 661, раздел **Настройки устройства**	Операции WMI 638

ОБНАРУЖЕНИЕ УСТРОЙСТВ СЕТИ

Драйвер хоста сети поддерживает функцию *обнаружения устройств*. Процесс обнаружения представляет собой сканирование некоторых хостов IP сети, нахождение доступных служб/приложений и создание учетных записей для обнаруженных усройств.

Более подробное описание обнаружения сети можно посмотреть в документации AtomMind Network Manager.

Информация о драйвере

ID плагина 518 драйвера : com.tibbo.linkserver.plugin.device.ip

Общие настройки

Драйвер девайса хост сети предоставляет следующие общие настройки:

КОНФИГУРАЦИЯ

Эта настройка управляет внутрисерверными сервисами, предложенными драйвером хоста сети, с целью разрешить конфигурировать устройство:

- **Включить сервер FTP при загрузке**. Контролирует, внутри ли сервера FTP хосты AtomMind Server.
- **Количество портов FTP**. Порт, который слушает внутренний сервер FTP.
- **Включить сервер SSH при загрузке**. Контролирует, внутри ли сервера SSH хосты AtomMind Server.
- **Количество портов SSH**. Порт, который слушает внутренний сервер SSH.
- **Включить сервер TFTP при загрузке**. Контролирует, внутри ли сервера TFTP хосты AtomMind Server.
- **Количество портов TFTP**. Порт, который слушает внутренний сервер TFTP.

ЧИТАТЬ СКРИПТЫ КОНФИГУРАЦИИ

Таблица настраивает скрипты ожидания 613, которые используются для резервного копирования настроек устройства. Каждый ряд описывает:

- **Имя скрипта**. Это имя используется относительно скрипта, например, из последовательности резервного копирования конфигураций.
- **Скрипт**. Таблица, определяющая скрипт, например, какие данные отправить устройству и что ожидать в ответ.

ПИСАТЬ СКРИПТЫ КОНФИГУРАЦИИ

Таблица настраивает скрипты ожидания 613, которые используются для восстановления настроек устройства. Каждый ряд описывает:

- **Имя скрипта**. Это имя используется относительно скрипта, например, из последовательности восстановления настроек.
- **Скрипт**. Таблица, определяющая скрипт, например, какие данные отправить устройству и что ожидать в ответ.

СКРИПТЫ КОНФИГУРАЦИИ

Эта таблица конфигурирует скрипты, которые используются, чтобы изменить настройку устройства или выполнить контрольные действия. У каждого срипта есть следующие свойства:

- **Имя скрипта**. Это имя используется относительно скрипта.
- **Описание**. Детальное описание цели скрипта.
- **Тип**. *Скрипт* (то есть скрипт ожидания 613) или *Параметеризатор* (метод отправки отдельной динамически построенной команды устройству)
- **Скрипт**. Таблица, определяющая скрипт (если **тип** установлен на *скрипт*), например, какие данные отправить устройству и что ожидать в ответ.
- **Параметеризатор**. Текст исходных данных параметеризатора, определяющий формат парамметров входа скрипта и метод динамически построенной команды для отправки к устройству. См. движок параметеризатора 1444 для получения информации.

Настройки уровня пользователя

Не определены.

Свойства Device

Свойства драйвера хоста определяют параметры конфигурации, используемые AtomMind Network Manager для управления соответствующим хостом сети. Большинство свойств представляют настройки служб, поддерживаемых драйвером хоста сети. Они представлены в столбце **Свойства конфигурации** таблицы служб хоста сети 593.

Другие свойства Device:

- Свойство **Адрес хоста** (`address`). Оно определяет **IP адрес и имя хоста** отслеживаемого хоста сети.
- Свойство **Выражение статуса соединения** (`connectionStatusExpressions`). Определяет, как каждый сервис влияет на онлайн-статус всего хоста сети.

Активы Device

Активы IP хоста подразделяются на две корневые группы: SNMP активы и WMI активы:

- Если SNMP служба включена, актив создается для каждого MIB файла SNMP, поддерживаемого устройством. Поддерживаемые MIB файлы определяются автоматически при опросе агента SNMP.
- Если WMI служба включена, один актив создается для каждого доступного WMI класса.

Настройки Device

Настройки устройства хоста сети представляют собой:

- Статус хоста
- Результаты мониторинга служб данного хоста сети.

Обратитесь к столбцу **Результаты мониторинга** таблицы служб хоста сети 593.

Операции Device

Обратитесь к столбцу **Операции** таблицы служб хоста сети 593.

События Device

Хосты сети способны создавать два различных типа событий: **Прерывания работы службы** (**Service Outages**) (описаны ниже) и **SNMP уведомления** (описаны в главе SNMP драйвер устройства 638).

ПРЕРЫВАНИЕ РАБОТЫ СЛУЖБЫ

Служба создает событие, как только она переходит в режим "онлайн" спустя некоторое время простоя.

Имя события	Потеря работоспособности
Доступ	Для прав доступа уровня 486 *Пользователь*
Записи	1

Событие прерывания работы службы имеет следующий формат:

Имя поля	Тип поля	Примечание
служба	Строка	Служба, работа которой была прервана.
старт	Дата	Дата и время прерывания работы.
продолжительность	Длинное	Продолжительность прерывания работы в секундах.
причина	Строка	Причина прерывания работы.

Подключение

Драйвер Хоста сети имеет настраиваемую процедуру определения статуса Офлайн/Онлайн. Это позволяет установить для учетной записи устройства статус офлайн, если определенный сервис испытывает определяемую пользователем проблему работоспособности.

Настройка определения статуса осуществляется через таблицу **Выражений Статусов Соединений**. Каждая строка этой таблицы описывает сервис или пользовательскую формулу, которая влияет на онлайн статус устройства хоста сети. Таблица имеет три поля:

- **Комментарий**. Это поле представляет собой удобочитаемое для человека описание записи таблицы.
- **Активированное выражение**. Выражение 112, которое определяет, "активна" ли запись, т.е. действительно влияет на статус устройства. Выражение должно возвращать логическое значение (true/false). Если оно возвращает false, эта запись считается неактивной и не влияет на статус соединения хоста сети.

- **Выражение статуса соединения**. Выражение 112, которое определяет текущий "статус соединения" сервиса, т.е., доступен ли сервис и есть ли у него ошибки. Это выражение должно возвращать логическое значение (true/false) или ноль. Если оно возвращает true или false, соответствующий сервис считается в статусе онлайн или офлайн. Если выражение возвращает ноль, статус соединения сервиса считается неизвестным.

Таблица **Выражений статуса соединения** анализируется в конце каждого цикла синхронизации 514. Записи, которые то **Активированное выражение** возвращает как false, пропускаются. После этого статус соединения хоста сети рассчитывается по результатам **Выражения статуса соединения** оставшихся записей:

- Если хотя бы одно **Выражение статуса соединения** возвращается как ноль, статус соединения хоста сети считается **Неизвестным**
- В ином случае, если хотя одно **Выражение статуса соединения** возвращается как false, хост сети считается находящимся в статусе **Офлайн**
- В других случаях хост сети считается находящимся в статусе **Онлайн**

Этот пример показывает, как изменить настройку сервиса HTTP, чтобы установить у хоста сети статус офлайн, если ответный код HTTP не 200 (успешный):

- Комментарий: `HTTP`
- Активированное выражение: `{httpSettings$enabled}`
- Выражение статуса соединения: `hasVariable({.:}, 'http') && {http$successful} && {http$replyCode} != 200`

Статусы отдельных сервисов можно проверить, используя действие Настроить 1495 Device.

Синхронизация

Драйвер хоста устройства использует процедуры синхронизации определенных служб контролируемого устройства. Обратитесь к соответствующему разделу службы для описания синхронизации.

10.2.28.1 Мониторинг при помощи Ping

Пинг-служба позволяет:

- Проверять статус IP хоста путем мониторинга его достижимости/доступности.
- Измерить время двусторонней передачи сигнала.

Она действует, отправляя эхо-запросы **протокола межсетевых управляющих сообщений (ICMP)** контролируемому хосту и собирая статистику ICMP ответов.

Для использования Пинг-службы при работе на Linux требуются `root` права доступа или права доступа к сырому сокету для пользователя ОС, который выполняет приложение AtomMind Server.

Операция

Смотрите подраздел Настройки пинга 597, чтобы узнать список используемых параметров конфигурации.

Пинг-службы выполняют запрос "ping" и добавляют его результат в кэш. Размер кэша определяется конфигурацией службы. При переполнении кэша старые результаты удаляются.

Пинг-служба переходит в режим **Офлайн** в случае невыполнения нескольких последовательных запросов. Количество данных запросов определяется параметром конфигурации **Количество невыполненных эхо-запросов, активирующих статус офлайн**. Иными словами, если любые из последних запросов были выполнены успешно, служба переходит в режим **Онлайн**.

Кэшированные результаты используются для расчета *статистики пинга*: **среднее время между отправкой запроса и получением ответа (average round-trip time), частота потери пакетов (packet loss rate)** и **время последней отправки запроса и получения ответа (last round-trip time).** Смотрите подраздел Результаты мониторинга пинга 597.

Дополнительные адреса Ping

Иногда необходимо отслеживать устройство сети с помощью нескольких IP-адресов. Сервис Ping предоставляет эту функцию путем внедрения опции для указывания одного или более адресов и отслеживая их наряду с адресом, указанным как "первичный" для устройства. Настройки указываются в полях таблицы Настройки Ping 597 *Additional Addresses To Ping*, *Additional Addresses List* и *Additional Addresses Expression*. Результаты предоставляются дочерней таблицей Additional Results в результатах мониторинга Ping 597.

10.2.28.1.1 Настройки Ping

Следующие настройки используются пинг-службой 596:

Свойство	Описание
enabled	Включает/отключает пинг для данного хоста.
Размер данных пакета, в байтах	Размер ICMP пакета данных (не включая заголовки).
Количество эхо-запросов для расчета частоты потери пакета	Для расчета частоты потери пакета AtomMind Server берет несколько последних результатов ping, определенных данной настройкой, и рассчитывает процент невыполненных запросов.
Количество невыполненных эхо-запросов, активирующее статус офлайн	Количество последовательных невыполненных попыток проведения ping, которое переведет сетевой хост в статус подключения 595 **Офлайн**.
timeout, в миллисекундах	Если ответ не получен по истечению данного времени, попытка проведения ping считается невыполненной.
Дополнительные адреса для отслеживания	Включает или отключает Ping дополнительных адресов 596, которые связаны с аккаунтом устройства. Предоставляются три функции: • **Нет** значит, что дополнительные адреса не отслеживаются • **По списку** значит, что список дополнительных адресов для отслеживания находится в таблице Additional Addresses List (показывается, когда опция выбрана) • **По выражению** обозначает список
Список дополнительных адресов	Предоставляет список адресов, чтобы отслеживать единственную таблицу поля String, каждый ряд указывает на дополнительный IP-адрес или на имя хоста.
Выражение дополнительных адресов	Возвращает единственную таблицу поля String, которая содержит список дополнительных IP-адресов имя имена хоста для отслеживания (похоже на таблцу Additional Addresses List, которая описана выше).

10.2.28.1.2 Результаты Ping мониторинга

Приведенная ниже таблица содержит результаты мониторинга ping:

Свойство	Описание
Удачно	Указывает, что по крайней мере один из недавних запросов был выполнен. Количество данных запросов определяется параметром конфигурации 597 **Количество невыполненных эхо-запросов, активирующее статус Офлайн.**
Среднее время между отправкой запроса и получением ответа (Average Round-Trip Time), миллисекунды	Среднее время между отправкой запроса и получением ответа ряда выполненных попыток проведения ping, определяемых свойством **Количество эхо-запросов для расчета потери пакета.**
Частота потери пакета (Packet Loss Rate), %	Процентное соотношение невыполненных попыток проведения ping. Свойство **Количество эхо-запросов для расчета потери пакета** определяет, сколько попыток проведения ping используется для расчета.
Время последней отправки запроса и получения ответа (Last Round Trip Time), миллисекунды	Время последней отправки запроса и получения ответа последней выполненной попытки проведения ping.
Дополнительные результаты	Таблица результатов ping с дополнительными адресами. Такой же формат, как и у Ping Results, не включая поле Additional Results.

10.2.28.2 SNMP мониторинг

Simple Network Management Protocol - простой протокол сетевого управления (SNMP), предоставляемый **драйвером SNMP устройства.** Драйвер устройства хоста сети использует драйвер SNMP и представляет его свойства в качестве службы. Для более подробной информации обратитесь к разделу Драйвер устройства SNMP 637.

10.2.28.3 Мониторинг типовых служб TCP

TCP Служба представляет собой сетевую службу, использующую Протокол управления передачей (TCP) для взаимодействия с клиентами сети.

Драйвер устройства сетевого хоста производит *контроль доступности/работоспособности* для нескольких TCP служб каждого IP хоста.

Синхронизация

Статус типовой TCP службы проверяется во время синхронизации. Служба находится в режиме **Онлайн**, если все настроенные службы/порты также в режиме онлайн.

Проверить статус определенной TCP службы можно при помощи одной из представленных ниже процедур:

- проверка TCP подключения
- мгновенная проверка ответа (без передачи данных)
- передача данных и проверка ответа

Наиболее эффективной является процедура проверки, определяемая свойством **Передача данных,** которое описано ниже.

ПРОВЕРКА ПОДКЛЮЧЕНИЯ

Проверка подключения выполняется, если свойство **Передача данных** не определено, т.е. *null.* В данном случае выполняются следующие операции:

1. Устанавливается TCP соединение с портом устройства.
2. Осуществляется немедленное чтение ответа.

Проверка считается выполненной успешно в случае **установления соединения**. Ответ входит в результат записи данных, но не влияет на статус службы.

МГНОВЕННАЯ ПРОВЕРКА ОТВЕТА

Мгновенная проверка ответа выполняется, если свойство **Передача данных** установлено на *empty string.* Выполняются следующие операции:

1. Устанавливается TCP соединение с портом устройства.
2. Осуществляется немедленное чтение ответа.

Проверка считается выполненной успешно в случае **установления соединения и получения непустого ответа.** Ответ входит в результат записи данных.

ПЕРЕДАЧА ДАННЫХ И ПРОВЕРКА ОТВЕТА

Проверка передачи данных и овета выполняется, если свойство **Передача данных** установлено на *non-empty string.* Выполняются следующие операции:

1. Устанавливается TCP соединение с портом устройства.
2. Через данное подключение отправляются данные, определенные в конфигурации службы.
3. Осуществляется чтение ответа.

Проверка считается выполненной успешно в случае **установления подключения, отправки данных и получения непустого ответа.** Ответ входит в результат записи данных.

10.2.28.3.1 Настройки типовой TCP службы

Приведенные ниже настройки, представленные переменной **TCP сервиса/порта** (tcpSettings), используюся для управления определенной службой/портом (см. также Мониторинг типовой TCP службы)[598]:

Свойство	Описание
Порт	Номер порта, к которому осуществляется подключение.
Таймаут, миллисекунд	Данное время ожидания используется дважды: для установления TCP соединения и ожидания ответа.
Данные для отсылки	Отправка первичных данных в формате строки. Значение по умолчанию - null. Данное значение определяет статус службы. См. Раздел мониторинга типовой TCP службы[598].

10.2.28.3.2 Результаты мониторинга типовой TCP службы

Создается отдельная переменная для каждой контролируемой TCP службы. Данные переменные называются **Типовая TCP служба, Порт** (genericTcp) в соответствии с *номером порта*. Например, переменная службы порта 2345 называется **Типовая TCP служба, порт 2345** (genericTcp2345). Переменные включают следующие поля:

Поле	Описание
Успешно	Указывает, что соединение со службой установлено без ошибок.
Ошибка	Ошибка, обнаруженная в результате установления соединения или передачи данных. В случае отсутствия ошибок значение установлено на NULL.
Таймаут, миллисекунд	Время ответа службы.
Ответ	Ответ, полученный от службы в формате строки. В случае, если ответ не получен или произошла ошибка, значение установлено на NULL.

10.2.28.3.3 Операции типового TCP сервиса

Типовая TCP служба предоставляет единственную операцию: **Мониторинг службы TCP**. Эта операция позволяет выполнять проверку TCP службы с использованием Настроек типовой TCP службы[599]. Операция возвращает результаты[599] проверки.

10.2.28.4 Мониторинг типовых служб UDP

UDP служба представляет собой сетевую службу, которая использует протокол пользовательских датаграмм (UDP) для взаимодействия с ее клиентами.

Драйвер устройства сетевого хоста обеспечивает *контроль доступности/работоспособности* UDP служб разного рода (именно поэтому используется термин "типовая служба"). Конфигурация некоторых UDP служб может быть настроена так, что их мониторинг будет осуществляться через определенный IP хост.

Синхронизация

Мониторинг типового UDP сервиса подобен Мониторингу типового TCP сервиса[599] за исключением, что **UDP** (Протокол пользовательских датаграмм) не требует соединения. Таким образом, процедура мониторинга для определенной UDP службы приурочена к циклу отправки-получения:

1. Данные, определенные конфигурацией службы, отправляются на соответствующий порт IP хоста.
2. Ожидание ответа. Если во время ожидания получены некоторые данные, они вносятся в отчет записи данных.

Данная процедура выполняется для каждой сконфигурированной службы/порта. Служба находится в режиме **Онлайн**, если операции по отправке и получении были выполнены успешно. Вся типовая UDP служба находится в режиме **Онлайн**, если *все* сконфигурированные службы/порты также онлайн.

10.2.28.4.1 Настройки типовой UDP службы

Приведенные ниже свойства, представленные переменной **UDP сервис/порты** (udpSettings), используются для контроля определенной UDP службы (см. также раздел Мониторинг типовой UDP службы)[599]:

Свойство	Описание
Порт	Номер порта, к которому осуществляется подключение.
Таймаут, миллисекунд	Время ожидания ответа.
Данные для отсылки	Отправка первичных данных в формате строки.

10.2.28.4.2 Результаты мониторинга типовой UDP службы

Создается отдельная переменная настроек устройства для каждой контролируемой UDP службы (см. Мониторинг простых TCP служб)[599]. Данные переменные называются **Типовая UDP служба, Порт** (genericUdp) в соответствии с *номером порта*. Например, переменная службы порта 5432 называется **Типовая UDP служба, порт 5432** (genericUdp5432). Переменные включают следующие поля:

Поле	Описание
Порт	Номер порта, на который отправляются данные.
Успешно	Указывает, что ответ был получен.
Ошибка	Ошибка передачи данных или же значение NULL при отсутствии ошибок.
Ответ	Ответ, полученный от службы, в формате строки. В случае, если ответ не получен или произошла ошибка, значение установлено на NULL.
Таймаут, миллисекунд	Время ответа службы.

10.2.28.5 Мониторинг HTTP сервера

HTTP поддержка предоставляется **драйвером устройства HTTP.** Драйвер устройства сетевого хоста использует HTTP драйвер и представляет его свойства в качестве службы. Для более подробной информации смотрите раздел HTTP драйвер устройства[569].

10.2.28.6 Мониторинг DNS сервера

Сервер Системы доменных имён (DNS) предназначен для хранения DNS записей и предоставления ответов на запросы по всей базе данных.

Монитор **Системы доменных имен (DNS)** позволяет контролировать доступность и работоспособность DNS сервера путем отправки поискового запроса и получения ответов с сервера. Собранные ответы представляют собой результаты мониторинга и могут быть использованы для проверки корректности работы сервера DNS.

Детали синхронизации

DNS монитор может быть настроен на отправку нескольких поисковых запросов. Каждый запрос определяет **Тип записи** и **Имя хоста**. Служба выполняет все назначенные поисковые запросы и получает ответные результаты.

Служба находится в режиме **Онлайн**, если все запросы были выполнены успешно.

10.2.28.6.1 Настройки DNS службы

Монитор сервера DNS [600] имеет следующие настройки, представленные переменной **Мониторинг DNS** (dnsSettings):

Свойство	Описание
Активный	Включает/отключает мониторинг сервера DNS.
Порт	Номер порта сервера DNS.

Протокол	Коммуникационный протокол: **UDP** или **TCP**.
Таймаут, миллисекунд	Время ожидания выполнения операций DNS.
Запросы	DNS запросы для выполнения. Свойства запросов: • Тип запроса: **A**, **AAAA**, **CNAME**, **NS** или **MX**. • Запрос: включить имя хоста в запрос.

10.2.28.6.2 Результаты DNS мониторинга

Результат DNS мониторинга представлен переменной **DNS** (`dns`) и включает статус службы и ответы сервера на каждый сформированный поисковый DNS запрос:

Поле	**Описание**
Успешно	Указывает, что все запросы были выполнены успешно.
Детали	Подробные результаты определенных запросов:

Поле	**Описание**
Запрос	Данные запроса (имя хоста), только для чтения.
Тип запроса	Тип запроса, только для чтения.
Результат	Текст ответа сервера DNS (обычно IP адрес или имя хоста). В случае возникновения ошибок устанавливается значение NULL.
Ошибка	Текст ошибки или значение NULL, если во время выполнения запроса не было ошибок.

10.2.28.7 Мониторинг POP3 сервера

Сервер POP является компьютерной программой, которая доставляет e-mail сообщения, используя Протокол почтового отделения (POP). POP3 (POP версия 3) представляет собой ныне действующий стандарт протокола.

Монитор сервера POP3 позволяет проверить доступность и увеличить функциональность e-mail сервера POP3.

Синхронизация

POP3 монитор подключается к заданному почтовому серверу и выполняет аутентификацию, используя параметры, установленные в настройках службы 601. Затем он запрашивает размер папки *INBOX (Входящие)* и находящееся в ней количество сообщений.

При успешном выполнении всех операций служба находится в режиме **Онлайн** и статистика сообщений хранится в качестве результатов мониторинга 602. В случае возникновения ошибки служба переходит в режим **Офлайн**, и текст ошибки сохраняется в результатах мониторинга.

10.2.28.7.1 Настройки сервиса POP3

Конфигурация мониторинга POP3 601 включает в себя следующие свойства, представленные переменной **Настройки POP3** (`pop3Settings`):

Свойство	**Описание**
Активен	Включает/отключает мониторинг сервера POP3.

Порт	Номер порта сервера POP3.
Имя пользователя	Имя пользователя для входа в почтовый ящик.
Пароль	Пароль для входа в почтовый ящик.
Таймаут, миллисекунд	Время ожидания выполнения операций.
Дополнительные настройки	Таблица с названиями и значениями дополнительных настроек загрузки почты. См. доступные настройки здесь 1430.

10.2.28.7.2 Результаты мониторинга POP3

Результат мониторинга хранится в переменной **POP3** (pop3) в следующем формате:

Свойство	Описание
Успешно	Указывает, что во время последней проверки почтового ящика не возникло ошибок.
Количество сообщений	Количество сообщений в почтовом ящике.
Общий размер сообщений в байтах	Общий размер сообщений в почтовом ящике (исходный размер, включая заголовки).
Ошибка	Текст ошибки или значение NULL, если во время выполнения последней проверки не было ошибок.

10.2.28.8 Мониторинг IMAP сервера

Сервер IMAP является компьютерной программой, которая доставляет e-mail сообщения, используя Протокол прикладного уровня для доступа к электронной почте (IMAP).

Монитор сервера IMAP позволяет проверить доступность и функциональность e-mail сервера IMAP.

Синхронизация

Монитор IMAP подключается к серверу IMAP, выполняет аутентификацию и выводит количество сообщений и общий размер всех сообщений определенной папки. Требуемые параметры определяются в качестве настроек службы. 602

Если соединение с сервером IMAP было установлено и получена статистика сообщений, служба переходит в режим **Онлайн**. В противном случае сообщение об ошибке сохраняется и служба переходит в режим **Офлайн**. Для более подробной информации обратитесь к разделу Результаты мониторинга IMAP 603.

10.2.28.8.1 Настройки службы IMAP

Приведенные далее свойства определяют параметры мониторинга сервера IMAP 602, представленные переменной Настройки **IMAP** (imapSettings):

Свойство	Описание
Активен	Включает/отключает мониторинг сервера IMAP.
Порт	Номер порта сервера IMAP.
Таймаут, миллисекунд	Время ожидания выполнения операций.
Имя пользователя	Имя пользователя для входа в почтовый ящик.
Пароль	Пароль для входа в почтовый ящик.
Папка	Папка, которую необходимо проверить (свойство на выбор, папка по умолчанию не выбрана).
Проверять только новые сообщения	Если включено, то в статистике почтового ящика отображаются только непрочитанные сообщения.

Дополнительные настройки	Таблица с названиями и значениями дополнительных настроек загрузки почты. См. доступные настройки здесь 1430.

10.2.28.8.2 Результаты мониторинга IMAP

Результаты мониторинга хранятся в переменной **IMAP** (`imap`) и доступны в качестве настроек устройства в следующем формате:

Свойство	Описание
Успешно	Указывает, что во время последней проверки почтового ящика не возникло ошибок.
Количество сообщений	Количество e-mail сообщений в почтовом ящике или количество непрочитанных сообщений, если установлен флажок **Проверять только последние сообщения.**
Общий размер сообщений, байты	Общий размер сообщений в почтовом ящике (исходный размер, включая заголовки) или общий размер непрочитанных сообщений, если установлен флажок **Проверять только последние сообщения.**
Ошибка	Текст ошибки или значение NULL, если в процессе последней проверки не было ошибок.

10.2.28.9 Мониторинг SMTP сервера

Сервер SMTP является компьютерной программой, которая отправляет и получает e-mail сообщения, используя Простой протокол передачи почты (SMTP). Обычно SMTP используется e-mail приложениями клиента только для отправки сообщений.

Монитор сервера SMTP позволяет проверить доступность и функциональность e-mail сервера SMTP.

Синхронизация

Сервер SMTP контролируется путем установления соединения с сервером и прохождения процедуры авторизации. Отправки e-mail сообщений не происходит. Для более подробной информации о деталях конфигурации обратитесь к разделу Настройка службы SMTP 603.

Если соединение установлено и авторизация прошла успешно, служба находится в режиме **Онлайн.** В противном случае статус меняется на Офлайн и сообщение об ошибке сохраняется в Результатах мониторинга SMTP 603.

10.2.28.9.1 Настройки службы SMTP

Монитор сервера SMTP 603 обладает следующими свойствами конфигурации, представленными переменной **Настройки SMTP** (`smtpSettings`):

Свойство	Описание
Активен	Включает/отключает мониторинг сервера SMTP.
Порт	Номер порта сервера SMTP.
Имя пользователя	Имя пользователя для входа в почтовый ящик.
Пароль	Пароль для входа в почтовый ящик.
Таймаут, миллисекунд	Время ожидания выполнения операций.
Дополнительные настройки	Таблица с названиями и значениями дополнительных настроек отправки почты. См. доступные настройки здесь 1430.

10.2.28.9.2 Результаты мониторинга SMTP

Результаты мониторинга SMTP сохраняются в переменной **SMTP** (`smtp`). Они включают в себя статус режима работы и сообщение об ошибке при наличии таковой.

Свойство	Описание

Успешно	Указывает, что во время последней проверки сервера SMTP ошибки не были обнаружены.
Ошибка	Текст ошибки или значение NULL, если во время последней проверки ошибки не были обнаружены.

10.2.28.10 Служба мониторинга прохождения электронной почты

Служба мониторинга прохождения электронной почты способствует сквозной передаче e-mail сообщений, позволяя тем самым проверить функциональность почтовой системы в целом.

Синхронизация

Сущность данного рода мониторинга заключается в обеспечении успешной доставки e-mail сообщений через контролируемый сервер. Для этого монитор двусторонней передачи данных создает и отправляет тестовое сообщение. Во время следующей синхронизации служба проверяет наличие тестового сообщения во входящих сообщениях почтового ящика. Все тестовые сообщения после проверки немедленно удаляются из почтового ящика. Операции отправки/приема выполняются во время каждой синхронизации.

Адреса отправителя и получателя и используемый протокол получения e-mail сообщений определяются в качестве настроек 604 службы.

Монитор двусторонней передачи e-mail сообщений использует настройки службы SMTP 603 для отправки исходящих тестовых сообщений. Отправитель и получатель данных сообщений определяются в Конфигурации службы двусторонней передачи e-mail сообщений 604.

Пользователь может выбрать, какой протокол использовать (POP3 или IMAP) для получения ранее отосланных сообщений.

Монитор двусторонней передачи e-mail сообщений использует настройки службы POP3 601 или настройки службы IMAP 602 для возврата тестовых сообщений.

Служба двусторонней передачи e-mail сообщений находится в режиме **Онлайн**, если тестовые сообщения были разосланы и получены. Если во время выполнения операций отправки/приема обнаружена ошибка или ранее отосланные тестовые сообщения не поступили в почтовый ящик, статус службы меняется на **Офлайн**. Ошибки, обнаруженные во время операций отправки/приема, сохраняются в результатах мониторинга 604.

10.2.28.10.1 Настройки службы прохождения электронной почты

Мониторинг прохождения электронной почты 604 устанавливается при помощи следующих свойств, определяемых переменной **Настройки прохождения электронной почты** (`emailSettings`):

Свойство	**Описание**
Включено	Включает/отключает мониторинг прохождения электронной почты.
Протокол получения e-mail сообщений	Протокол для управления входящими сообщениями.
Адрес отправителя	E-mail адрес, указанный в поле FROM (От кого) тестового e-mail сообщения.
Адрес получателя	E-mail адрес получателя, которому было отправлено тестовое сообщение.

Монитор прохождения электронной почты использует опции SMTP 603, POP3 601 или IMAP 602 для отправки и получения сообщений.

10.2.28.10.2 Результаты мониторинга прохождения электронной почты

Результаты мониторинга прохождения электронной почты сохраняются в переменной **Прохождение электронной почты** (`emailRoundTripResult`) следующего формата:

Свойство	**Описание**

Успешно	Указывает, что во время последней проверки сервера SMTP не было обнаружено ошибок.
Ошибка при отправлении	Текст ошибки при отправлении или значение NULL при отсутствии ошибок.
Ошибка при получении	Текст ошибки при получении или значение NULL при отсутствии ошибок.

10.2.28.11 Мониторинг FTP сервера

Протокол передачи файлов (FTP) является стандартным сетевым протоколом, используемым для копирования файла с одного хоста на другой через TCP/IP сеть, такую как Интернет.

Монитор сервера FTP выполняет проверку корректной работы сервера FTP путем проверки статуса сервера и чтения атрибутов удаленного файла, используя FTP протокол.

Синхронизация

Настройки конфигурации мониторинга FTP описаны в подразделе Настройки службы FTP 607.

Во время синхронизации выполняются следующие операции:

- Установление соединения с сервером FTP.
- Проведение аутентификации.
- Получение информации о статусе сервера и ее сохранение в результатах мониторинга.
- Определение местонахождения файла с использованием заданного в параметрах пути.
- Получение атрибутов, таких как временная метка файла (обычно время последнего изменения) и размер (в байтах).
- Отключение соединения.

Все данные операции должны быть выполнены успешно для перевода сервера в режим **Онлайн**. Статус сервера и информация о файле хранятся в результатах мониторинга 607.

При невыполнении какой-либо операции статус сервера меняется на **Офлайн**. Ошибка, обнаруженная во время синхронизации, сохраняется в результатах мониторинга.

Имейте в виду, что путь, указанный в конфигурации службы, должен относиться к обычному файлу (т.е. не указывать на директорию или не включать специальных символов). Если монитору не удается обнаружить файл, используя заданный путь, служба FTP переходит в **Офлайн**.

Операции

ПОЛУЧИТЬ FTP

Получить файл из удаленного компьютера.

У формата параметров функции **входа** есть следующие поля:

Имя	Тип	Описание
Порт	Целочисленное	Номер порта, на котором запущен сервер FTP.
Имя пользователя	Строка	Имя пользователя для аутентификации.
Пароль	Строка	Пароль для аутентификации.
Режим	Булевое	Режим FTP: активный или пассивный (по умолчанию).
Путь	Строка	Путь файла или папки для мониторинга (опционально).
Кодировка файла	Строка	Определяет кодировку содержимого файла.

Таймаут	Длинное	Таймаут операций сервера FTP.

Адрес устройства хоста сервера используется в качестве адреса сервера FTP.

У формата параметров функции **выход** есть следующие поля:

Имя	**Тип**	**Описание**
Удачное	Булевое	Операция была выполнена успешно.
Ошибка	Строка	Текст сообщения об ошибке, если ошибка произошла в течение выполнения операции.
Результат	Блок данных	Запрашиваемый файл из удаленного компьютера.

СПИСОК FTP

Файлы списков удаленно соединенных компьютеров.

У формата параметров функции **вход** есть следующие поля:

Имя	**Тип**	**Описание**
Порт	Целочисленное	Номер порта, на котором запущен сервер FTP.
Имя пользователя	Строка	Имя пользователя для аутентификации.
Пароль	Строка	Пароль для аутентификации.
Режим	Булевое	Режим FTP: активный или пассивный (по умолчанию).
Путь	Строка	Путь файла или папки для мониторинга (опционально).
Кодировка файла	Строка	Определяет кодировку содержимого файла.
Таймаут	Длинное	Таймаут операций сервера FTP.

Адрес устройства хоста сервера используется в качестве адреса сервера FTP.

У формата параметров функции **выход** есть следующие поля:

Имя	**Тип**	**Описание**
Удачное	Булевое	Операция была выполнена успешно.
Ошибка	Строка	Текст сообщения об ошибке, если ошибка произошла в течение выполнения операции.
Результат	Блок данных	Запрашиваемый список файлов из удаленного компьютера.

10.2.28.11.1 Настройки службы FTP

Конфигурация монитора сервера FTP 605 состоит из следующих параметров, представленных переменной **Настройки FTP** (`ftpSettings`):

Свойство	Описание
Активен	Включает/отключает мониторинг сервера FTP.
Порт	Номер порта сервера FTP.
Имя пользователя	Имя пользователя для аутентификации.
Пароль	Пароль для аутентификации.
Режим	Режим FTP: активный или пассивный (по умолчанию).
Активный минимальный порт	Минимальное число портов, которое будет назначено соединению данных при использовании режима Активный.
Активный максимальный порт	Максимальное число портов, которое будет назначено соединению данных при использовании режима Активный.
Путь	Путь к контролируемым файлам и папкам (дополнительно).
Содержимое чтения	Определяет, должно ли читаться содержимое данного файла/папки.
Кодировка файла	Определяет кодировку содержимого файла.
Пошаговое чтение	Переключается на пошаговое чтение с чтения всего файла. Пошаговое чтение идеально подходит для анализа журнала регистрации. Если оно включено, сервер запоминает размер предыдущего файла и во время следующего цикла синхронизации читает данные, начиная с последней точки и до конца файла. Если размер файла не вырос с предыдущего цикла, чтение не осуществляется. Если размер файла уменьшился, чтение возобновляется с начала файла (это подходит для управления ротацией журнала регистрации).
Максимальный размер чтения	Максимальное количество байтов, читаемых сервером, если включено пошаговое чтение.
Размер чтения очереди сообщений	Количество байт, добавляемых с предыдущих этапов пошагового чтения.
Таймаут, миллисекунд	Время ожидания выполнения операций сервера FTP.

10.2.28.11.2 Результаты FTP мониторинга

Результаты FTP мониторинга доступны в качестве переменной **FTP** (`ftp`):

Свойство	Описание
Успешно	Указывает, что монитор сервера FTP не обнаружил ошибки во время последнего чтения файла.
Статус сервера	Ответ о статусе сервера FTP.
Содержимое папки	Таблица, представляющая атрибуты контролируемого файла или всех файов в контролируемой папке: имя файла, размер и время последнего изменения.
Содержимое файла	Содержимое контролируемого файла.
Ошибка	Описание ошибки или значение NULL при отсутствии ошибок.

10.2.28.12 Мониторинг при помощи трассировки

Трассировка позволяет отследить маршрут, по которому передаются пакеты данных внутри IP сети.

Служба мониторинга трассировки обеспечивает функциональность трассировки для проверки доступности/достижимости хоста IP сети. Это способствует обнаружению и анализу различных сетевых проблем.

Монитор трассировки использует оба известных метода выполнения трассировки: использование эхо-запросов ICMP и датаграмм UDP. Отслеживаемый во время мониторинга маршрут сети доступен для анализа любым экземплярам AtomMind, обрабатывающим данные. Например, невозможно запустить тревогу, если количество транзитных участков (хопов) превысило предельную величину или транзитный участок N имеет определенный IP адрес.

Для использования трассировки при работе на Linux требуются `root` права доступа или права доступа к сырому сокету для пользователя ОС, который выполняет приложение AtomMind Server.

Синхронизация

Синхронизация службы трассировки включает в себя следующие шаги:

1. Основная процедура трассировки осуществляется путем отправки ICMP или UDP датаграмм от локального целевому хосту. Сначала устанавливается значение 1 для TTL поля датаграммы, которое впоследствии будет увеличиваться. Таким образом собирается информация о промежуточных сетевых узлах (участках). Процесс останавливается, если целевой хост достигнут или обнаружено максимальное количество транзитных участков. Процедура трассировки использует параметры, определяемые в качестве настроек службы 608.
2. Собранная информация обрабатывается и анализируется, формируя тем самым результаты мониторинга 609.

Служба трассировки находится в режиме **Онлайн**, при успешном выполнении обоих шагов и достижении целевого хоста с надлежащим количеством участков. В противном случае служба переходит в режим **Офлайн** и сообщение об ошибке сохраняется в результатах мониторинга.

Настройка файервола для включения отслеживания маршрута

Некоторые операционные системы могут потребовать дополнительную конфигурацию, чтобы позволить AtomMind Server получать входящие ICMP сообщения, используемые ICMP трассировкой.

Например, для ОС Windows Vista/Windows 7 необходимо выполнить следующие действия:

- Перейти в **Пуск > Панель управления > Администрирование > Брандмауэр Windows в режиме повышенной безопасности**
- Выберите слева **Правила для входящих подключений**
- Выберите справа **Создать новое правило**
- Выберите **Для порта** и нажмите Далее.
- Выберите **TCP** и **Все локальные порты,** затем нажмите Далее
- Нажмите Далее в диалоговых окнах Действие и Профиль (необходимо выбрать все профили).
- Введите имя фильтра и нажмите ОК.
- Кликните два раза по только что созданному правилу в главном окне.
- Перейдите на вкладку **Протоколы и порты** и выберите **ICMPv4** в поле со списком **Тип протокола.**
- Нажмите ОК, чтобы закрыть диалоговое окно.

10.2.28.12.1 Настройки службы трассировки

Приведенные ниже настройки службы трассировки, представленные переменной **Настройки трассировки** (`tracerouteSettings`), используются для мониторинга трассировки 607:

Свойство	Описание
Активен	Включает/отключает трассировку.
Протокол	Переключает между ICMP (стандартный метод трассировки ICMP пакетов) и UDP (алгоритм трассировки Вана Якобсона UDP пакетов) методами трассировки.
Максимальное количество пакетов на прыжок	Количество повторных попыток соединения с каждым транзитным участком.
Максимальное количество прыжков	Если целевой хост не был достигнут в пределах данного количества транзитных участков, трассировка прерывается.

Таймаут, миллисекунд	Время ожидания передачи каждого пакета трассировки.

10.2.28.12.2 Результаты мониторинга трассировки

Результаты мониторинга трассировки представлены переменной **Трассировка** (`traceroute`). Она включает в себя следующие поля:

Поле	Описание
Успешно	Указывает, что процедура трассировки проведена успешно и целевой хост был достигнут.
Ошибка	Ошибка, обнаруженная во время отслеживания маршрута или обработки результатов, если есть.
Детали трассировки	Подробные результаты трассировки (т.е. маршрут к хосту). Для каждого участка представляется следующая информация:

Поле	Описание
IP адрес	IP адрес транзитного участка хоста.
Имя хоста	Имя хоста транзитного участка, если определено при помощи обратного DNS.
Время ответа, миллисекунд	Время ответа транзитного участка.
Ошибка	Ошибка, обнаруженная во время соединения с транзитным участком.

10.2.28.13 Мониторинг SSH сервера

SSH сервер представляет собой компьютерную программу, которая принимает соединения от клиентских систем и позволяет обмен данными между двумя сетевыми устройствами через безопасный канал с помощью Secure Shell (SSH) протокола. SSH используется в основном для доступа к учетным записям Shell в качестве замены для Telnet и других небезопасных удаленных подключений.

SSH монитор обеспечивает доступность и работоспособность мониторинга серверов SSH. Служба также предоставляет возможность удаленного выполнения сценария. Это означает, служба мониторинга SSH может быть использована для выполнения различных административных задач на удаленных системах.

Синхронизация

SSH монитор выполняет следующие операции во время синхронизации:

1. Устанавливает соединение с сервером SSH.
2. Аутентификация с использованием имени и пароля, указанными в настройках службы 610.
3. [Дополнительно] Загружает сценарий для удаленного хоста. Скрипт обозначен в поле **Скрипт** настройки в виде явного текста или файла. Монитор сначала пытается загрузить сценарий с помощью scp, если попытка не была выполнена, он создает файл с использованием эхо команд.
4. [Дополнительно] Выполняет загрузку скрипта, используя sh команды. Выход выполнения сценария будет перехвачен и сохранен в результатах мониторинга 610.
5. Закрывает соединение.

Если все операции выполняются успешно, служба SSH находится в режиме **Онлайн.** Если ни один сценарий не указан в параметрах службы (т.е. сценарий поле NULL), пункты 3 и 4 можно исключить; таким образом, выполняются только операции соединения, аутентификации и прекращения соединения.

Если произошла ошибка во время выполнения, служба переходит в статус **Офлайн** и сообщение об ошибке сохраняется в поле **Ошибка (Error).**

Выполнение удаленного сценария

Кроме того, действие 87 **Выполнение скрипта на удаленном сервере, используя SSH** (`executeRemoteScript`) устанавливается в корневой контекст 1559. Он принимает настройки соединения и скрипт в качестве параметров входа:

Свойство	Имя	Описание
Адрес	address	Адрес хоста.
Порт	port	Номер порта сервера SSH.
Таймаут, миллисекунд	timeout	Время ожидания выполнения SSH операций.
Имя пользователя	userName	Имя пользователя для авторизации.
Пароль	password	Пароль для авторизации.
Скрипт	script	Скрипт, указанный в виде явного текста или в виде файла; может быть NULL.

Результаты выполнения скрипта представляются в следующем формате:

Свойство	Имя	Описание
Успешно	successful	Указывает, что никакой ошибки не произошло во время подключения к серверу SSH, авторизации и выполнения удаленного скрипта.
Ошибка	error	Текст ошибки при наличии таковой.
Результат	result	Выход удаленного скрипта.

10.2.28.13.1 Настройки службы SSH

Приведенные ниже настройки, представленные переменной Настройки **SSH** (`sshSettings`), используются для управления SSH 609 сервером:

Свойство	Описание
Активен	Включает / отключает мониторинг SSH и удаленное выполнение скрипта.
Порт	Номер порта сервера SSH.
Имя пользователя	Имя пользователя для авторизации.
Пароль	Пароль для авторизации.
Скрипт	Скрипт, определяемый в виде текста или файла; может быть значение NULL.
Таймаут, миллисекунд	Время ожидания выполнения операций.
Частный ключ RSA или DSA	Частный ключ для авторизации. Ключ должен вводиться в текстовом формате. Например, ключ RSA обычно начинается с префикса `-----BEGIN RSA PRIVATE KEY-----` и оканчивается суффиксом `-----END RSA PRIVATE KEY-----`.
Пароль частного ключа	Дешифрование пароля частного ключа.

10.2.28.13.2 Результаты мониторинга SSH

Результаты мониторинга SSH представлены переменной **SSH** (`ssh`) в следующем формате:

Свойство	Описание
Успешно	Указывает, что ошибки не были обнаружены во время подключения к серверу SSH, авторизации и выполнения удаленного скрипта.
Ошибка	Текст ошибки при наличии таковой.
Результат	Выход удаленного скрипта.

10.2.28.13.3 Операции SSH

Сервис SSH предоставляет одну операцию: **Выполнить скрипт через SSH**. Эта операция позволяет загружать файл скрипта, соединяться с устройством и авторизовываться с использованием настроек SSH[610], выполнять скрипт на удаленном хосте и возвращать его выход.

10.2.28.14 Мониторинг DHCP

DHCP-сервер представляет собой компьютерную программу, которая предоставляет службу автоматической конфигурации IP-устройств сети с использованием протокола динамической конфигурации узла (DHCP).

Служба мониторинга DHCP позволяет проверять доступность и работоспособность сервера DHCP (протокол динамической конфигурации узла).

Синхронизация

DHCP монитор посылает DHCPDISCOVER сообщение контролируемому серверу и ждет ответа. Сообщение содержит MAC-адрес, указанный в настройках конфигурации DHCP[611]. Если ответ успешно получен в течение заданного времени ожидания, служба переходит в режим **Онлайн**. Ответ IP-адреса сервера хранится в результатах мониторинга[611]. В противном случае служба переходит в режим **Offline** и сообщение об ошибке сохраняется в результатах.

10.2.28.14.1 Настройки службы DHCP

Монитор DHCP[611] использует следующие настройки, представленные переменной **Настройки DHCP** (`dhcpSettings`):

Свойство	Описание
Активен	Включает/отключает мониторинг DHCP.
MAC адрес	MAC-адрес устройства клиента, используемый в запросе. (Он должен быть MAC-адресом сервера AtomMind в большинстве случаев).
Таймаут, миллисекунд	Время ожидания ответа службы.

10.2.28.14.2 Результаты мониторинга DHCP

Результаты мониторинга DHCP представлены переменной **DHCP** (`dhcp`) следующего формата:

Свойство	Описание
Успешно	Указывает, что запрос DHCP был успешно отправлен и получен правильный ответ.
Ошибка	Текст ошибки при наличии таковой.
Ответ	IP адрес в ответе сервера (значение в поле `yiaddr` DHCP сообщения - обратитесь к RFC 2131 для более подробной информации).

10.2.28.15 Мониторинг LDAP

Служба каталога является системой службой, которая хранит, организует и обеспечивает доступ к каталогам. **Сервер LDAP** является элементом службы каталогов, которая обеспечивает доступ к части каталога, используя LDAP протокол. Облегчённый протокол доступа к каталогам (LDAP) является протоколом уровня приложения для запросов и изменения данных из службы каталогов.

Мониторинг сервера LDAP обеспечивает доступность и оперативность мониторинга служб каталогов (в том числе каталогов Microsoft Active Directory).

Синхронизация

Монитор LDAP выполняет следующие процедуры синхронизации:

1. Установление соединения с сервером с использованием заданных настроек.[612]

2. Построение поискового запроса при использовании *имени контекста* и параметров *фильтра* и его выполнение.

Пример: Поиск с **Именем контекста**, заданным на "cn=Users,dc=host,dc=domain,dc=com" в сервере *Microsoft Active Directory* должен вернуть дочерние узлы узла "Users", т.е. пользователей, зарегестрированных на `host.domain.com`. Добавление `"cn=SQL*"` в качестве **Фильтра запроса** ограничит запрос, заданный для пользователей с именем, имеющим в начале `"SQL"`.

3. Закрытие соединения.

Служба переходит в режим **Онлайн**, если все операции были выполнены успешно; результаты поиска сохранены. Обнаруженная ошибка сохраняется в результатах мониторинга, и служба переходит в режим **Офлайн**.

10.2.28.15.1 Настройки службы LDAP

Монитор LDAP 611 использует следующие настройки, представленные переменной **Настройки LDAP** (`ldapSettings`):

Свойство	Описание
Активен	Включает/отключает мониторинг LDAP.
Порт	Номер порта, к которому может быть подключен сервер LDAP.
Имя пользователя	Имя пользователя для авторизации. Обычно включает имя домена, т.е. `user@domain.com`.
Пароль	Пароль для авторизации.
Имя контекста выполнения LDAP запроса	Определяемое имя контекста поиска.
Фильтр LDAP запроса	Выражение фильтра, используемое для поиска (согласно RFC 2254). Значение может быть null или empty, в этом случае оно просто игнорируется поиском.

10.2.28.15.2 Результаты мониторинга LDAP

LDAP монитор сохраняет результаты поиска в переменной **LDAP** (`ldap`) в виде таблицы с данными полями:

Свойство	Описание
Успешно	Указывает, что поиски LDAP были выполнены успешно.
Ошибка	Текст ошибки при наличии таковой.
Результаты запроса LDAP	Таблица, представляющая все строковые значения, полученные от сервера в качестве результата запроса LDAP.

10.2.28.15.3 Операции LDAP

Служба LDAP предоставляет единственную операцию: **Выполнение LDAP запроса**. Данная операция выполняет запрос с использование LDAP контекста и фильтра, указанных в параметрах операции и в других настройках LDAP 612, и возвращает результаты 612 запроса.

10.2.28.16 Radius мониторинг

Служба входящих подключений с удалённой аутентификацией (RADIUS) является протоколом, обеспечивающим централизованную аутентификацию, авторизацию и управление учетными записями компьютеров для подключения и использования сетевой службы. **RADIUS** сервер является компьютерной программой, которая может проводить аутентификацию, авторизацию и учет пользователей устройств для доступа к сети и сетевым службам.

RADIUS монитор позволяет проверить доступность и базовую работоспособность RADIUS сервера через прохождение процедуры аутентификации.

Синхронизация

Монитор RADIUS выполняет процедуру аутентификации (см. RFC 2138), используя параметры, заданные в настройках конфигурации 613: формируется *Access-Request* и отправляется на контролируемый хост. Если по истечению определенного времени получен в ответ пакет данных типа `Access-Accept`, служба переходит в статус **Онлайн**. При возникновении ошибки она сохраняется в результатах мониторинга и служба переходит в статус **Офлайн**.

10.2.28.16.1 Настройки службы Radius

Служба Radius 612 использует следующие настройки, представленные **настройками Radius** (`radiusSettings`):

Свойство	Описание
Активный	Включает/отключает Radius мониторинг.
Порт для аутентификации	Номер порта, на который отправляется запрос аутентификации.
Имя пользователя	Имя пользователя для аутентификации.
Пароль	Пароль аутентификации.
Секретный ключ Radius	Секретный ключ, используемый протоколом RADIUS для скрытия паролей.
Таймаут, миллисекунд	Время ожидания ответа службы.

10.2.28.16.2 Результаты мониторинга Radius

Следующие поля используются для хранения результатов мониторинга RADIUS, представленные переменной **Radius** (`radius`):

Свойство	Описание
Успешно	Указывает, что LDAP поиск был выполнен успешно.
Ошибка	Текст ошибки при наличии таковой.

10.2.28.17 WMI мониторинг

Поддержка **WMI** обеспечивается **драйвером устройства WMI**. Драйвер устройства сетевого хоста использует WMI драйвер и представляет его функции в качестве службы. Для более подробной информации обратитесь к разделу Драйвер устройства WMI 661.

10.2.28.18 Expect-скрипты

Expect-скрипты (скрипты ожидания) определяют серию команд, которые могут отправляться устройствам и ожидать от них ответа, контролирующего поведение скрипта. Для выполнения Expect-скрипта, AtomMind Server подключается к устройству при помощи протоколов Telnet или SSH.

Expect-скрипты делятся на три группы:

- **Скрипты чтения конфигурации**. Эти скрипты вызываются только из последовательностей резервирования конфигураций устройств.
- **Скрипты записи конфигурации**. Эти скрипты вызываются только из последовательностей восстановления конфигураций устройств.
- **Скрипты конфигурации**. Это скрипты общего назначения, используемые для изменения конфигураций устройств или контролирования устройств.

Скрипты управляются через настройки глобальной конфигурации 593 драйвера устройства Хоста Сети.

Структура скрипта

Каждый Expect-скрипт настраивается как таблица. Запись в этой таблице определяет команду для отправки устройству и ожидаемый выход устройства.

Каждая запись скрипта содержит:

- **Ожидать**. Это Выражение AtomMind 112, которое должно иметь результатом строку. Эта строка будет являться регулярным выражением 2142, которому должен соответствовать выход устройства. Регулярное выражение "завернуто" в Выражение AtomMind, чтобы дать возможность динамического построения регулярного выражения, зависящего от среды.

Чтобы поместить статическое ругулярное выражение в поле Expect, заключите его в одинарные или двойные кавычки, но не вставляйте какие-либо кавычки или обратные косые черты в само выражение.

Пример: `"(?s).*\\s*(([Uu]sername: )|([Ll]ogin.*: ))\\z"`

Это значение поля **Ожидать** будет иметь результатом следующее регулярное выражение: `(?s).*\s*(([Uu]sername: )|([Ll]ogin.*: ))\z`

Это регулярное выражение, в свою очередь, будет соответствовать приглашениям ввода `Login:` или `Username:`, отправленным устройством. Таким образом, это поле Ожидать соответствует тому, чтобы ожидать приглашение о вводе имени пользователя устройства.

- **Отправить**. Это Выражение AtomMind 112, которое должно иметь результатом строку. Эта строка будет отправлена устройству, когда от устройства приходит текст, соответствующий полю **Ожидать**.

Пример: `{username}`

Это значение поля **Отправить** будет иметь результатом имя пользователя, определенное в настройках резервирования и восстановления конфигурации каждого устройства.

Пример: `'enable'`

Это значение поля **Отправить** вызовет отправку устройству команды `enable`.

Пример: `'show ' + {env/configurationType} + '-config'`

Это значение поля **Отправить** ссылается на переменную среды `configurationType`, определенную ранее выполненными правилами последовательности резевирования/восстановления конфигурации, которая вызвала это Expect-скрипт.

- **Скрипт ошибки**. Подобно полю **Ожидать**, это Выражение AtomMind 112, которое должно иметь результатом строку. Эта строка будет обрабатываться как регулярное выражение 2142, которому должен соответствовать выход устройства. Если это происходит, ответ устройства считается ошибочным, а выполнение Expect-скрипта заканчивается сообщением об ошибке, определенным полем **Сообщение об ошибке**.
- **Сообщение об ошибке**. Это Выражение AtomMind 112, которое должно иметь результатом строку. Эта строка вернется как сообщение об ошибке, если выход устройства будет соответствовать регулярному выражению, определенному полем **Ожидать ошибку**.
- **Выход**. Это Выражение AtomMind 112, которое должно иметь результатом строку, представляющую имя переменной среды 123. Любой выход, произведенный устройством в ответ на команду **Отправить**, будет записан в эту переменную.

Пример: `'configuration'`

Это выражение разрешается в строку `configuration`. Таким образом, реакция устройства на команду **Отправить** будет записываться в переменую среды `configuration`.

- **Фильтр выхода**. Это Выражение AtomMind 112, которое должно иметь результатом строку. Эта строка будет обрабатываться как регулярное выражение 2142. Любая строка, соответствующая этому выражению, будет удаляться из выхода устройства, полученного в ответ на команду **Отправить** до того, как этот выход записывается в переменную **Выход**.

Пример: `"(Building)|(Current)|(Using)|(show)"`

Это значение поля **Фильтр выхода** будет разрешаться в регулярное выражение `(Building)|(Current)|(Using)|(show)`.

Таким образом, слова `Building`, `Current`, `Using` и `Show` будут удаляться из выхода устройства и помещаться в переменную **Выход**.

- **Комментарии**. Любое дополнение, описывающее назначение строки Expect-скрипта.

Среда вычисления для выражений Ожидать, Отправить, Ожидать ошибку, Сообщение об ошибке, Выход и Фильтр выхода

Среда вычисления 114 выражения Expect-скрипта:	
Контекст по умолчанию 119	Контекст устройства для конфигурирования.
Таблица данных по умолчанию 120	• Таблица резервирования или восстановления параметров для скрипта Чтение Конфигурации или Запись Конфигурации. Формат таблицы определяется **Форматом параметров резервирования** или **Форматом параметров восстановления**, определенным в той же записи Таблицы конфигурации 1923, которая определяет текущую последовательность резервирования/восстановления. • Отсутствует для обычных скриптов Конфигурации.
Ряд по умолчанию 119	0
Переменные среды 123	Стандартные 123 переменные. Переменные среды, определенные ранее выполненными правилами последовательности резервирования или восстановления, если скрипт вызывается из такой последовательности. Дополнительные переменные:

Имя переменной	Тип значения	Описание
configurationType	строка	Тип конфигурации устройства, который резервируется или восстанавливается. Эта переменная недоступна для обычных скриптов конфигурации.

10.2.29 NMEA 0183

Драйвер устройства 518 NMEA позволяет AtomMind Server извлекать данные из приемников GPS и другого оборудования, совместимого со стандартом 0183 текстового протокола связи морского (как правило, навигационного) оборудования между собой.

ПОДДЕРЖИВАЕМЫЕ СООБЩЕНИЯ NMEA

Строки глобальной системы позиционирования (GPS):

GPGGA. Данные привязки глобальной системы позиционирования (GPS). Время, место и координаты GPS-приемника.

GPGLL. Географическое положение - Широта /Долгота

GPGSA. GPS DOP и активные спутники.

GPGSV. Информация о видимых спутниках.

GPRMC. Рекомендуемый минимум навигационных данных.

GPVTG. Вектор путевой скорости.

Собственные *строки* фирмы Garmin:

- **PGRME**. Оценка ошибки измерений
- **PGRMF**. Указатель фиксации положения (Position Fix Sentence)
- **PGRMT**. Информация о статусе датчика.
- **PGRMV**. 3D Скорость.

НАСТРОЙКА ATOMMIND SERVER ДЛЯ ПОСЛЕДОВАТЕЛЬНОГО СОЕДИНЕНИЯ

См. Включение последовательного соединения 1448, если возникли проблемы при подключении к устройствам NMEA.

Информация о драйвере

ID плагина драйвера: com.tibbo.linkserver.plugin.device.nmea

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

НАСТРОЙКИ ПОДКЛЮЧЕНИЯ

Настройки подключения определяют, как AtomMind Server взаимодействует с определенным NMEA устройством. Данные настройки доступны через действие Изменить свойства Device контекста Device. Ниже приведены свойства подключения:

Свойство	Описание
Порт	Номер серийного порта.
Скорость передачи данных (Baud Rate)	Скорость передачи информации в бодах.
Управление входящим потоком (Incoming Flow Control)	Тип контроля входящего потока: None, CTS/RTS или XON/XOFF.
Управление исходящим потоком (Outgoing Flow Control)	Тип контроля исходящего потока: None, CTS/RTS или XON/XOFF.

Настройки Device

Драйвер Device NMEA обеспечивает следующие настройки устройства:

ДАННЫЕ NMEA

Данная настройка в виде таблицы содержит значения всех полей сообщений NMEA и имеет следующий формат:

Поле	Описание
Имя	Имя поля
Возраст	Время, прошедшее после получения значения от устройства NMEA. Имейте в виду, что возраст - время между получением значения от устройства и последней синхронизацией. Он **не** включает время, прошедшее после последней синхронизации. Это означает, что возраст будет увеличиваться, если прекратится получение данных NMEA или значение поля будет пустым в большинстве последних сообщений NMEA.
Значение	Строка значения поля (в том виде, в каком она представлена в сообщении NMEA).

Приведенный ниже список показывает имена поддерживаемых полей и их положение в сообщении NMEA:

- **GPGGA,utc,latitude,northHemi,longitude,eastHemi,quality,numberOfSatellites,horDilution,height,,geoidalHeight,,diffCorrection,diffStationId,**
- **GPGLL,latitude,northHemi,longitude,eastHemi,utc,**
- **GPGSA,mode,fixtype,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,posDilution,horDilution,verDilution,**
- **GPGSV,gsvnum,gsvcur,satInView**
- **GPRMC,utc,status,latitude,northHemi,longitude,eastHemi,speedOverGroundKnots,courseOverGround,utcdate,magnVariation,magnVarDirection,**
- **GPVTG,courseOverGround,,magnCourse,,speedOverGroundKnots,,speedOverGround,**

- **PGRME,HPE,,VPE,,EPE,**
- **PGRMF,GPSWeek,GPSSeconds,utcdate,utc,GPSLeapSecondCount,latitude,northHemi,longitude,eastHemi, mode,fixType,speedOverGround,courseOverGround,**
- **PGRMT,GPSModel,romChecksum,recvFailure,storedDataLost,timeLost,oscillatorDrift,dataCollection,boar dTemperature,boardConfig**
- **PGRMV,eastVelocity,northVelocity,upVelocity,**

МЕСТОНАХОЖДЕНИЕ

Местонахождение устройства: широта и долгота

Операции Device

Драйвер не выполняет операции.

События Device

Драйвер не представляет события.

Подключение

Драйвер переводит устройство в режим **онлайн**, если последовательный порт был открыт успешно.

Синхронизация

Данный драйвер постоянно получает и разбирает сообщения NMEA, отправленные устройством. Каждое поддерживаемое сообщение с подходящей контрольной суммой сохраняется внутри драйвера, чтобы самые последние данные были доступны во время следующей синхронизации.

Во время синхронизации 514 драйвер использует внутренние данные для заполнения таблицы **Данных NMEA** и настройки **местоположения.**

10.2.30 OPC

Драйвер устройства 518 *OLE for Process Control (OPC) позволяет* AtomMind Server взаимодействовать с серверами OPC, т.е. выполнять функции OPC Клиента. Данные, предоставленные OPC серверами (и аппаратными устройствами "позади" них), преобразуются в унифицированную форму для обеспечения доступа к различным экземплярам AtomMind. Для более подробной информации о "нормализованном" представлении устройств в AtomMind обратитесь к разделу Device.

В отличие от OPC Клиентов, которые требуют, чтобы OPC сервер и OPC клиент находились на одном компьютере, AtomMind Server взаимодействует с OPC серверами, используя технологию Distributed COM (DCOM). Она обеспечивает AtomMind Server доступ к OPC серверам через IP сети или интернет.

Еще одним преимуществом использования DCOM является тот факт, что AtomMind Server обеспечивает доступ к OPC серверам не только для оперативной системы Microsoft Windows, но также для Linux/Unix и Mac OS.

OPC драйвер AtomMind Server поддерживает спецификацию доступа к OPC данным. Поддержка других стандартов и спецификаций, включая OPC Тревоги и События, будет реализована в будущей версии AtomMind..

Для обеспечения доступа AtomMind Server к вашему OPC серверу необходимо настроить службу DCOM на компьютере, который использует ее для удаленного доступа. Для более подробной информации обратитесь к разделу Настройка DCOM для удаленного доступа 2162.

НАЗВАНИЕ ЭЛЕМЕНТА

Из-за того что ограничений занывания имен переменных 61 AtomMind Server (латиница, цифры и знак нижнего подчеркивания) не все символы имени элемента могут быть преобразованы в имена переменных напрямую. Для элемента, чье имя содержит неподдерживаемые символы, применяется конверсия имен. процесс конверсии имени происходит следующим образом:

- AtomMind Server пытается автоматически закодировать наиболее часто используемые символы с помощью латинских букв (похоже на транскрипцию). Если форма представления не была найдена, используется знак подчеркивания.
- Хэш-код имени элемента источника прилагается, чтобы избежать дуплицирования имен после первого этапа.

ОБНАРУЖЕНИЕ OPC СЕРВЕРА

Драйвер OPC устройства поддерживает функцию *обнаружения OPC сервера.* Обнаружение представляет собой процесс сканирования количества IP хостов сети, нахождение доступных OPC серверов и создание для них учетных записей.

Более подробное описание обнаружения OPC сервера смотрите в AtomMind SCADA/HMI документации.

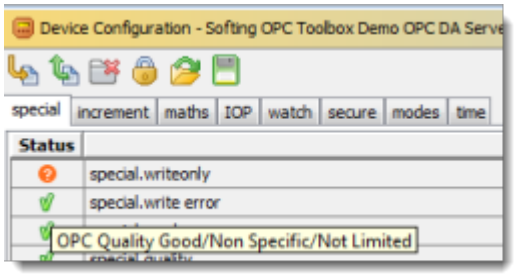
ЗНАЧЕНИЕ OPC ЭЛЕМЕНТА

Каждый OPC элемент является переменной с тремя полями:

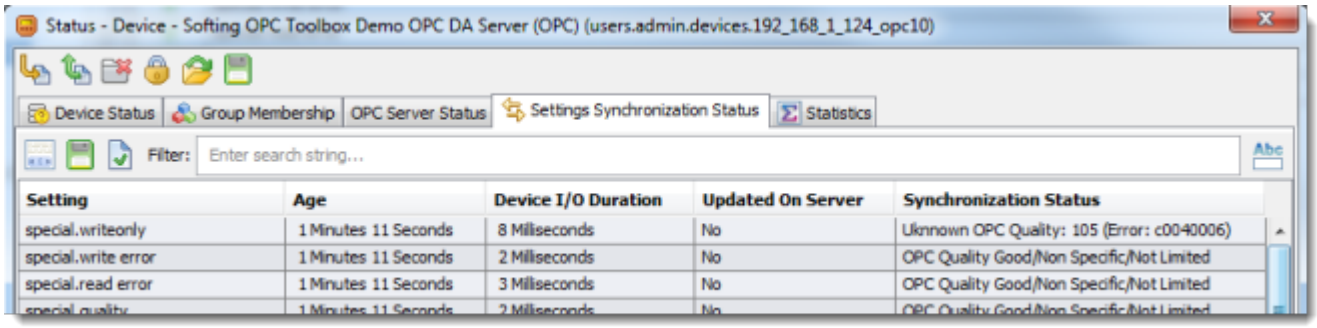
Поле	Тип	Описание
Временная метка	Дата	Временная метка тэга, полученная от сервера OPC.
Качество	Целое	Целое значение качества тэга OPC сервера.
Значение	Любой скалярный тип или Таблица данных 49.	Для скалярных тегов – это просто значение тега с соответствующим типом поля. Для тегов одномерного массива – это таблица данных с одним полем и несколькими записями. Для тегов двумерного массива – это таблица данных с несколькими полями (названными: 1,2,3 и так далее) и несколькими записями. Каждая ячейка таблицы данных значения является скалярным значением элемента массива.

КАЧЕСТВО OPC ЭЛЕМЕНТА

Качество OPC элементов отмечается в статусе 62 переменных настроек соответствующего устройства. Чтобы проверить статус определенной настройки, откройте конфигурацию OPC устройства в редакторе свойств и наведите указатель мышки на иконку статуса настройки. Признак качества будет отображен во всплывающей подсказке:



Чтобы посмотреть качество сразу всех элементов, откройте вкладку "**Настройка статуса синхронизации**" в диалоговом окне "Статус устройства" 111. Качество отображается в столбце "**Статус синхронизации**":



Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.opc

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

СВОЙСТВА ПОДКЛЮЧЕНИЯ OPC СЕРВЕРА

Настройки подключения определяют, как AtomMind Server взаимодействует с определенным OPC сервером. Данные настройки доступны через действие Редактировать свойства устройства[1494] в контексте устройства. Ниже приведен список доступных свойств подключения:

Настройка	Описание
Хост	IP адрес или имя хоста OPC сервера.
Домен	Имя домена Windows OPC сервера (дополнительно).
Пользователь	Имя учетной записи Windows, используемое для доступа к OPC серверу через DCOM.
Пароль	Пароль для учетной записи Windows.
Идентификатор класса приложения (CLSID)	Идентификатор класса OPC сервера. (Например, `F8582CF2-88FB-11D0-B850-00C0F0104305`).
Программный идетификатор (PROGID)	Программный идентификатор OPC сервера. (Например, `Matrikon.OPC.Simulation.1`).
Таймаут	Время ожидания для операций сервера OPC.
Имя группы	Имя группы элемента для добавления элементов OPC, контролируемых AtomMind Server. Должно быть настроено на **Автогенерацию** в большинстве случаев.
Режим чтения	Режим чтения OPC тегов: • **Синхронный (Устройство)** - читает теги *синхронно* из устройства в обход кэша сервера OPC. • **Синхронный (Кэш)** - читает теги *синхронно* из кэша сервера OPC. • **Асинхронный** - читает теги *асинхронно* из устройства в обход кэша сервера OPC.
Период обновления	Период в секундах, в течение которого происходит асинхронное чтение. Определяет, как часто могут обнавляться кешированные данные на сервере OPC. **0** - означает, что все обновления будут немедленно доставлены.
Процент нечувствительности	Процент нечувствительности значения. Значение должно находиться в пределе от 0 до 100. Значение по умолчанию – **0**, которое определяет, что любое изменение значения обновит кэш сервера OPC. Отличное от нуля значение дает в результате значение кэша, обновляемое только если разница между кэшированным значением и текущим значением элемента превышает: `deadbandPercent * (High EU - Low EU) / 100`. Данное свойство влияет только на элементы, that have an analogue data type и свойства 'High EU' и 'Low EU', определенные (ID Свойства 102 и 103 соответственно).

Достаточно определить только CLSID или PROGID OPC сервера, а не оба сразу.

Активы Device

Иерархическая структура активов, предоставляемая драйвером OPC устройства, полностью совпадает с древовидной структурой групп OPC элементов.

Настройки Device

Драйвер OPC устройства создает одну переменную настройки Device для каждого элемента OPC сервера.

ТИП КОНВЕРТАЦИИ

Представленная ниже таблица показывает, как OPC типы преобразуются в типы[49] AtomMind:

OPC тип	Тип AtomMind
VT_BOOL	логическое
VT_BSTR	строка
VT_NULL	строка

VT_EMPTY	строка
VT_INT	целое
VT_I1	целое
VT_I2	целое
VT_I4	целое
VT_UI1	целое
VT_UI2	целое
VT_UI4	целое
VT_DECIMAL	целое
VT_DATE	дата
VT_I8	плавающее
VT_R4	плавающее
VT_R8	плавающее
VT_CY	плавающее

Операции Device

Драйвер не выполняет операции.

События Device

Драйвер не предоставляет события.

Подключение

Драйвер переводит устройство в режим **онлайн** если:

- было установлено TCP подключение к OPC серверу
- DCOM авторизация была выполнена успешно

Синхронизация

OPC серверы синхронизируются 514 с AtomMind Server подобно другим устройствам. Синхронизация между AtomMind Server и OPC сервером включает следующие шаги:

- Чтение информации о настройках, предоставляемой OPC сервером, и создание кэша настроек 502. Настройки подразделяются на несколько групп согласно внутреннему делению OPC сервера.
- Чтение значений настроек OPC сервера и их хранение в кэше настроек.

Статус OPC сервера

Действие Посмотреть статус устройства 111 контекста устройства предоставляет дополнительную информацию о статусе OPC сервера:

- Ширина полосы (Bandwidth)
- Номер сборки (Build Number)
- Текущее время
- Количиество групп
- Время последнего обновления
- Основная версия (Major Version)
- Дополнительная версия (Minor Version)
- Состояние сервера (Доступные состояния сервера: *Сбой соединения, Ошибка, Ошибка конфигурации, Работает, В режиме ожидания, В режиме тестирования* и *Неизвестно*)

- Код состояния сервера
- Время запуска
- Информация о производителе

10.2.31 Omron FINS

Драйвер устройства 518 Omron FINS позволяет AtomMind Server взаимодействовать с устройствами, поддерживающими п**ротокол Omron FINS**.Данные устройства могут быть подключены к системе и, подобно всем другим типам устройств, их данные преобразовываются в специальную форму, так что доступ к ним возможен от разных экземпляров AtomMind. Обратитесь к разделу Devices для получения более детальной информации о "нормализованном" представлении устройств в AtomMind.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.omronfins

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

НАСТРОЙКИ ПОДКЛЮЧЕНИЯ

Настройки подключения определяют, как AtomMind Server взаимодействует с определенным устройством Omron. Данные настройки доступны через опцию изменить свойства 1494 Device контекста Device. Доступны следующие свойства подключения:

Свойство	Описание
IP адрес или имя хоста	Адрес устройства Omron FINS.
Порт	Порт устройства (по умолчанию 9600).
Адрес исходной сети FINS	Этот параметр указывает номер адреса сервера AtomMind внутри исходной сети (по умолчанию 1).
Адрес исходного узла FINS	Этот параметр указывает номер исходного узла (по умолчанию 127).
Адрес исходного элемента FINS	Этот параметр указывает номер исходного элемента (по умолчанию 127).
Адрес сети назначения FINS	Этот параметр указывает номер адреса назначения (по умолчанию 1).
Адрес узла назначения FINS	Этот параметр указывает номер узла назначения (по умолчанию 0).
Таймаут	Таймаут (в мс) ответов, которые должны быть получены обратно от удаленных узлов (по умолчанию 1 секунда).
Повторы	Количество раз, которые пакет будет послан прежде чем он будет признан недоставляемым (по умолчанию 2).

РЕГИСТРЫ УСТРОЙСТВА

Это свойство содержит список регистров Omron устройств, которые доступны и управляются AtomMind. Как только добавлено новое устройство Omron, один или более регистров должны быть настроены для того, чтобы данные устройства были доступны для системы. Каждый регистр представлен одной переменной 61 контекста 1494 Device.

Устройства Omron не предоставляют **метаданные**, поэтому AtomMind Server не может узнавать о доступных регистрах Omron отдельного устройства. Вот почему необходимо настраивать регистры устройства вручную.

Вот список свойств каждого регистра Omron:

Свойство	Описание
Имя	Имя регистра. Переменная контекст 1494 Device, которая будет использована для доступа к регистру. Поэтому она может содержать только буквы, цифры и нижнее подчеркивание.
Описание	Текстовое описание регистра. Используется как описание переменной контекста Device.
Адрес элемента FINS	Этот параметр указывает номер элемента назначения (по умолчанию 0).
Тип данных	Определяет как интерпретировать значение одного или более смежного регистра. Обратитесь к разделу конвертация 623 за деталями. **Типы данных**: Boolean, Short, Word, Long, DWord, Float, Unsigned BCD, Unsigned Long BCD, Signed BCD, Signed Long BCD, String
Порядок байт	Порядки байт: Hi-Lo, Lo-Hi, Hi Only, Lo Only.
Тип памяти	Номер типа памяти регистра устройства Omron в десятичной форме.
Изменяемая	Определяет, является ли регистр изменяемым.
Адрес регистра (смещение)	Адрес (смещение) регистра устройства Omron введён в *десятичной* форме или форме *с плавающей запятой*. Форма *с плавающей запятой* используется только для логических регистров. Значение после запятой ссылается на особый разряд регистра. Например: 100.0, значение регистра равно 100, а разряд равен 0.
Размер	Количество регистров для чтения одной операцией ввода/вывода Omron FINS и хранения в одной переменной AtomMind Server контекста. Чтение регистров сразу может быть полезным в следующих случаях: • Если они логически представляют массив • Если они логически представляют строку • Если они логически представляют различные части сложного элемента данных, который должен быть прочтен за одну неделимую операцию Фактическое количество регистров, читаемых за одну операцию ввода/вывода будет ровняется N * M, где • N - это количество регистров, требующихся, чтобы содержать одно значение на стороне AtomMind Server в зависимости от **типа данных** • M - это значение параметра **размер** Значение данной настройки по умолчанию - 1. В большинстве случаев его не нужно изменять.
Порядок слов	Порядки слов: Hi-Lo, Lo-Hi.

Вы можете импортировать список регистров из файла (напр. файла CSV), используя функцию Импорт 382 компонента Редактор таблицы данных 382.

Если Вы хотите подключить несколько похожих устройств Omron к AtomMind, вы можете заполнить таблицу регистров устройства только один раз, а после скопировать её на другие устройства, используя действие репликации 110.

Активы Device

Драйвер не поддерживает активы.

Настройки Device

Драйвер устройства Omron создает одну переменную настроек Device на каждый регистр устройства.

КОНВЕРТАЦИЯ

Данная таблица показывает, как Omron регистры конвертируются в переменные контекста Device. Заметьте, что номер рядов в каждой переменной зависит от значения параметра **Размер**. По умолчанию, все переменные имеют одну строку, т.е. скалярны.

Тип данных	Описание	Формат переменной AtomMind Server
Boolean	2-байтный Int Unsigned	Читаемый/записываемый, 1 колонка типа Логическое
Short	2-байтный Int Signed	Читаемый/записываемый, 1 колонка типа Целое
Word	2-байтный Int Unsigned	Читаемый/записываемый, 1 колонка типа Целое
Long	4-байтный Int Signed	Читаемый/записываемый, 1 колонка типа Целое
DWord	4-байтный Int Unsigned	Читаемый/записываемый, 1 колонка типа Целое
Float	4-байтный Float	Читаемый/записываемый, 1 колонка типа Двойное
Unsigned BCD	2-байтный BCD Unsigned (0 - 9999)	Читаемый/записываемый, 1 колонка типа Целое
Unsigned Long BCD	4-байтный BCD Unsigned (0 - 99999999)	Читаемый/записываемый, 1 колонка типа Целое
Signed BCD	2-байтный BCD Signed (-7999 - 7999)	Читаемый/записываемый, 1 колонка типа Целое
Signed Long BCD	4-байтный BCD Signed (-79999999 - 79999999)	Читаемый/записываемый, 1 колонка типа Целое
String	Строковый	Читаемый/записываемый, 1 колонка типа Строка. Сервер читает количество регистров, указанных в параметре **Размер,** и представляет их как строку.

Операции Device

Драйвер не проводит операции.

События Device

Драйвер не представляет события.

Подключение

Драйвер переводит устройство в режим **онлайн** если:

- TCP подключение к Device было успешно установлено

Синхронизация

Синхронизация между AtomMind Server и устройством Omron включает в себя следующие шаги:

- Создание кэша настроек 502 в соответствии со списком регистров устройства. Каждая переменная используется для доступа к одному регистру устройства Omron.
- Чтение значений регистра Omron и хранение этих значений в кэше настроек.

10.2.32 OPC UA

Драйвер устройства 518 *Единая архитектура OPC (OPC UA)* позволяет AtomMind Server взаимодействовать с серверами OPC UP, то есть действует как Клиент OPC UA. Данные, предоставленные серверами OPC UA (и аппаратные устройства "за" ними) преобразовываются в унифицированную форму, так что доступ к ним возможен от разных экземпляров AtomMind. Обратитесь к разделу Устройства 497 для получения более детальной информации о "нормализованном" представлении устройств в AtomMind.

Информация о драйвере

ID плагина драйвера: com.tibbo.linkserver.plugin.device.opcua

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства аккаунта Device

СВОЙСТВА ПОДКЛЮЧЕНИЯ СЕРВЕРА OPC UA

Настройки подключения определяют, как AtomMind Server взаимодействует с определенным сервером OPC UA. Данные настройки доступны через действие Редактировать свойства устройства контекста Device . Доступны следующие свойства подключения:

Настройка	Описание
URL	URL сервера OPC UA.
Начальные пути адресного пространства	Таблица узлов адресного пространства OPC UA сервера, которая будет видна из AtomMind Server с их подузловыми деревьями. Проще говоря, это список корневых узлов сервера OPC UA, подключенных к AtomMind Server.
Пользователь	Имя пользователя для аутентификации.
Пароль	Пароль для аутентификации.
Режим безопасности	Режим безопасности связи OPC UA, один из: • Отсутствует • Базовый 128-разрядный RSA15, подписать • Базовый 128-разрядный RSA15, подписать и зашифровать • Базовый 256-разрядный, подписать • Базовый 256-разрядный, подписать и зашифровать
Максимальный возраст кэша	Контролирует, должен ли OPC UA сервер использовать кэшированное значение, или пытаться прочитать его из базового источника данных (например, устройства) по запросу чтения от AtomMind Server'а. Если на сервере нет значения в пределах максимального возраста, он будет читать новое значение из источника данных. Если установить это значение на 0, это приведет к прямому доступу к источнику данных во время каждой синхронизации. Это может понизить производительность, но помогает избежать устаревших значений.

ПОДПИСКА НА СОБЫТИЯ

Таблица подписок на события определяет, как события OPC UA конвертируются в события AtomMind. Она имеет следующие колонки:

Имя	Имя события AtomMind. Должно соответствовать правилам именования событий, например, включать только английские буквы, числа и нижние подчеркивания.
Описание	Удобочитаемое описание события AtomMind.
Путь к объекту	Путь узлов OPC UA, события которых будут конвертированы в события AtomMind.
Тип события	Тип события OPC UA, который будет конвертирован в событие AtomMind.

Активы Device

Драйвер устройства OPC UA создает актив для каждого узла дерева сервера OPC UA.

Настройки Device

Драйвер устройства OPC UA создает переменную настроек Device для каждого атрибута узла OPC UA.

Операции Device

Драйвер устройства OPC UA создает функцию контекста для каждого метода узла OPC UA.

Драйвер также предоставляет операцию **Чтение архивных данных** для получения диапазона исторических значений тегов от сервера OPC UA.

События Device

Драйвер устройства OPC UA создает событие контекста Device для каждой записи в таблице **Подписка на события.** Он прослушивает события, соответствующие записи, и когда такое событие происходит, оно исправляется на событие контекста %AG%>, определенное записью.

Подключение

Драйвер переводит устройство в режим **онлайн** если:

- TCP подключение к серверу OPC UA успешно установлено
- Авторизация OPC UA прошла успешно

Синхронизация

Серверы OPC UA синхронизируются 514 с AtomMind Server как и любые другие Devices. Синхронизация между AtomMind Server и сервером OPC UA включает в себя следующие шаги:

- Чтение информации о настройках, предоставленных сервером OPC UA, и создание кэша настроек 502. Настройки разделены на несколько групп, в соответствии с внутренним делением сервера OPC UA.
- Чтение значений настроек сервера OPC UA и хранение этих значений в кэше настроек.

10.2.33 SIP

Драйвер устройства 518 SIP (Session Initiation Protocol) позволяет AtomMind Server совершать автоматические SIP звонки для проверки доступности, степени исправности и работоспособности VoIP серверов.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.sip

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

СВОЙСТВА ПОДКЛЮЧЕНИЯ

Настройки подключения определяют, как AtomMind Server взаимодействует с определенным SIP сервером. Данные настройки доступны через опцию изменить свойства 1494 Device контекста Device. Доступны следующие свойства подключения:

Настройка	Описание
Локальный IP-адрес	Локальный IP-адрес.
Локальный порт	Локальный порт для прослушивания ответов. **Вы должны использовать порт отличный от удалённого порта если AtomMind Server или Агент запущенны на одном с SIP устройством хосте.**

Удаленный IP-адрес	IP-адрес SIP устройства.
Удаленный порт	Порт SIP устройства.
Протокол	Транспортный протокол: TCP или UDP.
Логин	Логин для регистрации.
Пароль	Пароль для регистрации.
Таймаут	Таймаут SIP операций в миллисекундах.

Комманды и звонки

Таблица команд и звонков определяет, какие SIP комманды и звонки будут выполнены драйвером во время каждого цикла синхронизации 514.

Параметр	**Описание параметра**
Имя	Имя SIP агента. Также соответствует имени переменной настроек Device, которая предоставит результаты SIP комманды/звонка.
Описание	Описание переменной настроек Device, которая предоставит результаты SIP комманды/звонка.
Таймаут	Таймаут SIP операции в секундах.
Команда	Команда, которая будет послана SIP устройству. **Имя команды** / **Описание команды** INVITE – Используется для установки медиа сеанса между двумя Агентами. MESSAGE – Посылает текстовое сообщение SIP устройству. OPTIONS – Информация запросов о возможностях звонящего без настроек звонка.
SIP данные	Содержит текст для команды MESSAGE.

SIP ответы

1XX = ИНФОРМАЦИОННЫЕ SIP ОТВЕТЫ

Код	**Описание**
100	Trying – Запрос обрабатывается.
180	Ringing – Агент адресата получил сообщение INVITE и предупреждает позвонившего пользователя.
181	Call Is Being Forwarded – Необязательный, посылается сервером для обозначения того, что звонок перенаправляется.
182	Queued – Вызываемый абонент временно не доступен, вызов поставлен в очередь.
183	Session Progress - Этот ответ может быть использован для отсыла дополнительной информации о звонке, который все еще устанавливается.
199	Early Dialog Terminated – Посылается сервером пользовательских агентов для обозначения того, что более ранний диалог был прерван.

2XX = ОТВЕТЫ ОБ УСПЕШНОМ ВЫПОЛНЕНИИ

Код	**Описание**

200	OK – Показывает, что запрос был успешно обработан.
202	Accepted - Показывает, что запрос был принят для обработки, в основном используется для рефералов. Устаревшее.
204	No Notification – Показывает, что запрос был успешным, но ответ не будет получен.

3XX = ОТВЕТЫ ПЕРЕАДРЕСАЦИИ

Код	Описание
300	Multiple Choices – Указывает несколько SIP-адресов, по которым можно найти вызываемого пользователя.
301	Moved Permanently - вызываемый пользователь больше не находится по адресу, указанному в запросе, новый адрес дан в заголовке Contact.
302	Moved Temporarily - Пользователь временно сменил местоположение.
305	Use Proxy - Вызываемый пользователь не доступен непосредственно, входящий вызов должен пройти через прокси-сервер.
380	Alternative Service – Звонок не удался, но альтернативные сервисы детализированы в теле сообщения.

4XX = ОШИБКИ ЗАПРОСА

Код	Описание
400	Bad Request - Запрос не понят из-за синтаксических ошибок в нем, ошибка в сигнализации.
401	Unauthorized - Нормальный ответ сервера о том, что пользователь еще не авторизировался, обычно после этого абонентское оборудование отправляет на сервер новый запрос, содержащий логин и пароль.
402	Payment Required - Требуется оплата (зарезервирован для использования в будущем).
403	Сервер понял запрос, но не может его выполнить. No Such User - нет такого пользователя, ошибка в номере, логине или пароле. User Disabled - пользователь отключен. Wrong Guess - ошибка в пароле. Conflict - такой SIP-номер уже используется. Forbidden - абонент не зарегистрирован. Empty Route Set - нет ни одного шлюза в роутинге. Caller Not Registered - нет такого пользователя. Out of Look-Ahead Retries - перебор узлов закончен. Invalid Phone Number - нет такого направления. No Money Left on RFC Account - на счету нет денег для совершения звонка.
404	Not Found – Сервер получил информацию о том, что пользователь не существует (пользователь не найден).
405	Method Not Allowed - Метод, заданный в строке запроса понят, но не разрешен.
406	Not Acceptable - Ресурс способен генерировать только запросы с недопустимым контентом.
407	Proxy Authentication Required - Запрос требует аутентификации пользователя на прокси-сервере.
408	Request Timeout – Время обработки запроса истекло: Абонента не удалось найти за отведенное время.
409	Conflict – Пользователь уже зарегистрирован (устаревший)

410	Gone – Пользователь существовал ранее, но теперь недоступен.
411	Length Required - Сервер не принимает запрос без корректной длины содержимого (устаревший).
413	Request Entity Too Large - Размер запроса слишком велик для обработки на сервере.
414	Request URI Too Long – Сервер отказывается обрабатывать запрос, запрошенный URI длинее, чем сервер может интерпретировать.
415	Unsupported Media Type - Звонок совершается неподдерживаемым кодеком.
416	Unsupported URI Scheme - URI запроса неизвестен серверу.
417	Uknown Resource-Priority - Есть тег опции приоритета ресурса, но нет заголовка приоритета ресурса.
420	Bad Extension – Неизвестное расширение: Сервер не понял расширение протокола SIP.
421	Extension Required – В заголовке запроса (Supported) не указано, какое расширение сервер должен применить для его обработки.
422	Session Interval Too Small - Запрос содержит поле заголовка Session-Expires с длительностью ниже минимальной.
423	Interval Too Brief - Срок действия ресурса слишком короткий.
424	Bad Location Information – Содержание запроса было искажено или являлось неудовлетворительным по другим причинам.
428	Use Identity Header - Политика сервера требует заголовка Identity, не один не был предоставлен.
429	Provide Referrer Identity - Сервер не получил действительный "Referred-By" признак в запросе.
430	Flow Failed - Особый поток для пользовательского агента дал сбой, хотя другие потоки могут быть успешными.
433	Anonymity Disallowed - Запрос был отклонен из-за своей анонимности.
436	Bad Identity Info – Запрос имеет заголовок Identity-Info, и схема URI не может быть разыменована.
437	Unsupported Certificate - Сервер был неспособен подтвердить правильность сертификата для домена, который подписал запрос.
438	Invalid Identity Header – Сервер получил корректный сертификат для подписи запроса, но был неспособен проверить подпись.
439	First Hop Lacks Outbound Support - Первый исходящий прокси-сервер не поддерживает “исходящую” функцию.
470	Consent Needed - У источника запроса не было разрешения от адресата на данный запрос.
480	Invalid Phone Number - неправильный номер телефона, не соответствует к-во цифр или неправильный код страны или города. Destination Not Found In Client Plan - направления нет в тарифном плане абонента. Wrong DB Response - проблемы с центральной базой сети. DB Timeout - проблемы с центральной базой сети. Database Error - проблемы с центральной базой сети. Codec Mismatch - несоответствие кодеков. No Money Left on RFC Account - нет денег на счету, обратитесь к администратору сети. Empty Route Set - пустое направление, нет принемающих шлюзов. No money left - недостаточно денег на счете. Temporarily Unavailable - временно недоступное направление попробуйте позвонить позже.

481	Call/Transaction Does Not Exist - Сервер получил запрос, который не соответствует какому-либо диалогу или транзакции.
482	Loop Detected - Обнаружен замкнутый маршрут передачи запроса.
483	Too Many Hops - Запрос на своем пути прошел через большее число прокси-серверов, чем разрешено (Max-Forwards достиг значения '0').
484	Address Incomplete - URI запроса неполный.
485	Ambiguous - URI адрес вызываемого пользователя не однозначен.
486	Busy Here - Вызываемый абонент занят.
487	Request Terminated - Запрос был прерван Bye или Cancel.
488	Not Acceptable Here - Некоторые аспекты описания сессии URI запроса недопустимы.
489	Bad Event - Сервер не понял пакет событий, указанный в поле заголовка Event.
491	Request Pending - Запрос поступил в то время, когда сервер еще не закончил обработку другого запроса, относящегося к тому же диалогу.
493	Undecipherable - Сервер не в состоянии подобрать ключ дешифрования: невозможно декодировать тело S/MIME сообщения.
494	Security Agreement Required - Сервер получил запрос, который требует согласованного механизма обеспечения защиты.

5XX = ОШИБКИ СЕРВЕРА

Код	**Описание**
500	Server Internal Error - Внутренняя ошибка сервера. DB Timeout - Нет ответа от базы данных. Database Error - Ошибка базы данных. Wrong DB Response - Неправильный ответ базы данных. Undefined Reason - Неопределенная причина.
501	Not Implemented – Метод SIP запроса не реализован.
502	Bad Gateway - Сервер получил некорректный запрос от последующего сервера, пытаясь выполнить запрос.
503	Service Unavailable - Сервер находится на техническом обслуживании или временно перегружен и не может обработать запрос.
504	Server Time-out – Сервер не получил ответа в течение установленного промежутка времени от сервера, к которому он обратился для завершения вызова.
505	Version Not Supported – Сервер не поддерживает версию запрошенного SIP протокола.
513	Message Too Large - Длина сообщения больше, чем сервер может обработать.
580	Precondition Failure - Сервер не способен или не хочет соответствовать ограничениям, определенным в предложении.

6XX = ОБЩИЕ ОШИБКИ

Код	**Описание**
600	Busy Everywhere - Все возможные адресаты заняты.
603	Decline – Адресат не может/не хочет участвовать в звонке, альтернативных адресатов нет.
604	Does Not Exist Anywhere - Сервер получил достоверную информацию, что запрошенный пользователь не существует.
606	Not Acceptable - Связь с агентом пользователя была успешно установлена, но некоторые аспекты описания сессии были неприемлимыми.

Активы Device

Драйвер не поддерживает активы.

Настройки Device

Драйвер создает отдельные переменные настроек Device для каждой записи в таблице звонков и команд. Эта переменная содержит результат команды или звонка.

Операции Device

Драйвер не проводит операции.

События Device

Драйвер не представляет события.

Подключение

Sip драйвер посылает команды SIP устройству и преобразует ответ в таблицу с данными.

10.2.34 SMB/CIFS (мониторинг совместно используемого ресу

Драйвер устройства SMB/CIFS обеспечивает мониторинг файлов и папок, совместно используемых через технологию **Server Message Block (SMB)**, которая также называется **Common Internet File System (CIFS)** или **Microsoft Windows Network**.

Информация о драйвере

ID плагина драйвера: com.tibbo.linkserver.plugin.device.samba

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

Учетная запись устройства SMB/CIFS настраивается при помощи следующих свойств:

Имя поля	Описание поля
Адрес	IP адрес или имя хоста сервера SMB.
Протокол	Выберите ваш протокол: `SMB v1` или `SMB v2/v3`.
Путь	Расположение (полный путь) контролируемого удаленного файла или папки.
Содержимое чтения	Включает/выключает чтение содержимого файла.
Рекурсивный	Включает/выключает рекурсивное чтение для папок.
Пошаговое чтение	Переключает с чтения всего файла на пошаговое чтение. Пошаговое чтение идеально подходит для анализа журнала регистрации. Если оно включено, сервер запоминает размер предыдущего файла и во время следующего цикла синхронизации читает данные, начиная с последней точки и до конца файла. Если размер файла не вырос с предыдущего цикла, чтение не осуществляется. Если размер файла уменьшился, чтение возобновляется с начала файла (это подходит для управления ротацией журнала регистрации).
Максимальный размер чтения	Максимальное количество байт, которое читает сервер, если включено пошаговое чтение.

Размер чтения очереди сообщений	Количество дополнительных байт с предыдущего этапа пошагового чтения.
Разрешить редактирование	Разрешает или запрещает изменение содержимого файла.
Рассчитать контрольную сумму	Включает/выключает расчет контрольной суммы файла.
Активировать аутентификацию	Включает/выключает доступ к защищенным паролем ресурсам.
Домен	Домен, с котором существует имя пользователя.
Имя пользователя	Имя пользователя для аутентификации.
Пароль	Пароль для аутентификации.

активы Device

Драйвер не поддерживает активы.

Настройки Device

Драйвер устройства SMB/CIFS создает следующие переменные настроек Device:

Имя переменной	Описание переменной	Комментарии
attributes	атрибуты файла	Эта переменная создается, только если **Путь** указывает на файл. Содержит атрибуты файла: • **Время последнего изменения** (modificationTime) • **Размер** (size) • **Контрольные суммы**, рассчитанные с ипользованием алгоритма Adler-32 (checksum)
attributes	атрибуты папки	Эта переменная создается, только если **Путь** указывает на папку. Содержит атрибуты папки: • **Время последнего изменения** (modificationTime) • **Размер** (size) • **Количество файлов** (fileCount)
contents	содержимое файла	Эта переменная создается, только если **Путь** указывает на файл. Она имеет содержимое файла только для чтения (если включена опция Читать содержимое 576) или редактирования (если включена опция Разрешить редактирование 577).
contents	содержимое папки	Эта переменная создается, только если **Путь** указывает на файл. Список элементов папки (файлы и подпапки) со следующими полями: • **Тип** (itemType) • **Относительный путь** (itemPath) • **Размер** (itemSize)

Операции Device

ЗАКАЧАТЬ ФАЙЛ

Операция закачивает файл.

Вход:

- **Файл**. Data Block 52. Содержимое файла.

Результат:

- **Результат**. String 51. Статус операции закачивания файла.

СКАЧАТЬ ФАЙЛ

Операция скачивает файл.

Вход:

- **Файл**. String 51. Имя файла.

Результат:

- **Результат**. Data Block 52. Содержимое файла.

ПЕРЕИМЕНОВАТЬ ФАЙЛ

Операция переименовывает файл.

Вход:

- **Файл**. String 51. Имя файла.
- **Новое имя файла**. String 51. Новое имя файла.

Результат:

- **Результат**. String 51. Статус операции переименования файла.

УДАЛИТЬ ФАЙЛ

Операция удаляет файл.

Вход:

- **Файл**. String 51. Имя файла.

Результат:

- **Результат**. String 51. Статус операции удаления файла.

События Device

Драйвер не предоставляет событий.

Подключение

Устройство SMB/CIFS имеет статус онлайн, если успешно выполнены следующие шаги:

- Установлено соединение с удаленным сервером
- Аутентификация прошла успешно (если активирована)
- Удаленный ресурс, указанный в настройках **Пути**, существует и доступен
- Все операции ввода/вывода с ресурсом (т.е. чтение и запись содержимого файла/папки) были успешно завершены

Синхронизация

Во время синхронизации 514 устройства SMB/CIFS с сервером выполняются следующие операции:

- Если **Путь** указывает на файл, получаются атрибуты и содержимое (опционально) файла
- Если **Путь** указывает на папку, извлекаются атрибуты папки и список файлов, находящихся в этой папке и ее подпапках (если активирован **Рекурсивный**)
- Если редактирование разрешено и пользователь отредактировал серверную копию содержимого файла, новое содержимое записывается в файл

10.2.35 SMI-S

Драйвер устройства Storage Management Initiative – Specification (стандарта управления дисковыми хранилищами) позволяет мониторить и управлять устройствами SMI-S.

С драйвером AtomMind Server вы можете:

- Читать и записывать свойства объектов SMI-S
- Вызывать методы, предоставляемые объектами SMI-S

Информация о драйвере

ID плагина драйвера: com.tibbo.linkserver.plugin.device.smis

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

НАСТРОЙКИ СОЕДИНЕНИЯ

Настройки	Описание
Протокол	Протокол соединений HTTP или HTTPS.
Адрес	IP-адрес устройства SMI-S.
Порт	Порт устройства SMI-S.
Логин	Логин для аутентификации.
Пароль	Пароль для аутентификации.
Таймаут	Таймаут операций SMI-S в мс.
Обнаружить пространства имен	Обнаружить или не обнаружить пространства имен на устройстве SMI-S.
В виде дерева	Использовать просмотр активов в виде дерева (иерархический).
Пространства имен	Список пространств имен на устройстве для чтения активов.

Активы Device

Для каждого класса SMI-S драйвер создает актив с таким же именем. Чтобы включить актив, пользователь требует у драйвера извлечь все экземпляры соответствующего класса. По умолчанию большинство активов отключены, только некоторые часто используемые пользователем могут быть активны.

У некоторых классов SMI-S может быть тысяча или миллион экземпляров. Включение этих классов может стать причиной замедления синхронизации и повышенного потребления ресурсов.

Настройки Device

Драйвер устройства SMI-S создает переменные настроек Device следующим образом:

- Для каждого включенного актива (класса SMI-S) он создает переменную с таким же именем.

Настройка соединения

Драйвер SMI-S приводит устройство в режим **Online**, если:

- Соединение с устройством SMI-S с использованием имени пользователя, пространства имени и предоставленного пароля установлено.
- Доступ к указанному пространству имени получен.

Операции Device

Драйвер SMI-S создает функцию контекста Device и соответствующее действие для каждого метода, который предоставляется активными классами SMI-S. Эти действия группируются по именам класса.

В этом случае действие SMI-S запрашивает путь объекта и параметры методов, вызыват метод для объекта, преобразует вывод и показывает это в виде таблицы результатов.

Возможно также указать пользовательский таймаут до вызова любого метода SMI-S.

Детали синхронизации

Устройства SMI-S синхронизируются 514 с AtomMind Server так же, как и любые другие Device. Синхронизация включает в себя следующие шаги:

- Чтение определений активов (если они еще не были прочитаны или были перезагружены).

Каждый элемент актива связан с классом SMI-S с таким же именем.

- Сбор метаданных устройства:
  - Для каждого включенного актива драйвер возвращает определение соответствующего класса SMI-S, включающего его свойства и спецификации методов.
- Чтение/написание настроек устройства:
  - Драйвер читает свойства всех экземпляров классов, определенных активами.
  - Драйвер записывает измененные свойства объектов SMI-S обратно на управляемое устройство.

Преобразование данных SMI-S

Объекты SMI-S преобразуются в таблицы AtomMind следующим образом:

- Таблица содержит данные одного или нескольких (массивов) объектов SMI-S.
- Каждый объект представляется единственной записью данных.
- Таблица содержит поле **путь объекта**, которое определяет объект, используя его путь в пространстве имен.
- Другие поля в таблице отражают свойства объекта SMI-S. Следующая таблица показывает, как типы SMI-S преобразуются в типы 52 AtomMind и наоборот:

Тип WMI	**Тип AtomMind**
Беззнаковый 8-битный целочисленный	Целочисленный
Знаковый 8-битный целочисленный	Целочисленный
Беззнаковый 16-битный целочисленный	Целочисленный
Знаковый 16-битный целочисленный	Целочисленный
Беззнаковый 32-битный целочисленный	Целочисленный
Знаковый 32-битный целочисленный	Целочисленный
Беззнаковый 64-битный целочисленный	Длинный
Знаковый 64-битный целочисленный	Длинный
UCS-2 строка	Строка
Булевое	Булевое
IEEE 4-байтовая плавающая точка	Плавающее
IEEE 8-байтовая плавающая точка	Двойное

Дата	Данные
Ссылка	Строка
16-битный символ UCS-2	Строка
Объект	Строка
Массивы	Таблица данных
Недопустимый тип данных	Строка
Класс	Строка

Методы SMI-S

Чтобы выполнить метод драйвера SMI-S, выберите метод в меню контекста активов, далее заполните параметры в открывшемся окне и нажмите OK. Результат будет предоставлен в виде **таблицы данных** 49.

При выполнении методов будьте внимательны, некоторые из них могут изменять содержимое классов или влиять на производительность устройства.

10.2.36 SMPP

Драйвер устройства 518 SMPP позволяет AtomMind Server отправлять SMS сообщения через шлюз Простое сообщение равноправных узлов (SMPP).

Информация о драйвере

ID плагина 518 драйвера : com.tibbo.linkserver.plugin.device.smpp

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

НАСТРОЙКА СОЕДИНЕНИЯ

Настройки соединения определяют, как AtomMind Server общается с модемом. К этим настройкам можно получить доступ, используя действие Редактировать свойства 1494 Device контекста Device. Доступны следующие свойства соединения:

Свойство	Описание
IP адрес или имя хоста	Адрес шлюза SMPP.
Порт	Порт шлюза SMPP для соединения.
Номер отправителя	Номер телефона, который используется для определения отправителя SMS сообщения.
ID системы	Имя пользователя (также известное как идентификатор системы) вашей учетной записи SMPP.
Тип системы	Текст, который шлюз SMPP отправляет серверу SMPP, чтобы обозначить, какого рода это устройство. Некоторые серверы SMPP требуют значения системы специального типа.
Пароль	Пароль для аутентификации на шлюзе SMPP.

<table>
<tr><td>Версия интерфейса</td><td>Версия протокола SMPP, т.е. 3.2. Если не установлена, будет использоваться самая последняя поддерживаемая версия (3.4).</td></tr>
<tr><td>Тип номера отправителя (TON)</td><td>Десятичный тип номера (TON) отправителя. Поддерживаются следующие значения TON:
<table>
<tr><th>TON Значение</th><th>Описание</th></tr>
<tr><td>0</td><td>Неизвестное</td></tr>
<tr><td>1</td><td>Международное</td></tr>
<tr><td>2</td><td>Национальное</td></tr>
<tr><td>3</td><td>Специфическое для сети</td></tr>
<tr><td>4</td><td>Номер подписчика</td></tr>
<tr><td>5</td><td>Алфавитно-цифровое</td></tr>
<tr><td>6</td><td>Аббревиатура</td></tr>
</table></td></tr>
<tr><td>Числовой индикатор плана отправителя</td><td>Десятичный числовой индикатор плана (NPI) номера отправителя. Поддерживаются следующие значения NPI:
<table>
<tr><th>NPI Значение</th><th>Описание</th></tr>
<tr><td>0</td><td>Неизвестное</td></tr>
<tr><td>1</td><td>ISDN (E163/E164)</td></tr>
<tr><td>3</td><td>Дата (X.121)</td></tr>
<tr><td>4</td><td>Телекс (F.69)</td></tr>
<tr><td>6</td><td>Наземное мобильное (E.212)</td></tr>
<tr><td>8</td><td>Национальное</td></tr>
<tr><td>9</td><td>Частное</td></tr>
<tr><td>10</td><td>ERMES</td></tr>
<tr><td>14</td><td>Интернет (IP)</td></tr>
<tr><td>18</td><td>WAP Client Id</td></tr>
</table></td></tr>
<tr><td>Тип номера получателя</td><td>Десятичный тип номера (TON) получателя.</td></tr>
<tr><td>Числовой индикатор плана получателя</td><td>Десятичный числовой индикатор плана (NPI) номера получателя.</td></tr>
<tr><td>Кодирование данных</td><td>Схема кодирования данных.</td></tr>
<tr><td>Кодирование сообщения</td><td>Кодирование символов сообщения.</td></tr>
<tr><td>Размер окна</td><td>Размер окна SMPP (максимальное количество неподтвержденных запросов).</td></tr>
<tr><td>Время ожидания подключения</td><td>Время ожидания операций сокетов.</td></tr>
</table>

Время ожидания запроса	Время ожидания для запросов SMPP.
Время ожидания окончания срока действия, миллисекунды	Время ожидания ответа шлюза SMPP на запрос, прежде чем закончится срок его действия. По умолчанию стоит на выключено (-1).

Настройки Device

Драйвер предлагает переменную `smppMessagesSent`, указывающую, сколько сообщений было отправлено через учетную запись устройства SMPP с начала работы сервера.

Операции Device

ОТПРАВКА SMS

Эта операция используется для отправки одного или многих SMS сообщений через шлюз SMPP. Каждое сообщение определяется номером получателя и текстом сообщения.

События Device

Драйвер не предоставляет событий.

Подключение

Драйвер переводит устройство в режим **онлайн,** если соединение с шлюзом SMPP было установлено и команда SMPP **привязка** выполнена успешно.

10.2.37 SNMP

Драйвер устройства 518 SNMP обеспечивает коммуникации через **Simple Network Management Protocol** (**SNMP**), позволяя осуществлять обмен информацией управления с *сетевыми устройствами SNMP.* Поддерживаются все версии SNMP протокола (v1, v2(c) и v3), включая безопасное подключение, представленное в SNMP v3.

Подобно другим типам устройства, данные, собранные с SNMP устройств, конвертируются в унифицированную форму для обеспечения доступа к различным экземплярам AtomMind. Для более подробной информации о "нормализованном" представлении устройств в AtomMind см. раздел Device.

Существует два основных метода SNMP мониторинга, их описания представлены в данных разделах: опрос 640 и мониторинг ловушек (**Traps** 643) и сообщений (**Inform** 643).

Для более подробной информации об общих настройках, используемых службой SNMP, обратитесь к подразделу Глобальные SNMP настройки 644.

Чтобы драйвер устройства SNMP работал на AtomMind Server с Linux , убедитесь, что локальный компьютер правильно настроен в файле `/etc/hosts`:

```
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
```

РЕЖИМ ВЫБРАННЫХ СУЩНОСТЕЙ

У драйвера устройства SNMP особое поведение, когда **активные сущности,** переключаемые в общих свойствах устройства 510, устанавливают на **выбранные сущности**.

Обычно, когда активные сущности установлены на **все сущности**, драйвер представляет SNMP walk возможность исследовать, какие переменные доступны в каждом активе (файл MIB).

Если активные сущности установлены на **выбранные сущности**, список **переменных** в свойствах аккаунта устройства содержит все перменные, которые определены в активе (файл MIB). В этом случае драйвер прямо читает выбранные переменные, опираясь на файл MIB без предоставления SNMP walk.

Поэтому для устройств, которые предоставляют большой объем инфорамции SNMP, рекомендуется режим **выбранные сущности** для увеличения скорости синхронизации устройства и избежания сохранения ненужных значений.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.snmp

Общие настройки

См. Общие SNMP настройки 644.

Настройки уровня пользователя

Не определены.

Свойства Device

Свойства Device определяют параметры SNMP коммуникации для определенного устройства. Они включают в себя:

- Свойство **Адрес хоста** (`address`), определяющее **IP адрес или имя хоста** контролируемого сетевого хоста.
- Настройки опроса 640, используемые для периодического получения текущего статуса SNMP устройства (опрашивает 640 его).

Активы Device

Драйвер создает один корневой актив для каждого MIB файла, поддерживаемого устройством. Поддерживаемые MIB файлы определяются автоматически при опросе SNMP агента.

Активы для MIBs, отмеченные **По умолчанию** в MIB директории, будут включены автоматически.

Настройки Device

Настройки SNMP устройства представляют результаты последней операции опроса 642.

Операции Device

- **Получить переменные SNMP**. Эта операция служит для извлечения необработанных значений SNMP OID из устройства. Используются настройки соединения, определенные в учетной записи устройства.
- **Установить переменые SNMP**. Эта операция служит для ввода необработанных значений SNMP OID в устройство. Используются настройки соединения, определенные в учетной записи устройства.

Адрес операции получить (Get) и установить (Set) полностью индексируется скалярными переменными. Это значит, что вам нужно добавить `.0` в качестве индекса к идентификатору объекта переменной. Например, если вы хотите получить значение `sysUpTime` (идентификатор переменной: `1.3.6.1.2.1.1.3`) от устройства, вам нужно ввести `1.3.6.1.2.1.1.3.0` в качестве значения для параметра идентификатора переменной операции получить. То же самое применимо и для операции установить.

События Device

SNMP устройства формируют два типа событий: ***Время простоя службы (*Service Outage)**, указывающее бездействие SNMP, как описано в главе Драйвер устройства сетевого хоста 595, и **SNMP уведомление** (см. далее).

SNMP УВЕДОМЛЕНИЕ

SNMP драйвер формирует события при получении SNMP уведомлений (traps и informs). для более подробной информации обратитесь к разделу Мониторинг SNMP уведомлений 643.

Так как SNMP драйвер прослушивает уведомления **всех** сетевых хостов (не только тех, которые соотнесены к определенной учетной записи устройства) 497, события **Trap** формируются в контексте **Сетевое управление** вместо контекста 1494 Device.

Имя события	trap
Права доступа	Доступен на уровне 486 доступа *пользователя.*
Записи	1

Trap-событие имеет следующий формат:

Имя поля	Тип поля	Примечание
agentAddress	строка	Адрес SNMP агента, т.е. IP адрес обекта, формирующего ловушку (trap).
version	строка	Версия ловушки.
type	строка	Тип ловушки (**Trap** или **Inform)**
enterprise	строка	Организация, которая создала trap. Представляет собой **идентификатор объекта** (**OID**), относимый к поставщику агента. Совпадает со значением переменной sysObjectID, является уникальным для каждого реализации SNMP агента.
genericTrap	целое	**Идентификатор общего типа Trap (Generic Trap Type ID).** Целое число, см. далее таблицу.
specificTrap	целое	**Идентификатор определенного типа (Specific Trap Type ID)**. Определяется, когда значение поля "genericTrap" установлено на "enterprise".
oid	строка	Trap OID.
timestamp	дата	Затраченное время, измеряемое в сотых долях секунды, после последней реинициализации агента до события, формирующего trap.
variableBindings	таблица данных	Привязки переменной. Дополнительная информация в зависимости от общего типа trap.
engineID	строка	Идентификатор SNMP инициатора ловушки "trap".

Следующие значения **ID общего типа Trap** определяются для поля *genericTrap:*

Значение	Имя	Описание
0	coldStart	Агент в процессе инициализации. Данные конфигурации и/или значения MIB переменной могли измениться. Запустить повторно периоды измерения.
1	warmStart	Агент в процессе инициализации, но данные конфигурации или значения MIB переменных не изменились.
2	linkDown	Агент обнаружил, что сетевой интерфейс был отключен.
3	linkUp	Агент обнаружил, что сетевой интерфейс был включен.
4	authenticationFailure	Полученное сообщение не может быть аутентифицировано.
5	egpNeighborLoss	Соседний объект протокола внешнего шлюза (EGP) был отключен.
6	enterpriseSpecific	Обнаружены идентификаторы некоторых специфичных для производителя типов события.

Подключение

Драйвер переводит службу в режим **Онлайн**, если во время операции SNMP опроса не было обнаружено ошибок.

Синхронизация

Процедура синхронизации выполняет операции SNMP опроса. Для более подробной информации обратитесь к разделу SNMP опрос 640.

10.2.37.1 Опрос сетевого хоста SNMP

SNMP опрос представляет собой синхронный процесс выборки статусов внешнего устройства, используя SNMP операции получения значений.

AtomMind выполняет опрос SNMP в виде процедуры синхронизации 514 Device.

Процедура синхронизации устройства использует параметры, заданные в настройках SNMP 640, для выполнения операций SNMP. Существует два типа синхронизации, которая может быть применима к контролируемым устройствам.

ПОЛНАЯ СИНХРОНИЗАЦИЯ SNMP

Полная синхронизация SNMP включает в себя следующие шаги:

- Проверка SNMP коммуникации с контролируемым устройством. Выполняется посредством подключения к устройству и выборки одной переменной SNMP.
- Чтение *метаданных* устройства и значений переменной при помощи операции SNMP Walk.

SNMP walk является операцией, которая использует SNMP *GETNEXT* запросы для опроса всех доступных переменных контролируемого устройства и их значений.

- Сопоставление полученных метаданных вместе с записями Директории MIB файлов 644 для получения большей информации о каждой настройке (тип, описание и т.д.)
- Конвертирование настроек SNMP устройства в переменные контекста 61 AtomMind Server и создание кэша настроек 502.
- Сохранение полученных данных в кэше настроек. Для более подробного описания того, как результаты мониторинга SNMP представляются драйвером устройства, обратитесь к разделу Результаты опроса SNMP 642.

Служба мониторинга SNMP находится в режиме **Онлайн** в случае успешного выполнения всех операций. Каждая ошибка, обнаруженная во время синхронизации, сохраняется в результатах мониторинга и переводит службу в режим **Офлайн**.

Обратите внимание, что операция SNMP walk предоставляет список переменных SNMP вместе с их значениями. Таким образом, для синхронизации конкретных переменных не нужно выполнять дополнительных действий.

Но с другой стороны, **SNMP walk является длительной операцией**. Поэтому она не выполняется во время первой синхронизации устройства. Следовательно, служба просто проверяет, что контролируемое устройство поддерживает коммуникацию SNMP и быстро возвращает результат в виде статуса службы. Полная синхронизация выполняется сразу после завершения первой синхронизации. Данный метод откладывает длительную синхронизацию для только что добавленных устройств.

ЧАСТИЧНАЯ СИНХРОНИЗАЦИЯ

Если нет необходимости в повторном чтении всех метаданных устройства, но нужно обновить всего лишь несколько известных переменных, полная синхронизация не выполняется. Вместо неё используются операции SNMP *get* для опроса данных переменных, и кэш устройства обновляется соответственно.

10.2.37.1.1 Настройки опроса SNMP

Следующие настройки используются для опроса 640 устройств SNMP:

Настройка	Описание
Порт	Номер порта агента SNMP.
Версия протокола SNMP	Используемая версия протокола SNMP.
Имя сообщества для чтения (Read Community)	Имя сообщества SNMP, используемое во время операций чтения. Обычно используется имя **public.**

<table>
<tr><td>Имя сообщества для записи (Write Community)</td><td>Имя сообщества SNMP, используемое для операций записи. Обычно используется имя private.</td></tr>
<tr><td>Уровень безопасности</td><td>Конфигурация безопасности SNMP v3. Доступны три уровня безопасности:
Аутентификация, приватность (шифрование) выключены.
Аутентификация включена, приватность (шифрование) выключены.
Аутентификация включена, приватность (шифрование) включены.</td></tr>
<tr><td>Имя пользователя</td><td>Имя пользователя для аутентификации SNMP v3 и приватности (шифрования).</td></tr>
<tr><td>Протокол аутентификации</td><td>Аутентификация SNMPv3 на основе MD5 или SHA.</td></tr>
<tr><td>Пароль для аутентификации</td><td>Пароль для аутентификации SNMP v3.</td></tr>
<tr><td>Протокол приватности (Encryption)</td><td>Шифрование SNMPv3 на основе DES или AES.</td></tr>
<tr><td>Пароль приватности (Encryption)</td><td>Пароль шифрования SNMP v3.</td></tr>
<tr><td colspan="2">(Дополнительные настройки:)</td></tr>
<tr><td>Протокол</td><td>Коммуникационный протокол (UDP или TCP).</td></tr>
<tr><td>Количество повторов команды</td><td>Количество повторно отправляемых запросов до неудачного выполнения операции SNMP.</td></tr>
<tr><td>Тайм-аут, миллисекунд</td><td>Время ожидания выполнения запросов SNMP.</td></tr>
<tr><td>Максимальный размер PDU, байт</td><td>Максимальный размер протокольной единицы обмена (PDU), отправленной AtomMind Server. Обычно используются значения 484 (поддерживаемое любым SNMP устройством) и 1472 (поддерживается большинством устройств).</td></tr>
<tr><td>Обрабатываемые данные</td><td>Определяет, какой вид переменных SNMP должен быть прочитан и обработан. Допустимые значения:
<table>
<tr><th>Значение</th><th>Описание</th></tr>
<tr><td>Все значения, доступные в MIB директории</td><td>Используются только те значения, описания которых могут быть найдены в директории MIB файлов 644. Все нераспознанные идентификаторы объектов SNMP (SNMP OIDs) будут игнорироваться.</td></tr>
<tr><td>Все значения, обнаруженные при выполнении операции SNMP walk (снижает быстродействие операции)</td><td>Будут использоваться все значения, обнаруженные при выполнении операции SNMP walk. Это может привести к замедлению синхронизации и увеличению использования памяти сервера.</td></tr>
</table></td></tr>
<tr><td>Группировка настроек</td><td>Определяет представление данных SNMP:
<table>
<tr><th>Значение</th><th>Описание</th></tr>
<tr><td>По MIBs</td><td>Значения сгруппированы по MIB, представляющими логическое группирование данных SNMP.</td></tr>
<tr><td>По OIDs</td><td>Значения сгруппированы по OID, которые представляют иерархическую структуру дерева значений SNMP.</td></tr>
</table></td></tr>
<tr><td>Исходный OID</td><td>OID, используемый для проверки доступности устройства (статус соединения) и инициализации операций SNMP walk во время синхронизации устройства. По умолчанию это .1.3.6.</td></tr>
</table>

	Обратите внимание, что исходный OID определяет не поддерево, а минимальный OID, из которого начинается задержка.
Использовать многозначные запросы для чтения таблиц	Когда опция включена (по умолчанию), многозначные запросы используются, чтобы извлечь данные таблицы. Это быстрее, чем однозначные запросы, но в определенных случаях могут создавать незаконченные данные. В этом случае отключите эту опцию, чтобы извлекать все доступные значения.
Шифрование строки SNMP	Символ шифрования, который используется, чтобы представить строки, полученные от устройства SNMP.
Максимальное значение Request ID	Каждый запрос SNMP обладает своим идентификатором (ID запроса), которым связывается с соответствующими запросами SNMP. Если же максимально значение Request ID не установлено (по умолчанию), то Request ID генерируется в диапазоне положительных 32-разрядных чисел со знаком (то есть 1 к 2147483647 [0x7fffffff]). Некоторые устройства SNMP не принимают большие числа, в этом случае максимальное значение Request ID может быть установлено для лимитирования значений Request ID.

10.2.37.1.2 Результаты опроса SNMP

Статус операции опроса SNMP представлен переменной SNMP (`snmp`), состоящей из следующих полей:

Поле	Описание
Успешно	Указывает, что SNMP коммуникации поддерживаются управляемым устройством и процедура последней синхронизации была выполнена успешно.
Ошибка	Текст ошибки, обнаруженной в процессе последней синхронизации.

Результаты SNMP опроса представлены в виде переменной настроек Device согласно следующим правилам:

- Если группа OIDs представляет таблицу SNMP, для них создается переменная настройки устройства в виде таблицы.
- Во всех других случаях для каждого OID создается одна переменная настройки устройства.

Имена переменных настройки устройства

Имена переменных настройки устройства для переменных SNMP формируются согласно следующим правилам:

- Если для определенного OID найден MIB-символ, имя настройки устройства совпадает с MIB-символом (например, `sysDescr`, `hrStorageTable`).
- Если для определенного OID не найден MIB-символ, имя настройки устройства совпадает с самим OID. Точки заменяются на подчеркивания для создания верного формата имени (например, `1_3_6_1_2_1_5_27_1_5_2_0`).

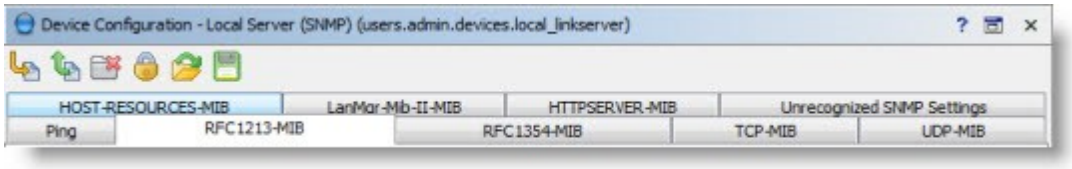
Описания переменных настройки устройства

Описания переменных настройки устройства для переменных SNMP формируются согласно следующим правилам:

- Если соответствующий MIB файл не найден для OID, описание формируется согласно настройке "Описание переменной"[644], заданной для данного MIB-файла.
- Если полные MIB данные не найдены, описание переменной включает имя, состоящее из символов распознанной части OID, за которыми следуют цифры нераспознанной части. Полная переменная OID заключена в скобки. Например: `iso.org.dod.internet.mgmt.mib-2.icmp.27.1.5.1.0 (1.3.6.1.2.1.5.27.1.5.1.0)`

Группировка переменных настройки устройства

Переменные настройки устройства группируются MIB-файлами, в которых определены соответствующие OID. Переменная настройки ненайденного в MIB директории идентификатора объекта входит в группу "**Нераспознанные идентификаторы объектов SNMP**".



Конверсия типа SNMP в тип AtomMind

Типы SNMP конвертируются в типы 49 AtomMind, как определено в следующей таблице:

Тип SNMP	Тип AtomMind
OctetString	строка
Counter32	длинное
Counter64	строка
Integer (Integer32)	целое
UnsignedInteger32 (Gauge32)	длинное
Counter32	длинное
IpAddress	строка
TimeTicks	длинное
Object Identifier	строка

10.2.37.2 Мониторинг SNMP уведомлений

SNMP уведомления представляют собой асинхронные уведомления, которые AtomMind SNMP отправляет менеджеру.

Существует два типа запросов уведомлений, определяемых стандартами SNMP: **Trap** запросы и **Inform** запросы. AtomMind Network Manager поддерживает оба типа запросов уведомлений.

Мониторинг SNMP уведомлений контролируется опцией Общая конфигурация 643 драйвера SNMP. Включение опции "**Получать SNMP уведомления"** позволяет прослушивать уведомления. Если данная опция включена, AtomMind Server прослушивает SNMP уведомления определенного порта (см. **Порт прослушивания SNMP уведомлений**). Будут получены уведомления от любого устройства SNMP при использовании протокола UDP или TCP в зависимости от **Транспортного протокола уведомлений SNMP.** Для безопасных уведомлений SNMP версии 3 будет использоваться определенный ID локального устройства.

Драйвер SNMP может получать SNMP уведомления от агентов SNMP и конвертировать их в события контекста 73, как описано здесь 595. Запись, ассоциируемая с событием AtomMind, представляет свойства уведомления.

Данные событий уведомлений SNMP могут быть использованы для контроля важных событий в вашей сети, их анализа и своевременной реакции.

10.2.37.2.1 Общая конфигурация мониторинга SNMP уведомлений

Драйвер SNMP включает следующие **общие настройки мониторинга уведомлений SNMP :**

Поле	Описание
Получение уведомлений SNMP	Если включено, AtomMind Server получает SNMP уведомления и формирует trap события.
Транспортный протокол уведомлений SNMP	Протокол для доставки уведомлений SNMP: UDP или TCP.
Порт прослушивания уведомлений SNMP	Номер порта, используемого для прослушивания уведомлений SNMP.
ID локальной библиотеки доступа	ID локальной библиотеки доступа для уведомлений SNMP версии 3.

10.2.37.2.2 Автоматическое обнаружение источников SNMP уведомлений

Драйвер SNMP включает в себя опцию **Автоматического обнаружения источников уведомлений SNMP.** Когда данная опция включена, AtomMind Server автоматически *обнаружит* все устройства, которые отправили trap- уведомления SNMP. Должен быть загружен *плагин "Обнаружение сети"*. Для получения более подробной информации о *мониторинге уведомлений SNMP* смотрите уведомления SNMP 644.

10.2.37.3 Общие настройки SNMP

Драйвер SNMP включает в себя следующие общие настройки SNMP:

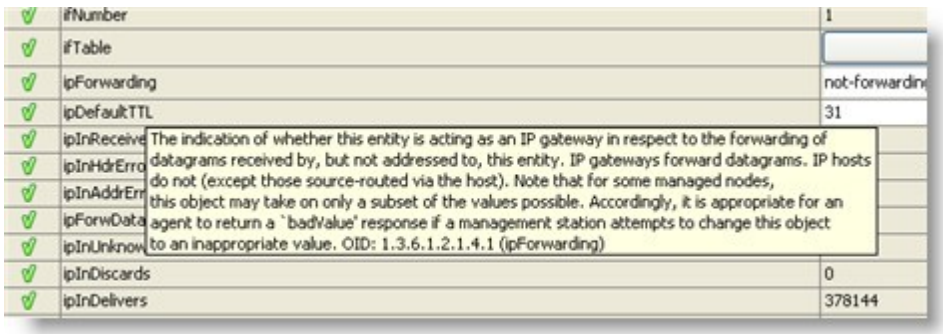
- настройки мониторинга уведомлений SNMP 643
- таблицу директории MIB-файлов 644
- таблицу пользователей SNMP 645
- настройку автоматического обнаружения Trap уведомлений SNMP 644

10.2.37.3.1 Директория MIB-файлов

Агенты SNMP не предоставляют полные метаданные о своих свойствах (SNMP идентификаторах объекта). Большинство данных, предоставляющих настройки устройств SNMP, извлекаются из файлов Management Information Base (MIB).

MIB (**Management Information Base**) файлы содержат подробную информацию об объектах, предоставленную устройствами SNMP.

Драйвер устройства SNMP имеет встроенное хранилище MIB файлов. Таблица **директории MIB файлов** позволяет управлять данным хранилищем. Таблица содержит следующие поля:

Поле	Описание
Имя переменной SNMP	Описание MIB-файла.
MIB файл	Данные MIB-файла.
Описания переменной	Определяет формирование описаний переменных настройки устройства. Три варианта на выбор: • По имени MIB символов (по умолчанию) • По первому предложению описания из MIB-символов (предложения должны разделяться точками). • По первой строке описания из MIB символов. Правильное значение данной настройки зависит от структуры MIB файла и может отличаться для MIB файлов аппаратных средств других производителей. Значение по умолчанию данной настройки отображает имя в MIB-символах в описании переменной, но можно всегда посмотреть полное описание в символах, проверив "помощь" (подробное описание) переменной. Далее приведен скриншот из AtomMind Client 359: 
Включено	Если данная опция выключена, MIB-файл не загружается при запуске сервера и не используется для получения информации о настройках устройства.

	Если другие MIB-файлы зависят от отключенного MIB-файла, их загрузка может быть не выполнена.
По умолчанию	Если данная опция включена, MIB обозначена по умолчанию. Она будет включена автоматически в списке активов.
Загружено	Флажок "только для чтения", указывающий, успешно ли загрузился MIB и используется драйвером.
Ошибки загрузки	Данное поле содержит ошибки в случае их возникновения в процессе загрузки и компилирования MIB.

MIB-файлы, определенные как "загруженные" в данной таблице, используются драйвером SNMP для получения информации о настройках и trap-уведомлениях устройства SNMP.

Встроенные MIB-файлы

Для мониторинга SNMP необходимо более двухсот MIB-файлов, которые в связи с этим привязаны к дистрибутиву AtomMind Server. Они содержат информацию о MIB объектах контролируемых устройств SNMP, таких как роутеры, серверы, сетевые принтеры и т.д.

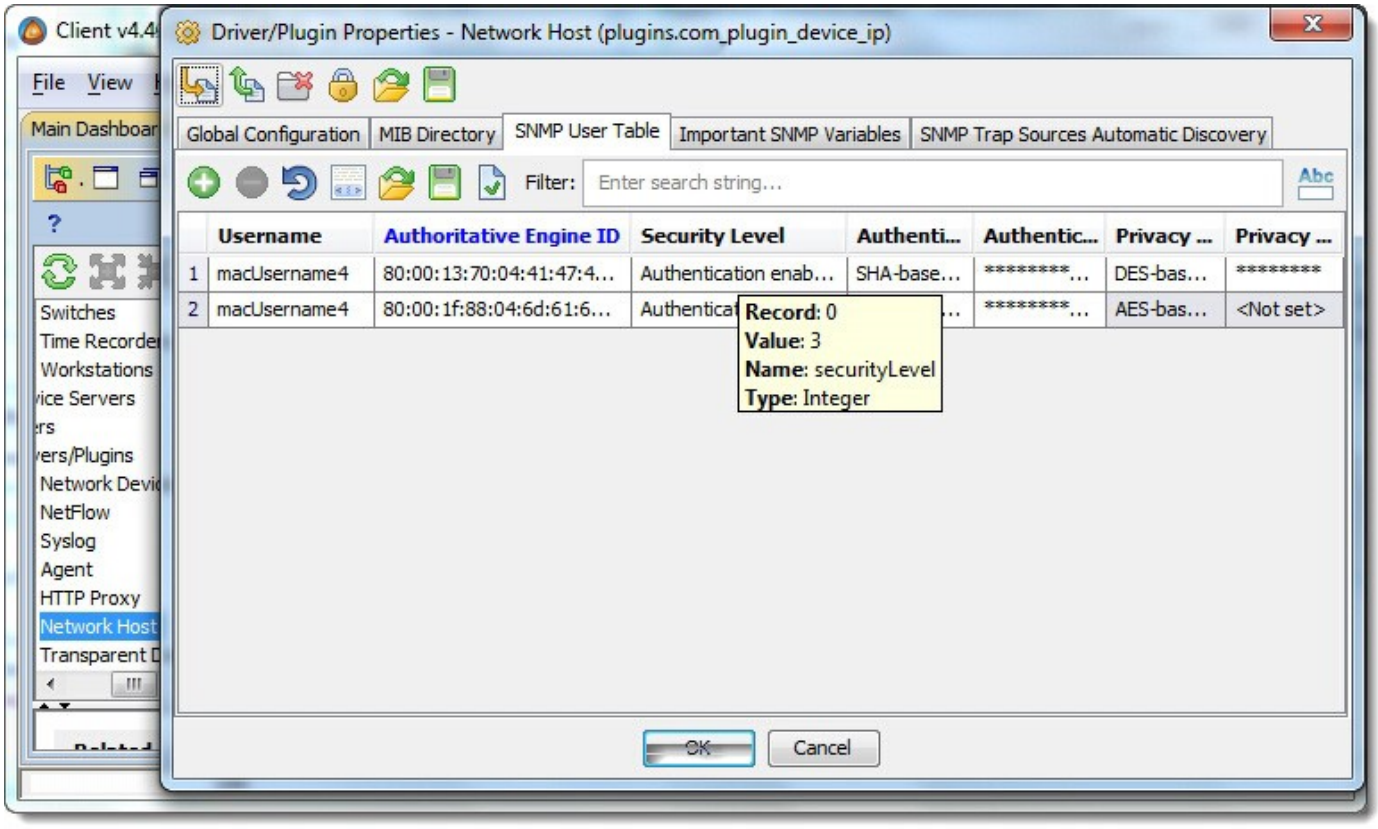
Автозагрузка MIB-файлов

Драйвер устройства SNMP может автоматически загрузить один или более MIB-файл с диска и добавить их в подкаталог `/mib` установки AtomMind Server и запустить/перезапустить сервер. MIB-файлы будут добавлены в базу данных MIB и использованы драйвером.

10.2.37.3.2 Таблица пользователей SNMP

Таблица пользователей SNMP представляет собой безопасную базу данных пользователя, применимую для коммуникации через протокол SNMP версии 3.

Как указано в стандартах SNMPv3, информация пользователя сочетает в себе Имя пользователя и Идентификатор устройства, определяющие аутентичное приложение SNMP в диалоговом окне.



Информация в таблице пользователя SNMP представлена в следующем формате:

Поле	Описание
Имя пользователя	Имя пользователя
Идентификатор устройства	ID устройства. Данное поле можно оставить пустым. В таком случае AtomMind Network Manager попытается его установить, как указано в стандарте SNMPv3.
Уровень безопасности	Один из трех стандартных уровней безопасности: • Аутентификация выключена, приватность (шифрование) выключена. • Аутентификация включена, приватность (шифрование) выключена. • Аутентификация включена, приватность (шифрование) включена.
Протокол аутентификации	Аутентификация SNMPv3 на основе **MD5** или **SHA.**
Пароль аутентификации	Пароль аутентификации SNMPv3.
Протокол шифрования	Шифрование SNMPv3 на основе **DES** или **AES.**
Пароль шифрования	Пароль шифрования SNMPv3.

10.2.38 SOAP (веб сервисы)

Драйвер устройства 518 SOAP (Simple Object Access Protocol) позволяет AtomMind Server вызывать Web-сервисы. Объекты Web-сервисов, посланные Web-сервисом и полученные от Web-сервиса, конвертируются из/в таблицы данных 49 AtomMind. Обратитесь к разделу Devices для получения более детальной информации о "нормализованном" представлении устройств в AtomMind.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.soap

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Device Account Properties

СВОЙСТВА СОЕДИНЕНИЯ

Настройки соединения определяют, как AtomMind Server взаимодействует с определенным сервером SOAP. Эти настройки доступны через опцию изменить свойства 1494 Device контекста Device. Доступны следующие свойства подключения:

Настройка	Описание
URL WSDL	URL файла WSDL, описывающий один или более Web-сервисов.
Таймаут выполнения операции	Указывает максимальное количество времени ожидания для завершения выполнения операции.
Движок SOAP	Фреймворк, используемый для реализации SOAP клиента.
URL сервиса	URL конечной точки Web-сервиса.
Тип аутентификации	Метод аутентификации: Basic или NTLM.
Имя пользователя	Имя пользователя, используемое при авторизации на веб сервере.

Пароль	Пароль, используемый при авторизации на веб сервере.
Доверять всем сертификатам	Если true, проверка сертификата будет отключена. В ином случае, все сертификаты сервера будут проверяться с использованием файла хранилища сертификатов. Не устанавливайте это свойство, если вы собираетесь работать в сети, которой вы полностью не доверяете. Особенно если что-то будет происходит в рамках общего доступа к интернету.
Путь к хранилищу сертификатов	Путь к файлу хранилища сертификатов , содержащий клиентские сертификаты. Это локальный путь файловой системы на устройстве, на котором работает AtomMind Server.
Пароль к хранилищу сертификатов	Пароль, дешифрующий файл хранилища сертификатов.
Отключить проверку CN (Common Name)	Устанавливает, должно ли устройство отключать проверку, если имя хоста, указанного в URL, совпадает с Common Name (CN) на сертификате сервера.
Класс заголовка	Имя класса Qualified Java, представляющее SOAP заголовок.
Параметры операции	Имя и параметры SOAP операции.

Активы Device

Драйвер не поддерживает активы.

Настройки Device

Драйвер не предоставляет настроек.

Операции Device

Драйвер создает функцию устройства для каждой SOAP операции (функции), которую можно найти в указанном WSDL файле. Любой вызов функции Device соединен с вызовом функции SOAP. AtomMind Server использует таблицу данных ввода функции для конструирования SOAP объектов, которые обслуживаются как параметры ввода. Полученные в результате SOAP объекты конвертируются в таблицу данных вывода.

События Device

Драйвер не предоставляет событий.

Подключение

Драйвер не устанавливает/не тестирует соединения. Любое SOAP устройство всегда находится **онлайн**, несмотря на то, что отдельные SOAP вызовы могут дать сбой из-за неполадок сети.

Синхронизация

Драйвер не предпринимает каких-либо действий при периодической синхронизации.

10.2.39 База данных SQL

Драйвер базы данных SQL обеспечивает метод контроля любой JDBC-совместимой системы управления базами данных (СУБД). Практически все современные серверы базы данных предоставляют Java Database Connectivity (JDBC) драйверы, таким образом, управление и контроль над ними может осуществляться через AtomMind.

Драйвер обладает следующими возможностями:

- Проверка доступности и статуса сервера базы данных;
- Выполнение запросов выбора и обеспечение доступа к результатам запроса;
- Выполнение быстро формирующихся запросов вставить/обновить/удалить.

Необходимо добавить хотя бы один "проверочный" запрос для проверки работоспособности базы данных. Если запросы не добавляются, драйвер установит соединение при первом цикле синхронизации 514 и ничего не сделает во время последующих синхронизаций для предоставления данных о доступности и работоспосоности базы данных.

Всегда можно использовать очень простой тестовый запрос, такой как `SELECT 1` или `SELECT 1 FROM table`.

Драйвер использует пул соединения, чтобы улучшить производительность благодаря возможности параллельно выполнять несколько запросов. См. свойства "Пул соединения ..." в разделе Свойства соединения БД ниже.

Возможно протестировать объединенные соединения, используя две расширенные настройки: **Проверка тестового соединения** и **Тестовый период незадействованного соединения**. Эти настройки можно использовать либо вместе, либо независимо дург от друга, чтобы минимизировать вероятность просроченного или разорванного соединения. Тестирование отключено по умолчанию. Не изменяйте эти настройки до тех пор, пока вам это не понадобится, тогда вы поймете, что делать.

Настройка **Максимальное свободное время** определяет, когда объеденненное свободное соединение отклонено. Эта настройка должна быть установлена на значение ниже, чем настройка **wait_timeout** соотвествующего сервера SQL (для MySQL, если **wait_timeout** не находится непосредственно в файле конфигурации (my.ini or my.cnf), его значение по умолчанию 8 часов).

Добавление драйверов базы данных JDBC

Чтобы позволить AtomMind Server загружать сторонний драйвер базы данных JDBC, поставляемый вашим поставщиком сервера базы данных, файл JAR (*Java* Архив), содержащий этот драйвер, должен добавляться к *classpath* сервера в поддиректорию `/lib` инсталляции AtomMind Server или путем использования файла свойств загрузчика 162 AtomMind Server.

Драйвер базы данных SQL может также использоваться для соединения с источниками данных Open Database Connectivity (ODBC) через мост JDBC-ODBC, являющейся частью инсталляции AtomMind Server. В этом случае не добавляется дополнительных драйверов. The **Database URL** syntax to use is `jdbc:odbc:odbc_data_source_name`.

ВЕРСИЯ JAVA

Большинство дистербьютеров AtomMind Server включают в себя связанную JVM. Любые виртуальные машины Java VMs, установленные в ОС, не используются по умолчанию. Поэтому любой пользовательский драйвер JDBC должен быть скомпилирован для версии Java, которая используется на сервере. Эту версию можно найти в системных требованиях 146.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.database

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

СВОЙСТВА ПОДКЛЮЧЕНИЯ К БАЗЕ ДАННЫХ

Свойство	Описание
URL база данных	Данная строка базы данных определяет тип базы данных, путь (локальный или сетевой) к базе данных для выполнения запросов и

	любых дополнительных опций. Чтобы установить требуемое значение для выбранного вами драйвера базы данных JDBS, пожалуйста, обратитесь к прилагаемой к нему документации.
Класс драйвера	Имя класса Java драйвера JDBC. Можно найти в документации драйвера JDBC. Требуется только для драйверов, которые не совместим с JDBC4.
Имя пользователя базы данных	Данное свойство определяет, какое имя пользователя будет использоваться для входа на сервер базы данных.
Пароль базы данных	Данное свойство определяет пароль, используемый для входа на сервер базы данных.
Дополнительные свойства	Таблица, содержащая дополнительные свойства соединения базы данных. В таблице две колонки: **Свойство** и **Значение**. Список поддерживаемых свойств можно найти в документации драйвера JDBC.
Минимальный размер запросов пула	Минимальное число соединений пула будет сохраняться в любое данное время.
Максимальный размер запросов пула	Максимальное число соединений пула будет сохраняться в любое данное время.
Увеличение пулов запроса	Определяет, сколько соединений в это время пул будет пытаться получить, когда будет исчерпан.
Максимальное неиспользуемое время	Время, отброшенное перед объединенным свободным соединением (0 обозначает, что свободное соединение никогда не истечет).
Проверка тестового соединения	Если установлен на "истина", тест соединения будет произведен асинхронно при каждой проверке соединения, чтобы проверить, что соединение валидно.
Период теста свободного соединения	Выделенные свободные соединения будут тестироваться асинхронно каждый определенный период времени (0 значит никогда).

ЗАПРОСЫ

Данная табличная настройка определяет, какие запросы будут выполняться при помощи сервера базы данных.

Поле	Описание
Имя	Имя переменной настройки устройства, содержащей результат запроса.
Описание	Описание переменной настройки устройства, содержащей результат запроса.
Выражения	Заставляет обрабатывать запросы Чтение и Запись в виде выражений 112. Данные выражения должны быть реальными SQL запросами, разрешенными в базе данных.
Запрос для чтения	Текст SQL запроса SELECT или текст выражения, если установлен флажок **Выражения**. Результат выполнения запроса доступен в AtomMind в качестве переменной настройки устройства.
Запрос для записи	Текст SQL запроса INSERT/UPDATE/DELETE или текст выражения, если установлен флажок **Выражения**. Если данное поле не пустое, переменная, содержащая результат выполнения **запроса на чтение**, будет доступна для записи (изменения). Данная модификация запустит выполнение запроса на запись. Если запрос на запись является выражением, он может ссылаться на Таблицу данных, содержащую результат запроса на чтение. Она является *таблицей по умолчанию* для ссылок 117 внутри данного выражения.

Активы Device

Активы не поддерживаются этим драйвером.

Настройки Device

Драйвер устройства базы данных создает одну переменную настройки устройства для каждой записи таблицы запросов. Формат данной переменной динамичен в зависимости от результата выполнения запроса на чтение.

Помимо этого создается переменная **статистики базы данных**, которая содержит следующие поля:

- Имя базы данных
- Версия базы данных
- Имя драйвера базы данных
- Версия драйвера базы данных

Операции Device

ВЫПОЛНЕНИЕ ЗАПРОСА

Данное действие позволяет вам выполнять произвольные запросы базы данных. Поддерживаются запросы "выбрать" и "обновить".

Поля входа функции:

Поле	Описание
Запрос	Текст запроса. Может содержать ссылки параметра запроса, встроенные как символ ?.
Обновление	Флажок, который определяет, это запрос "выбрать" или "обновить".
Параметры	Таблица параметров запросов. Значение ячейки в первом ряду и в первой колонке будет заменять первый параметр запроса (первое появление символа ? в тексте запроса), значение яйчейки в первом ряду и второй колонке идет во второй параметр и так далее.

События Device

Драйвер не предоставляет события.

Подключение

Данный драйвер приводит устройство в режим **Онлайн**, если JDBC соединение было выполнено успешно.

Правила преобразования типа данных

Когда драйвер JDBC, который выполняет запрос, преобразовывает типы данных SQL в типы JDBC, тогда AtomMind Server преобразовывает типы JDBC в типы поля 52 AtomMind, следуя правилам:

Тип JDBC	Тип поля
TINYINT	Целочисленное поле
SMALLINT	Целочисленное поле
INTEGER	Целочисленное поле
BIGINT	Длинное поле
VARCHAR	Строковое поле
BOOLEAN	Булевое поле
BIT	Булевое поле
NUMERIC	Длинное поле
REAL	Поле с плавающей запятой
DOUBLE	Двойное поле

DECIMAL	Поле с плавающей запятой
TIMESTAMP	Временное поле
LONGVARCHAR	Поле таблицы данных
LONGNVARCHAR	Поле таблицы данных
BLOB	Поле таблицы данных
LONGVARBINARY	Поле таблицы данных

Информацию о преобразованиях между типами данных SQL и типами JDBC можно найти в документации драйвера JDBC. Эти соответствия влияют на конечные типы во время выполнения запросов.

В некоторых случаях отдельный тип данных SQL может быть представлен разными форматами поля AtomMind. Например, драйвер JDBC для Microsoft SQL Server представляет `nvarchar(n)` как `VARCHAR` и `nvarchar(max)` как `LONGNVARCHAR` (см. подробнее в статье Использование дополнительных типов данных). А AtomMind Server преобразует эти типа в строковое поле и поле таблицы данных соотвественно.

10.2.40 Спутниковый контроль транспорта

Драйвер спутникового контроля транспорта (Satellite vehicle tracker driver) используется для мониторинга и контроля различных контроллеров GPS/GLONASS при помощи общего независимого от поставщика подхода. Драйвер позволяет применять гибкие правила для парсинга сообщений трекера.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.tracker

Общие настройки

Общая конфигурация драйвера включает в себя таблицу **Конфигурация взаимодействия с трекерами и обработки данных**:

Поле	Описание
Тип	Идентификатор типа трекера может содержать только английские буквы, цифры и нижние подчеркивания. Тип выбирается для каждого устройства в настройках аккаунта.
Описание	Удобное для восприятия описание типа трекера. Тип выбирается для каждого устройства в настройках аккаунта.
Протокол	Протокол(-ы), который использует трекеры данного типа для взаимодействия с сервером: **TCP**, **UDP** или оба.
Порт подключения TCP	Порт для прослушивания входящих TCP подключений от трекеров данного типа.
Порт подключения UDP	Порт для прослушивания входящих UDP подключений от трекеров данного типа.
Разделитель сообщения	Символ или строка, которая разделяет сообщения трекера в TCP потоке. В UDP потоке каждое сообщение обычно посылается в отдельных дейтаграммах. Однако, несколько сообщений на одну дейтаграммму тоже поддерживаются.
Правила парсинга	Правила 743, использованные для парсинга строки сообщения трекера в таблицу данных 49. Таблица, определяющая последовательный набор правил 745, которые должны иметь результатом таблицу. Выражения набора правил должны посылать `command` переменная среды 123, которая содержит текст сообщения трекера. Ссылки текста команды будут иметь следующую форму: `{env/command}` Итоговая таблица, возвращенная набором правил, должна содержать сырые значения для идентификатора устройства (обычно номер IMEI), тип сообщения трекера и параметры сообщения (напр. широта и долгота).

<table>
<tr><td>Выражение идентификатора устройства</td><td>Выражение[112], которое должно вернуть идентификатор устройства, уникально в инсталяции AtomMind Server. Обычно IMEI номер трекера используется в качестве идентификатора. Этот ID будет использован для нахождения соответствующего аккаунта устройства, который имеет такой же ID, определенный в его свойствах. Все оставшиеся данные сообщений будут сопоставлены с аккаунтом этого устройства.
<table>
<tr><td colspan="2">Среда вычисления выражения[114] ID устройства</td></tr>
<tr><td>Контекст по умолчанию[119]</td><td>Отсутствует.</td></tr>
<tr><td>Таблица данных по умолчанию[120]</td><td>Таблица, возвращенная правилами парсинга.</td></tr>
<tr><td>Ряд по умолчанию[119]</td><td>0</td></tr>
<tr><td>Переменные среды[123]</td><td>Только стандартные[123] переменные.</td></tr>
</table></td></tr>
<tr><td>Выражение имени переменной</td><td>Выражение[112], которое должно вернуть имя переменной настроек устройства[501], содержащее сообщения данного трекера. Оно обычно отсылает тип сообщения трекера.
<table>
<tr><td colspan="2">Среда вычисления[114] Выражения имени переменной</td></tr>
<tr><td>Контекст по умолчанию[119]</td><td>Контекст устройства трекера.</td></tr>
<tr><td>Таблица данных по умолчанию[120]</td><td>Таблица, возвращенная правилами парсинга.</td></tr>
<tr><td>Ряд по умолчанию[119]</td><td>0</td></tr>
<tr><td>Переменные среды[123]</td><td>Только стандартные[123] переменные.</td></tr>
</table></td></tr>
<tr><td>Переменные устройства</td><td>Таблица, определяющая переменные настроек устройства[501], которые будут добавлены к каждому аккаунту устройства при помощи типа трекета. Каждая переменная определена:
Именем
Описанием
форматом[50]
Если определенный формат не описывает ни одно поле, он принимается за динамический, и любая таблица данных, возвращенная правилами парсинга будет использована как валидное значение переменной.</td></tr>
<tr><td>Выражение широты</td><td>Как только тип аккаунта устройства трекера определен или изменен, это выражение вписывается в поле выражение широты таблицы свойств типовых устройств[510]. В дальнейшем оно используется для оценки широты устройства, поэтому обычно оно ссылается на одну из переменных устройств.</td></tr>
<tr><td>Выражение долготы</td><td>Аналогично вышесказанному, повторно определяется выражение долготы аккаунта устройства, как только тип трекера изменен.</td></tr>
</table>

Настройки уровня пользователя

Не определены.

Свойства Device

- **Тип**. Один из типов трекера, определенных в таблице общих конфигураций **Конфигурация взаимодействия с трекерами и обработки данных**.
- **ID устройства**. Уникальный идентификатор трекера, обычно это IMEI номер. Этот ID должен быть получен в каждой команде трекера.
- **Интервал статуса оффлайн**. Если от трекера не было получено сообщений дольше, чем заданное время, аккаунт устройства трекера будет переключен в режим оффлайн.

Активы Device

Драйвер не поддерживает активы.

Настройки Device

Аккаунт спутникового трекера предоставит одну переменную для каждой записи подтаблицы **переменных устройств**, содержащейся в таблице **Конфигурация взаимодействия с трекерами и обработки данных**, соответствующей выбраному типу трекера.

Имена, описания и форматы переменных определены в таблице **переменные устройств**.

Операции Device

Драйвер не предоставляет операций.

События Device

Драйвер не предоставляет событий.

Подключение

Драйвер переводит устройство в режим **онлайн**, если какое-либо сообщение получено и успешно сопоставлено с аккаунтом данного трекера, т.е. одна **переменная устройства** была успешно обновлена.

Драйвер переключается обратно на режим **оффлайн**, если не было получено сопоставленных сообщений дольше, чем **интервал статуса оффлайн**.

Синхронизация

Драйвер не производит синхронизацию. Вместо этого, он следует следующей схеме:

- Драйвер прослушивает каждый порт, определенный в таблице **Конфигурация взаимодействия с трекерами и обработки данных**.
- Как только принято TCP подключение или получена UDP дейтаграмма, драйвер разделяет входящие данные на сообщения, используя **разделитель сообщения**.
- Как только получено полное сообщение, драйвер обрабатывает **правила парсинга** для формирования *таблицы команд*. Если обработка правила прерывается, ошибка регистрируется внутри контекста общей конфигурации драйвера спутникового контроля транспорта.
- Таблица комманд оценивается **выражением идентификатора устройства**.
- Драйвер пытается найти аккаунт устройства, чей **ID устройства** соответствует ID, которое вернуло предыдущее выражение. Если драйверу это не удается, регистрируется ошибка внутри контекста общей конфигурации драйвера спутникового контроля транспорта, и обработка сообщения прерывается.
- Драйвер оценивает **выражение имени переменной** для выявления того, что переменная настроек устройств должна быть использована при помощи таблицы комманд. Если это выражение возвращает null или некорректное имя, обработка прерывается и ошибка регистрируется внутри контекста аккаунта устройсва.
- Таблица команд используется как новое значение для переменной, т.е. вписанной в аккаунт устройства трекера.

10.2.41 TPS (Tibbo Project System)

The TPS Device Driver 518 allows AtomMind Server to read and write GPIO values and directions from LTPS. This driver is avaliable only in AtomMind for Linux TPS.

Драйвер устройства 518 TPS позволяет AtomMind Server читать и изменять линиии ввода/вывод общего назначения из LTPS. Драйвер доступен только AtomMind для Linux TPS.

Информация о драйвере

ID плагина драйвера: com.tibbo.linkserver.plugin.device.tps

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

Не определены.

Активы Device

Не определены.

Настройки Device

Драйвер устройства TPS создает одну переменную настройки Device для каждого значения и направления линиии ввода/вывод общего назначения:

Variable Name	Variable Description	Comments
valueS1A	S1A	Содержит значения сокета S1A (true or false)
...	...	...
valueS25C	S25C	Содержит значения сокета S25C (true or false)
directionS1A	S1A	Содержит направление сокета S1A (input or output)
...	...	...
directionS25C	S25C	Содержит направление сокета S25C (input or output)

Операции Device

Драйвер не проводит операции.

События Device

Драйвер не представляет события.

Синхронизация

Синхронизация между AtomMind Server и устройством TPS включает в себя следующие шаги:

- Чтение полученных значений и направлений линий ввода/вывода общего назначения и хранение этих значений в кэше настроек.

10.2.42 Виртуальное устройство

Драйвер Виртуального устройства является драйвером устройства, используемым для создания и управления виртуальными устройствами, т.е. теми, которые не являются аппаратным оборудованием, но распознаются AtomMind как таковые. Каждое виртуальное устройство представлено контекстом устройства и обладает настройками (переменными данного контекста), операциями (функциями и действиями данного контекста) и событиями. Виртуальное устройство является очень полезным во время тестирования, устранения ошибок и изучения системы, т.к. его поведение схоже с поведением аппаратного устройства, подключенного к системе.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.virtual

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

НАСТРОЙКИ СОЕДИНЕНИЯ

Свойство	Описание
Расширенный	Виртуальные устройства с этим включенным свойством предоставляют доступ к расширенному набору переменных устройства.

ТЕСТИРОВАНИЕ ПРОИЗВОДИТЕЛЬНОСТИ

Определяет, как создаются переменные, связанные с тестированием производительности системы.

Свойство	Описание
Количество тестовых переменных	Определяет, как много переменных тестирования производительности будет создано в виртуальном устройстве. Каждые переменные содержат три поля: случайно сгенерированное значение данных плавающей точки, временная метка значения и случайно сгенерировнное целочисленное значение, эмулирующее качество значения данных.
Период асинхронного обновления	Определяет, как часто каждая переменная тестирования производительности будет обновляться по инициативе виртуального устройства. Нулевое значение показывает асинхронные обновления, позволяющие системному ядру опросить переменные производительности.

Активы Device

Драйвер не поддерживает активы.

Настройки Device

Виртуальное устройство обладает различными типами настроек, которые помогают тестировать тревоги 779, отчеты 928, виджеты 943 и другие элементы системы. Далее приведен список доступных настроек (переменных 61 контекста Device):

ОСНОВНЫЕ СВОЙСТВА

Имя настройки (переменной)	Комментарии
normal	Стандартная настройка строка.
readonly	Настройка строка, доступная только для чтения.
selvals	Настройка строка с значениями выборки.
eselvals	Настройка строка со значениями расширенной выборки.
nullable	Настройка строка, допускающая значение NULL.

НАСТРОЙКИ РАЗЛИЧНЫХ ТИПОВ

Имя настройки (переменной)	Комментарий
string	Настройка строка.
int	Настройка целое.

boolean	Настройка логическое.
float	Настройка плавающая точка.
date	Настройка дата.
table	Настройка в виде таблицы с двумя полями (целое и строка) и неограниченным количеством записей.

МЕСТОПОЛОЖЕНИЕ

Имя настройки (переменной)	Комментарии
track	GPS трек в формате GPX. Данный трек будет "перепроигрываться" устройством, т.е. новая путевая точка (взятая из первого трека, найденного в файле) будет вписана в настройку Местоположение после каждой синхронизации.
location	Имитируемое местоположение устройства (широта и долгота). Местоположение не определяется, если GPS трек не был загружен или его "перепроигрование" было закончено.

НАСТРОЙКИ ВОЛН (WAVE)

Настройки чтения/записи в данной группе определяют, как различные значения чтения/записи формируются в группе **Волны (Waves)** (период, амплитуда и т.д.).

Имя настройки (переменной)	Комментарии
sawtoothSettings	Настройки источника пилообразного колебания.
triangleSettings	Настройки источника треугольных волн.
squareSettings	Настройки источника прямоугольных волн.
sineSettings	Настройки источника синусоидальных волн.
randomSettings	Настройки источника случайных волн.

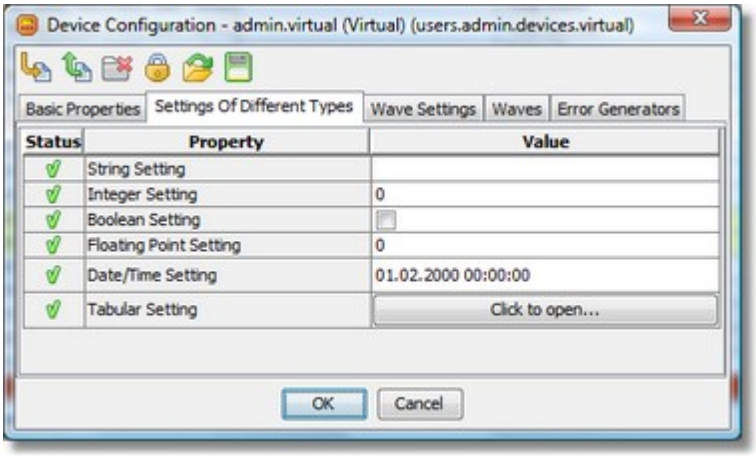
ВОЛНЫ

Имя настройки (переменной)	Комментарии
sawtooth	Источник пилообразного колебания.
triangle	Источник треугольных волн.
square	Источник прямоугольных волн.
sine	Источник синусоидальных волн.
random	Источник случайных волн.

ГЕНЕРАТОРЫ ОШИБОК

Имя настройки (переменной)	Комментарии
shouldGenerateError	**Необходимо сформировать ошибку.** Данная настройка определяет, возникла ли ошибка во время чтения/записи настроек Генератора Ошибок.
errorGenerator	**Генератор ошибок.** Операции чтения/записи данной настройки могут быть не выполнены в зависимости от значения настройки Необходимо Сформировать Ошибку.

Конфигурация виртуального устройства (см. действие конфигурировать 1495 в контексте Device) выглядит как в AtomMind Client:



Операции Device

ФОРМИРОВАНИЕ СОБЫТИЙ

Контекст виртуального устройства имеет функцию **generateEvent** и соответствующее действие **вызова функции** 103 **формирования события.** Данная функция состоит из трех полей: строковое поле **type**, строковое поле **str**, целочисленное поле **int**. Данная функция создает событие с именем, определяемым полем **type** (`Event 1` или `Event 2`, см. ниже). Значения полей в данных о событии взяты из входных параметров функции.

РАСЧЕТ

Функция **расчет** и соответствующее ей действие **вызова функции** 103 расчета обеспечивает проверку для выполнения простых операций (сложения, вычитания, умножения и деления) между двумя аргументами. Данная функция включает в себя три поля: "плавающее" поле **leftOperand,** "плавающее" поле **rightOperand,** строковое поле **operation**. Данная функция возвращает таблицу данных с единственной записью и "плавающим" полем **result.**

События Device

Контекст виртуального устройства включает в себя два события 73, которые называются **event1** (его описание - **Device Event #1**) и **event2** (**Device Event #2**). Данные события состоят из двух полей: строкового поля **str** и целочисленного поля **int.**

Возможно совместное использование данных событий и функции **generateEvent** для тестирования операций устройства и событий.

Подключение

Виртуальное устройство всегда находится в режиме **Онлайн**.

10.2.43 VMware

Драйвер устройства VMware реализует получение данных со счётчиков производительности VMware с помощью программного интерфейса VMware SOAP.

Для работы с последней версией продукции VMware, должна использоваться последняя версия AtomMind Server.

Информация о драйвере

ID плагина 518 **драйвера:** com.tibbo.linkserver.plugin.device.vmware

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

НАСТРОЙКИ ПОДКЛЮЧЕНИЯ

Свойство	Описание
URI службы vSphere (с учетом регистра)	URL веб-сервиса VMware.
Имя пользователя	Имя пользователя для авторизации.
Пароль	Пароль для авторизации.
Включена SSO-авторизация (только для vCenter)	Эта опция включает SSO(Single Sign On)-авторизацию. Она должна быть включена для подключения vCenter, и отключена в любом другом случае (напр. для сервера ESX).
Пропустить проверку SSL сертификатов	Отключает проверку пригодности сертификата сервера.

Активы Device

Драйвер не поддерживает активы.

Настройки Device

- **Сервис**. Предоставляет общую информацию о сервере VMware.
- **Гипервизоры**. Содержит список гипервизоров.
- **ВМ**. Содержит список виртуальных машин.
- **Счетчики**. Содержит список всех счетчиков производительности.

Операции Device

- **Включить/Возобновить ВМ**. Включает или возобновляет виртуальную машину.
- **Выключить ВМ**. Выключает виртуальную машину.
- **Перезагрузить ВМ**. Перезагружает виртуальную машину.
- **Приостановить ВМ**. Приостанавливает виртуальную машину.
- **Получить свойства**. Эта операция позволяет получить особое свойство из устройства VMWare. Есть следующие свойства:
  - **Тип**. Тип управляемого объекта.
  - **Имя**. Имя экземпляра управляемого объекта.
  - **Путь свойства**. Имя или путь свойства, отделенные точкой.

События Device

Драйвер не предоставляет событий.

Подключение

Драйвер переводит устройство в режим **онлайн** если было успешно произведено подключение к серверу VMware.

Синхронизация

Синхронизация между AtomMind Server и сервером VMware включает в себя чтение таблицы показаний счетчика VMware.

10.2.44 Веб-транзакция

Драйвер устройства веб-транзакции позволяет AtomMind Server наблюдать за веб-приложениями. Тестирование и мониторинг веб-приложений осуществляется с помощью выполнения тестовых скриптов. В оснеове этого драйвера лежит сторонняя платформа **Selenium**.

Selenium - это портативный фреймворк, который тестирует ПО для веб-приложений.

Больше информации о мониторинге веб-приложений вы можете найти здесь.

Информация о драйвере

ID плагина драйвера: com.tibbo.linkserver.plugin.device.webtransaction

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

НАСТРОЙКИ ПОДКЛЮЧЕНИЯ

Настройка	Описание
Скрипт	Этот тестовый скрипт будет выполняться при каждой синхронизации.

Активы Device

Драйвер не поддерживает активы.

Настройки Device

Скрипт состоит из нескольких действий. Когда скрипт выполняется, он вызывает действия одно за одним. Результаты выполнения действий предоставляются следующими настройками устройств:

- **Действия**. Таблица, содержащая каждый результат выполнения действия. Она включает **время** выполнения действия, **описание** действия и флаг **ошибки**.
- **Скриншоты**. Это особый тип действия, который может делать скриншоты, поэтому все скриншоты хранятся здесь.

Операции Device

- **Выполнить операцию**. Эта операция запускает тестовый скрипт, который определен в свойствах устройства. Операция возвращает таблицу с информацией о каждом шаге тестового скрипта.

События Device

Драйвер не предоставляет событий.

Подключение

Устройство веб-транзакции всегда находится онлайн.

Синхронизация

Тестовый скрипт выполняется при каждой синхронизации. Ошибка синхронизации может возникать в случае неполадок подключения. Информация о проблемах при выполнении скриптов включена в итоговую таблицу.

10.2.45 WebSphere MQ

Драйвер устройства 518 WebSphere MQ позволяет AtomMind Server проводить мониторинг работоспособности сервера IBM WebSphere MQ. Мониторинг проводится посредством собственного коммуникационного протокола MQ через IP сеть.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.mq

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

НАСТРОЙКИ СОЕДИНЕНИЯ

Настройки соединения определяют, как AtomMind Server взаимодействует с сервером MQ. Доступ к этим настройкам можно получить через действие Редактирование свойств 1494 Device контекста Device. Доступны следующие свойства соединения:

Свойство	Описание
IP адрес или имя хоста	Адрес сервера MQ.
Порт	Порт на сервере MQ (по умолчанию это 1414).
Канал	Имя канала MQ (по умолчанию это ADMIN.CONN).
Таймаут	Время ожидания операции (по умолчанию это 30 секунд).

Настройки Device

Драйвер устройства MQ предоставляет три табличных настройки только для чтения:

- **Каналы**. Эта таблица содержит информацию о каждом канале, представленном сервером MQ, включая имя канала, тип, статус, время запуска и объем входящего/исходящего трафика.
- **Очереди**. Эта таблица содержит информацию о каждой очереди, представленной сервером MQ, включая имя очереди, глубину, время последнего получения/вложения, максимальный и средний возраст сообщений, количество открытых входов/выходов и количество невыполненных сообщений.
- **Слушатели**. Эта таблица содержит информацию о каждом слушателе, представленном сервером MQ, включая имя слушателя, описание, статус, время запуска, очередь сообщений и счетчики команд/сессий.

Операции Device

Драйвер не предоставляет операций.

События Device

Драйвер не предоставляет событий.

Соединение

Драйвер помещает устройство в режим **Онлайн**, если соединение с сервером MQ было успешно установлено.

Синхронизация

Синхронизация между AtomMind Server и сервером MQ включает следующие шаги:

- Чтение списка каналов, доступных на сервере MQ, и получение информации о каждом отдельном канале.
- Чтение списка очередей, доступных на сервере MQ, и получение информации о каждой отдельной очереди.

- Чтение списка слушателей, доступных на сервере MQ, и получение информации о каждом отдельном слушателе.

10.2.46 WMI (инструментарий управления Windows)

Драйвер устройства 518 Инструментарий Управления Windows (WMI) позволяет AtomMind Servery контролировать компьютеры на Microsoft Windows. Как и с другими типами устройств, данные, собранные через WMI, конвертируются в унифицированную форму для обеспечения доступа к различным экземплярам AtomMind. Более подробную информацию о "нормализованном" представлении устройств в AtomMind смотрите в разделе Device.

Используя WMI драйвер AtomMind Server вы можете:

- читать и записывать свойства WMI объектов
- выбирать информацию при помощи языка запроса WMI (WQL)
- вызвать методы объектов WMI
- подписаться на события WMI.

Драйвер устройства WMI позволяет AtomMind Server контролировать и управлять компьютерами с поддержкой WMI локально или удаленно путем вызова или COM, или Distributed COM (DCOM). Технология, основанная на DCOM позволяет доступ к компьютеру с любой платформы, даже отличных от Windows, но требует должным образом настроенный компьютер клиента для удаленного доступа (см Настройка удаленного доступа к WMI 665). Технология COM позволяет "напрямую" использовать функции WMI без каких-либо дополнительных манипуляций на компьютере клиента, но эта технология работает только на основе Windows AtomMind Server.

Информация о драйвере

ID плагина 518 драйвера : com.tibbo.linkserver.plugin.device.wmi

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

СВОЙСТВА ПОДКЛЮЧЕНИЯ

Настройки подключения определяют, как AtomMind Server взаимодейсвует с определенным компьютером с поддержкой WMI. Данные настройки доступны через действие Изменить свойства 1494 Device контекста Device. Далее приведен список доступных свойств подключения:

Настройка	Описание
Тип соединения WMI	Технология Base: DCOM или прямые вызовы.
Локальный	Только для прямых соединений. Если проверяемый, устройство локально соединится с сервисами WMI; в этом случае нет необходимости в дополнительных данных соединения. Если непроверяемый, то драйвер соединяется с удаленными сервисами WMI, использующими данные, предоставленные ниже.
Адрес	IP адрес или имя хоста компьютера с поддержкой WMI. Спецификация `localhost` или `127.0.0.1` в качестве адреса устройства WMI не разрешена. Вместо этого используйте ваш адрес сети, например `192.168.1.2`.
Домен	Имя домена Windows (дополнительно).

<table>
<tr><td>Имя пользователя</td><td>Имя учетной записи пользователя Windows, используемое для доступа к WMI через DCOM.</td></tr>
<tr><td>Пароль</td><td>Пароль учетной записи пользователя Windows.</td></tr>
<tr><td>Уровень аутентификации</td><td>Только для прямых соединений.

Уровень аутентификации контролирует требования безопасности, которые клиент запрашивает с сервера. Уровень аутентификации от клиента и сервера сравнивается во время квитирования, самый высокий уровень настройки защиты безопасности используется для соединения.

Далее описаны два разных уровня аутентификации, с самого низкого уровня защиты безопасности до высокого:

<table>
<tr><th>Уровень аутентификации</th><th>Описание</th></tr>
<tr><td>Нет</td><td>В течение взаимодействия не предоставляется аутентификация между клиентом и сервером. Все настройки безопасности игнорируются.</td></tr>
<tr><td>Соединение</td><td>Нормальное квитирование аутентификации между клиентом и сервером, ключ сессии установлен, но этот ключ никогда не используется между клиентом и сервером. После квитирования все взаимодействия небезопасные.</td></tr>
<tr><td>Вызов</td><td>Only the headers of the beginning of each call are signed. The rest of the data exchanged between the client and server is neither signed nor encrypted. Most SSPs do not support this authentication level and silently promote it to Packet.</td></tr>
<tr><td>Пакет</td><td>Заголовок каждого пакета подпсиан, но не закодирован. Сами пакеты не подписаны или закодированы.</td></tr>
<tr><td>Целостность пакета</td><td>Каждый пакет данных полностью подписан, но не закодирован. Все данные подписаны отправителем, получатель должен быть уверен, что все данные были не повреждены во время передвижения.</td></tr>
<tr><td>Шифрование пакета</td><td>Каждый пакет данных подписан и закодирован. Это помогает полностью защитить взаимодействие между клиентом и сервером.

Этот уровень должен использоваться, если одно из пространств имен маркировано флажком Требуется кодирование (например, <code>root\MSCluster</code>).</td></tr>
</table>
</td></tr>
<tr><td>Все пространства имен</td><td>Определяет, должен ли драйвер изучать активы (классы WMI) во всех доступных пространствах имен WMI.</td></tr>
<tr><td>Область имен</td><td>Определяет список пространства имен WMI, который будет просканирован, чтобы проверить активы (классы WMI). Определяется:

• Путь пространства имен
• Рекурсивный флажок, который определяет, будут ли также сканироваться вложенные пространства имен.</td></tr>
<tr><td>Просмотр активов</td><td>Драйвер поддерживает два типа представления видов активов:

• "Плоский" вид, т.е простой список доступных WMI классов устройства.
• Древовидная структура, представляющая иерархию WMI классов.

Для применения данной опции к существующему устройству следует активизировать действие перезапуска драйвера устройства.</td></tr>
<tr><td>Таймаут соединения</td><td>Время ожидания для соединений на уровне TCP.</td></tr>
<tr><td>Таймаут выполнения</td><td>Время ожидания выполнения WMI операций.</td></tr>
</table>

WQL ЗАПРОСЫ

Таблица WQL запросов определяет запросы, которые могут быть использованы для получения мелкоструктурной информации от WMI устройства: определенные свойства объектов или объекты, которые отвечают установленным условиям.

Настройка	Описание
Имя	Имя запроса.
Описание	Описание запроса.
Выражения	Приводит к рассмотрению поля **WQL запроса** в виде выражения. 112 Данное выражение должно разрешаться в строку WQL запроса.
WQL запрос	Текст WQL запроса.
Таймаут	Таймаут на выполнение запроса.
Разрешить неполный результат	Контролирует, что происходит, когда полный результат запроса не получен до истечения **времени ожидания**. Если настройка **разрешить неполный результат** активирована, таблица результата запроса будет содержать частичные результаты, которые были получены в течение **времени ожидания**. Если настройка **разрешить неполный результат** отключена, выполнение запроса не получится, а переменная результата запроса бует переключена на состояние ошибки. Это значение будет равно предыдущему правильно полученному значение в таком случае.

WQL ЗАПРОСЫ СОБЫТИЙ

Таблица WQL запросов событий определяет, какие запросы используются для подписки на WMI события.

Настройка	Описание
Имя	Имя запроса.
Описание	Описание запроса.
Выражения	Приводит к рассмотрению поля **WQL запроса** в виде выражения. 112 Данное выражение должно разрешаться в строку WQL запроса.
Запрос WQL	Текст WQL запроса событий.
Таймаут получения следующего события	Указывает, сколько ждать до нового события в течение выполнения **WQL запроса**. Не рекомендуется слишком большой таймаут, потому что такой таймаут будет приостанавливать синхронизацию устройства 514. Если некоторые события не были получены до истечения времени ожидания, события все равно будут получены в течение следующего цикла синхронизации.

Активы Device

Для каждого WMI класса драйвер создает актив с таким же именем. Активируя актив, пользователь запрашивает драйвер выбрать информацию о всех экземплярах соответствующего класса. Большинство активов отключены по умолчанию, включены только те из них, которые наиболее часто используются.

Некоторые WMI классы могут иметь тысячи и даже миллионы экземпляров. Включение таких классов может повлечь за собой замедление синхронизации устройства и увеличение потребления ресурсов системы. Используйте четко сформулированный запрос для получения данных.

Настройки Device

Драйвер устройства WMI создает переменную настройки Device таким образом:

- Для каждого включенного актива (WMI класса) он создает переменную с таким же именем.
- Для каждого запроса, обозначенного в таблице WQL запросов, он создает переменную с таким же именем.
- Создается переменная **WMI Event**, которая хранит информацию о подписках на WMI события.

Операции Device

WMI драйвер создает функцию контекста Device и соответствующее действие для каждого метода включенных WMI классов. Данные действия группируются по именам классов.

При запуске WMI действие запрашивает путь к объекту и его параметры метода, затем вызывает метод для объекта, конвертирует полученные данные и отображает их в виде таблицы результатов.

Также есть возможность установить пользовательский таймаут до вызова любого WMI метода.

Драйвер также предоставляет операцию **Выполнить WQL запрос**, которая позволяет выполнить произвольный WQL запрос и вернуть его результаты.

События Device

WMI драйвер позволяет контролировать WMI события. Подписка на WMI события осуществляется посредством добавления **WQL запросов событий** в свойства устройства. Драйвер подписывается или отменяет подписку на уведомления о событиях при изменении данной таблицы.

Подключение

WMI драйвер приводит устройство в режим Онлайн, если:

- было установлено соединение с сервером DCOM определенного компьютера с использованием имени пользователя, домена и пароля.
- получен доступ к определенной области имен WMI, используя DCOM подключение.

Синхронизация

WMI устройства синхронизируются 514 с AtomMind Server подобно другим Device. Синхронизация включает в себя следующие шаги:

- Чтение определений активов (если они еще не были прочитаны или были заново установлены).

Каждый элемент актива относится к одноименному WMI классу.

- Получение метаданных устройства:
  - Для каждого включенного актива драйвер выбирает определение соответствующего WMI класса, включая его свойства и спецификации методов.
  - Для каждого запроса, определенного в таблице WQL запросов, драйвер создает переменную, которая будет содержать в себе все результаты.
  - Добавляет переменную **WMI Events**, которая контролирует подписку на все WMI события, указанные в таблице **WQL запросов событий.**
- Чтение/запись настроек устройства:
  - Драйвер читает свойства всех экземпляров классов, определяемых активами.
  - Записывает измененные свойства WMI объектов на контролируемое устройство.
  - Выполняет все WQL запросы, отправляя выбранные данные в переменные запроса.
  - Управляет подпиской на WMI события согласно переменной **WMI Event.**

Конвертация WMI данных

WMI объекты конвертируются в таблицы AtomMind следующим образом:

- Таблица, включающая данные одного или нескольких WMI объектов.
- Каждый объект представлен в виде единичной записи данных.
- Таблица содержит поле **Путь к объекту**, которое определяет объект по его пути в области имен WMI.
- Другие поля в таблице отражают свойства WMI объекта. Далее приведена таблица, показывающая, как типы WMI конвертируются в типы 49 AtomMind и наоборот:

WMI Тип	**Тип AtomMind**
Неподписанное 8-битное целое	целое
Подписанное 8-битное целое	целое

Неподписанное 16-битное целое	целое
Подписанное 16-битное целое	целое
Неподписанное 32-байтовое целое	целое
Подписанное 32-байтовое целое	целое
Неподписанное 64-байтовое целое	длинное
Подписаное 64-байтовое целое	длинное
UCS-2 строка	строка
Логическое	логическое
IEEE 4-байтовое плавающая точка	плавающее
IEEE 8-байтовое плавающая точка	двойное
Дата-время	дата
Ссылка	строка
16-байтовое UCS-2 символ	строка
Объект	таблица данных
Массивы данных	таблица данных

- Для улучшения конвертации используются квалификаторы класса и свойства (Key, Write, Abstract и т.д.).

10.2.46.1 Настройка удаленного доступа к WMI

Для настройки удаленного доступа к WMI следуйте приведенной далее инструкции.

Конфигурация DCOM

WMI использует DCOM для выполнения удаленных вызовов. Таким образом, DCOM должна быть доступна удаленно для определенного пользователя. Для более подробной информации об управлении удаленными вызовами обратитесь к разделу Конфигурация DCOM для удаленного доступа 2162.

Настройки доступа к области имен WMI

- Перейдите в **Панель управления > Администрирование > Управление компьютером > Службы и приложения.**
- Нажмите правой кнопкой на **Управляющий элемент WMI** и выберите **Свойства.**
- Перейдите на вкладку **Безопасность** и нажмите кнопку **Безопасность**.
- Добавьте контролирующую учетную запись пользователя и проверьте необходимые разрешения прав доступа (включая **Разрешить удаленный доступ**).

Убедитесь, что настройки безопасности унаследованы из Корня. В противном случае вы сможете настроить безопасность для определенного пространства имен.

Доступ к WMI через брандмауэр Windows

Если на WMI сервере включен брандмауэр Windows, вам необходимо разрешить в его настройках пропуск удаленных WMI запросов. Это можно сделать, используя командную строку:

```
netsh firewall set service RemoteAdmin enable
```

Имперсонация прав доступа к WMI

Перейдите в консоль "Редактор групповой политики", нажав **Пуск**, затем **Выполнить**, введите **gpedit.msc** и нажмите **OK.** В разделе **Политика локального компьютера** разверните раздел **Конфигурация компьютера**, затем раздел **Настройки Windows**. Разверните **Настройки безопасности**, перейдите в **Локальные политики,** затем выберите **Назначение прав пользователя**. Убедитесь, что в учетной записи **СЛУЖБА** появились **"Права олицетворять клиента после проверки подлинности"**.

Разрешения для реестра

Установите права полного доступа для пользователя, управляющего WMI на ветвь **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurePipeServers\WinReg.**

Если синхронизация устройства WMI с сервером Windows не выполняется с ошибкой **Нет доступа**, проделайте следующее на удаленном компьютере:

- Запустите Редактор Реестра ('**regedit.exe**').
- Найдите ключ реестра: 'HKEY_CLASSES_ROOT\CLSID\{76A64158-CB41-11D1-8B02-00600806D9B6}'.
- Кликните по нему правой кнопкой мыши и выберите '**Права доступа**'.
- Поменяйте владельца ключа на группу **'администраторы'**.
- Предоставьте права доступа **'Полный контроль'** группе '**администраторы**'.
- Поменяйте владельца ключа обратно на **TrustedInstaller** ("NT Service\TrustedInstaller")
- Повторите данную операцию для **всех** ключей реестра под именем {76A64158-CB41-11D1-8B02-00600806D9B6} (например, HKEY_LOCAL_MACHINE\Software\Classes\Wow6432Node\CLSID\{76A64158-CB41-11D1-8B02-00600806D9B6} и все другие). Эти ключи можно найти при поиске реестра.
- Перезапустите службу "**Удаленный реестр**".

Повторная инициализация поставщика WMI для получения доступа к недостающей информации

В некоторых редких случаях счетчики производительности WMI (это, как известно, происходит со счетчиками Microsoft Exchange 2003) могут оказаться недоступны через WMI, потому что поставщик WMI использует AutoDiscovery / AutoPurge (ADAP) для создания внутренней таблицы счетчика производительности. Если службы, предоставляющие счетчики, не запустились, когда процесс WMI ADAP запущен, счетчики производительности не переносятся в WMI.

Чтобы обойти проблему, вы можете запустить wmiadap.exe / F для принудительного перевода всех библиотек производительности. Чтобы сделать это, выполните следующие действия:

- Нажмите **Пуск**, **Выполнить**, напечатайте cmd, затем нажмите **OK**.
- При запросе команды напечатайте wmiadap.exe /f, а потом нажмите ENTER.
- Напечатайте exit и нажмите ENTER, чтобы закрыть запрос команды.

10.2.46.2 Массовый удаленный доступ к WMI

Внедряя AtomMind в больших сетях, часто необходимо включить WMI на сотнях и даже тысячах рабочих станций. Эта статья представляет собой пошаговую инструкцию для осуществления этого действия.

Нужно завершить две операции верхнего уровня:

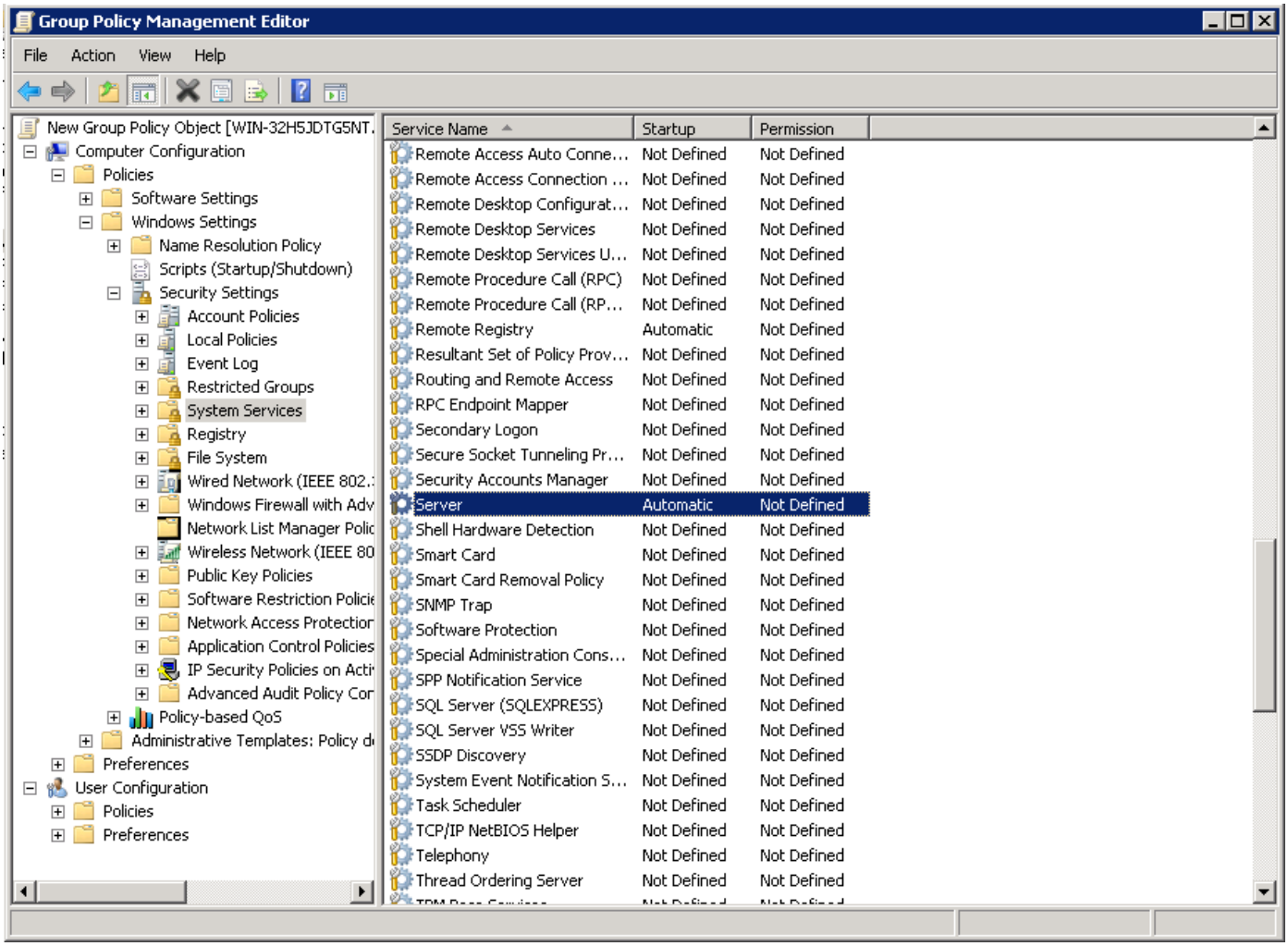
1. Настройка групповой политики
2. Запуск конфигурационного скрипта WMI

1. Настройка групповой политики

Групповая политика создается и настраивается на машине Контроллер Доменов.

2.1 ВКЛЮЧИТЕ СЕРВЕР И СЕРВИСЫ УДАЛЕННЫХ РЕЕСТРОВ

Используйте групповую политику для включения **Сервера** и сервиса **Удаленный Реестр**.

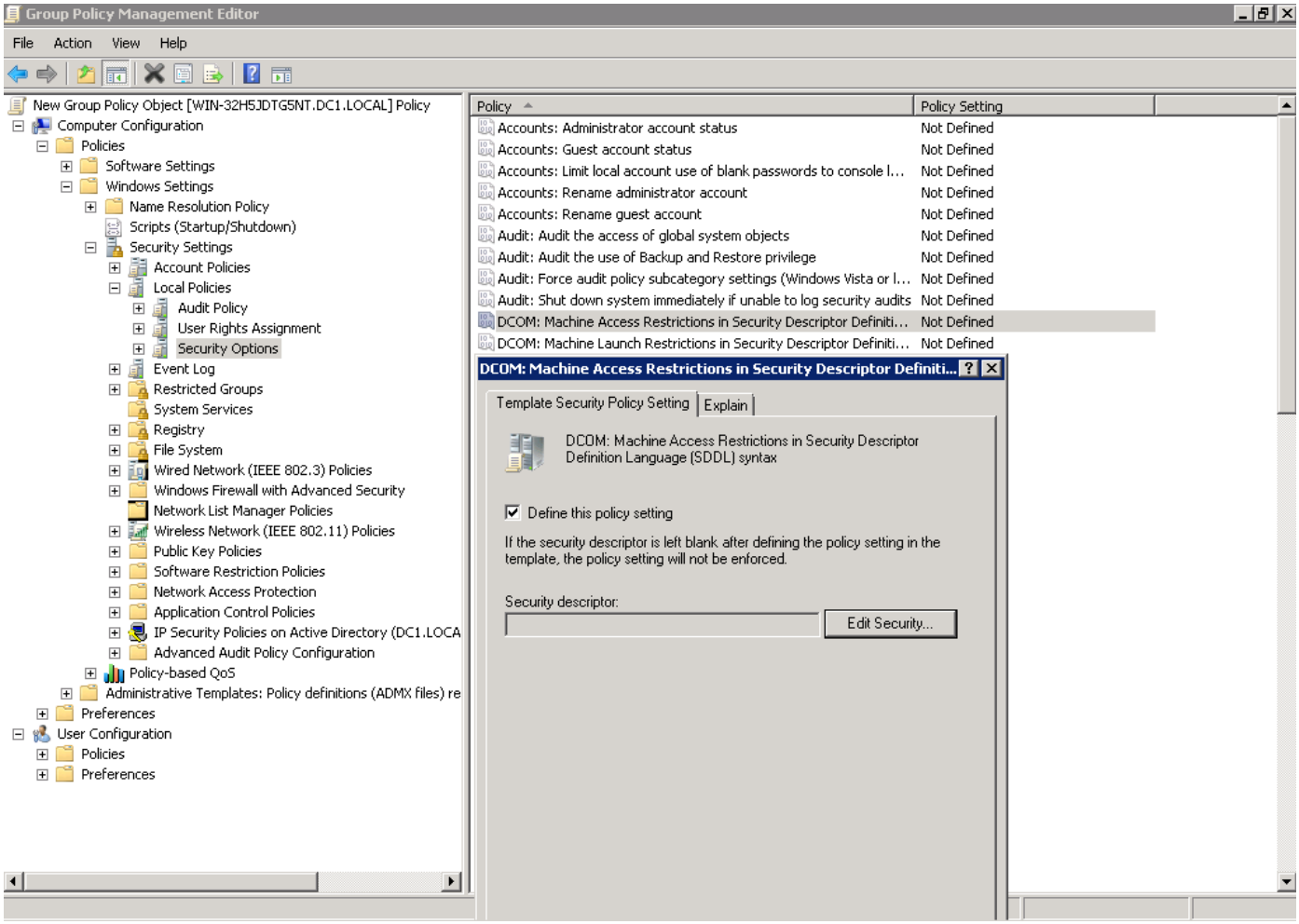


2.2 ВКЛЮЧИТЕ ДОСТУП DCOM

Используйте групповую политику для включения доступа DCOM.

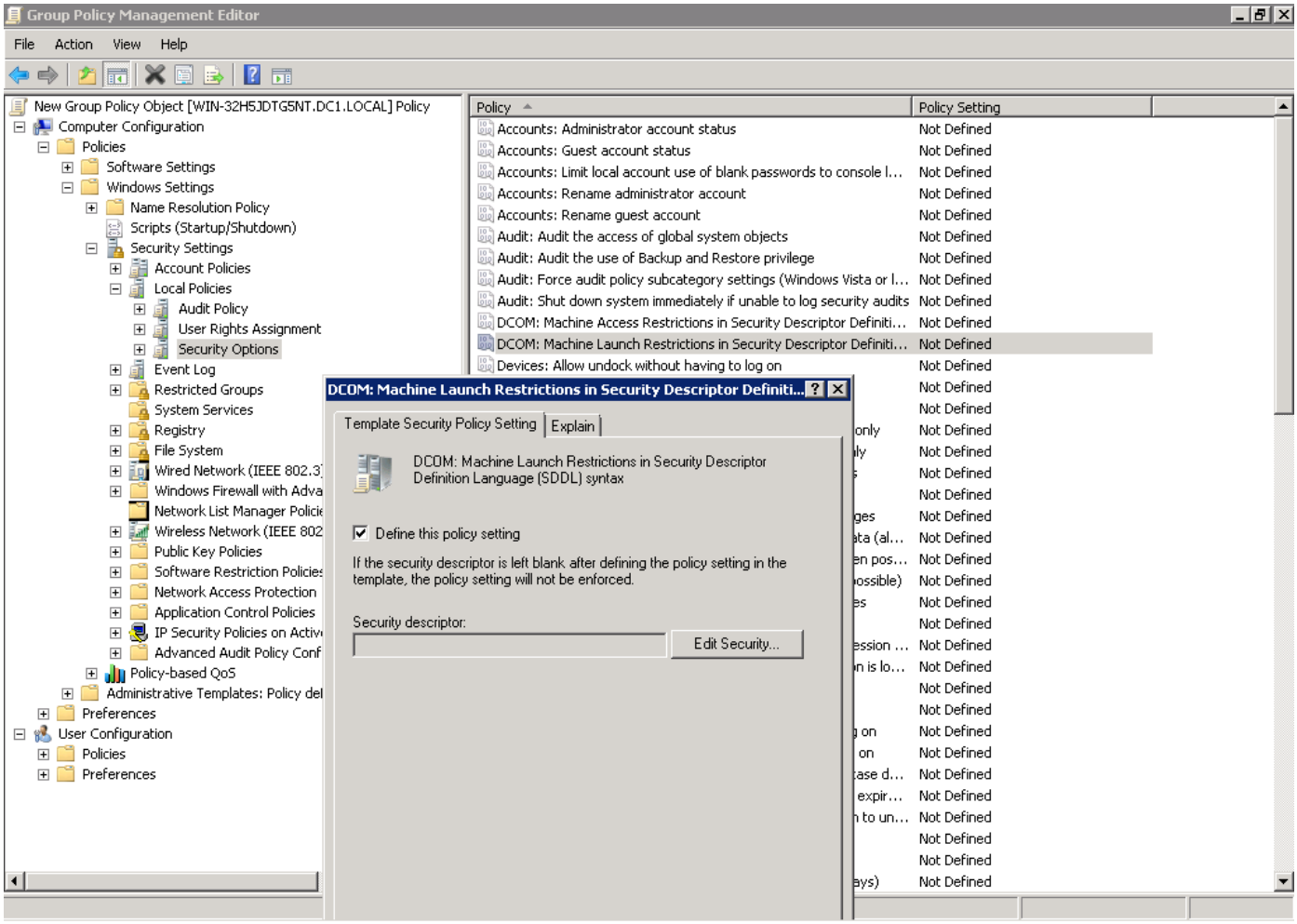
2.2.1. Дважды нажмите на политике **DCOM: Ограничения доступа машины**, затем используйте кнопку **Редактировать безопасность**, чтобы добавить пользователя, чьи параметры доступа будут использоваться для мониторинга.

Предоставьте этому пользователю права доступа **Удаленный доступ**.

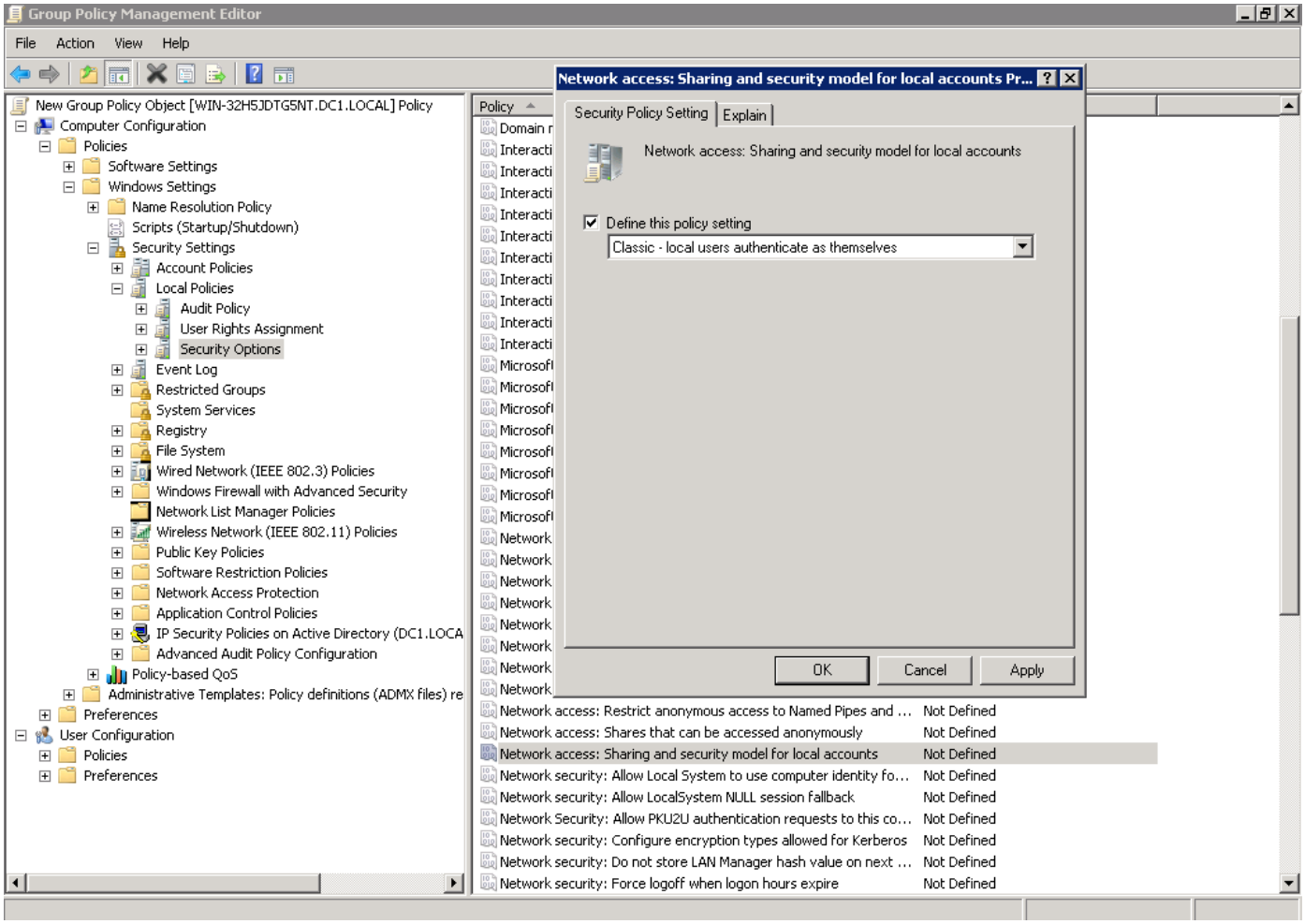


2.2.2. Нажмите дважды на политике **DCOM: Ограничения при запуске машины**, затем используйте кнопку **Редактировать безопасность**, чтобы добавить пользователя, чьи параметры доступа будут использоваться для мониторинга.

Предоставьте этму пользователю права доступа **Локальный запуск**, **Удаленный запуск**, **Локальная активация** и **Удаленная активация**.



2.2.3. Дважды нажмите на политике **Доступ к сети: совместное использование и модель обеспечения защиты для локальных учетных записей**, в раскрывающемся списке выберите **Классическая - пользователи аутентифицируются как они сами**.



2.3 НАСТРОЙТЕ ИЛИ ВЫКЛЮЧИТЕ БРАНДМАУЭР WINDOWS

Перенастройте **Брандмауэр Windows** с использованием групповой политики, если необходимо.

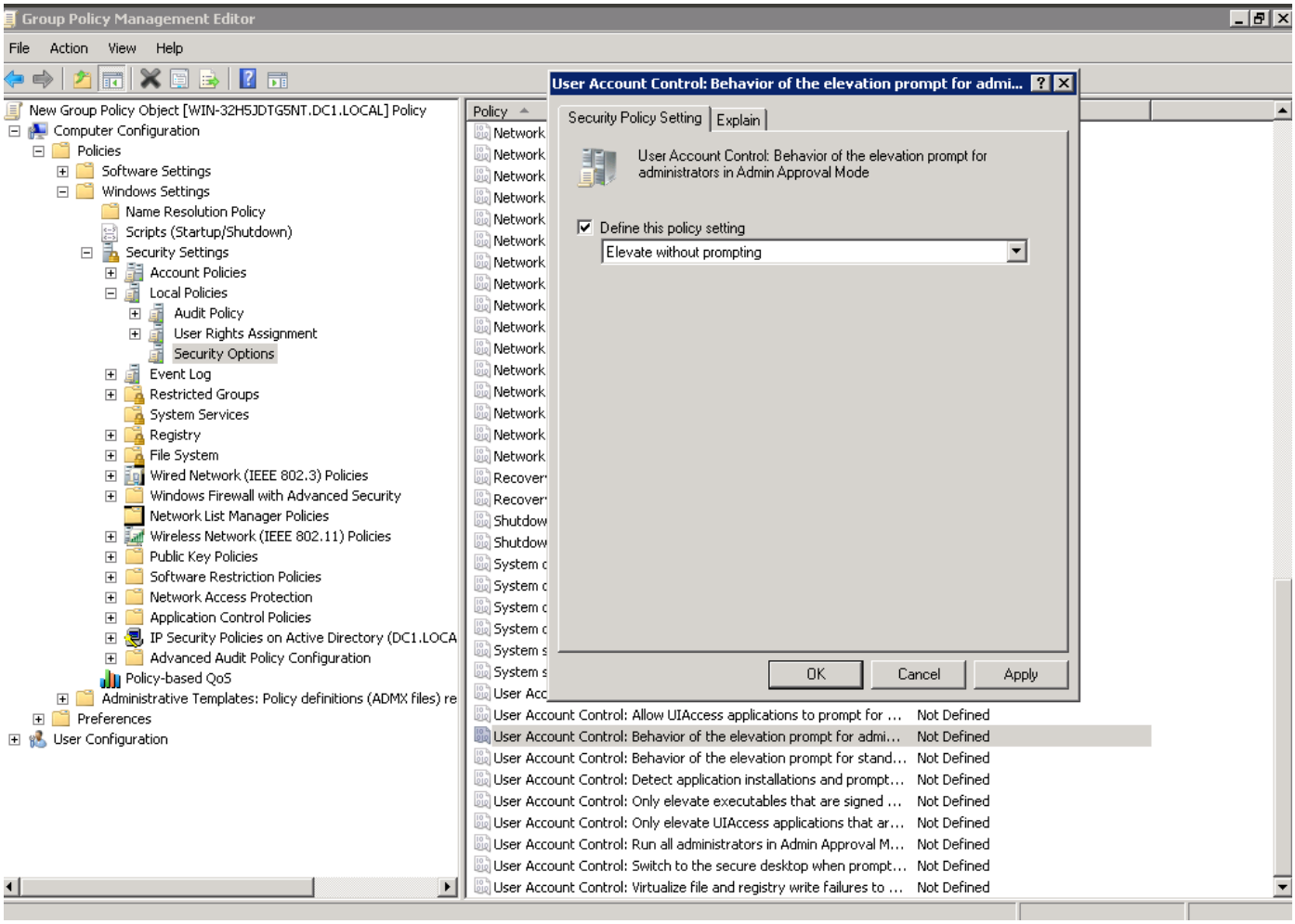
В некоторых случаях **Брандмауэр Windows** настраивается для блокировки протокола **DCOM**. Убедитесь, что **Брандмауэр Windows** не блокирует протокол **DCOM** или выключен.

2.4 ВЫКЛЮЧИТЕ КОНТРОЛЬ УЧЕТНЫХ ЗАПИСЕЙ ПОЛЬЗОВАТЕЛЕЙ (UAC)

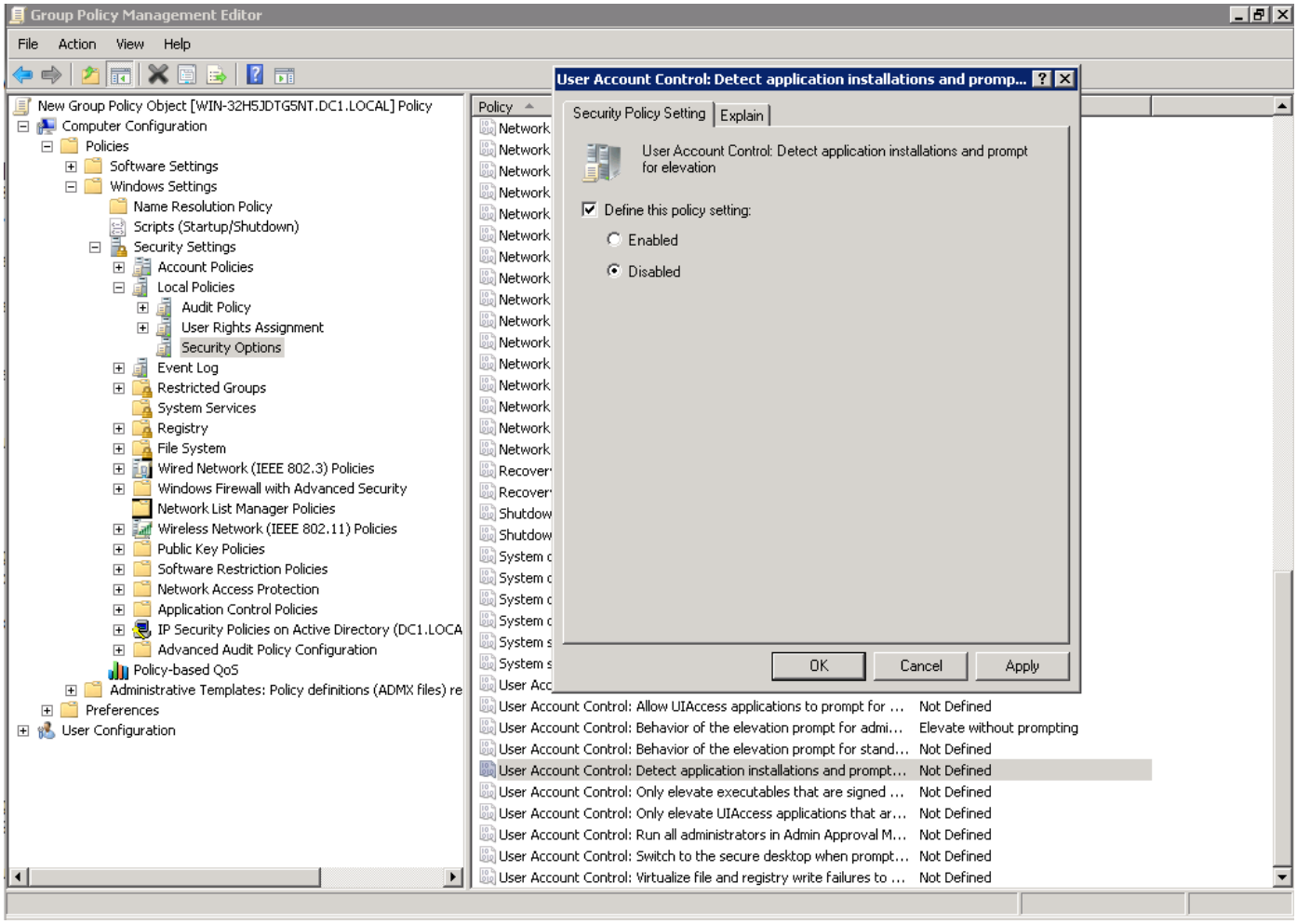
Когда Контроль учетных записей пользователей (**UAC**) активен, у учетной записи администратора по факту есть два маркера безопасности, стандартный маркер пользователя и также маркер администратора (который активируется только тогда, когда вы проходите контроль учетных записей). К сожалению, удаленные запросы, идущие через сеть, получают стандарный маркер пользователя для администратора, и так как нет никакого способа для обработки учетных записей строки удаленно, маркер не может быть возведен в истинно администраторский маркер безопасности.

Таким образом, **UAC** должен быть отключен с помощью групповой политики для обеспечения удаленного доступа DCOM.

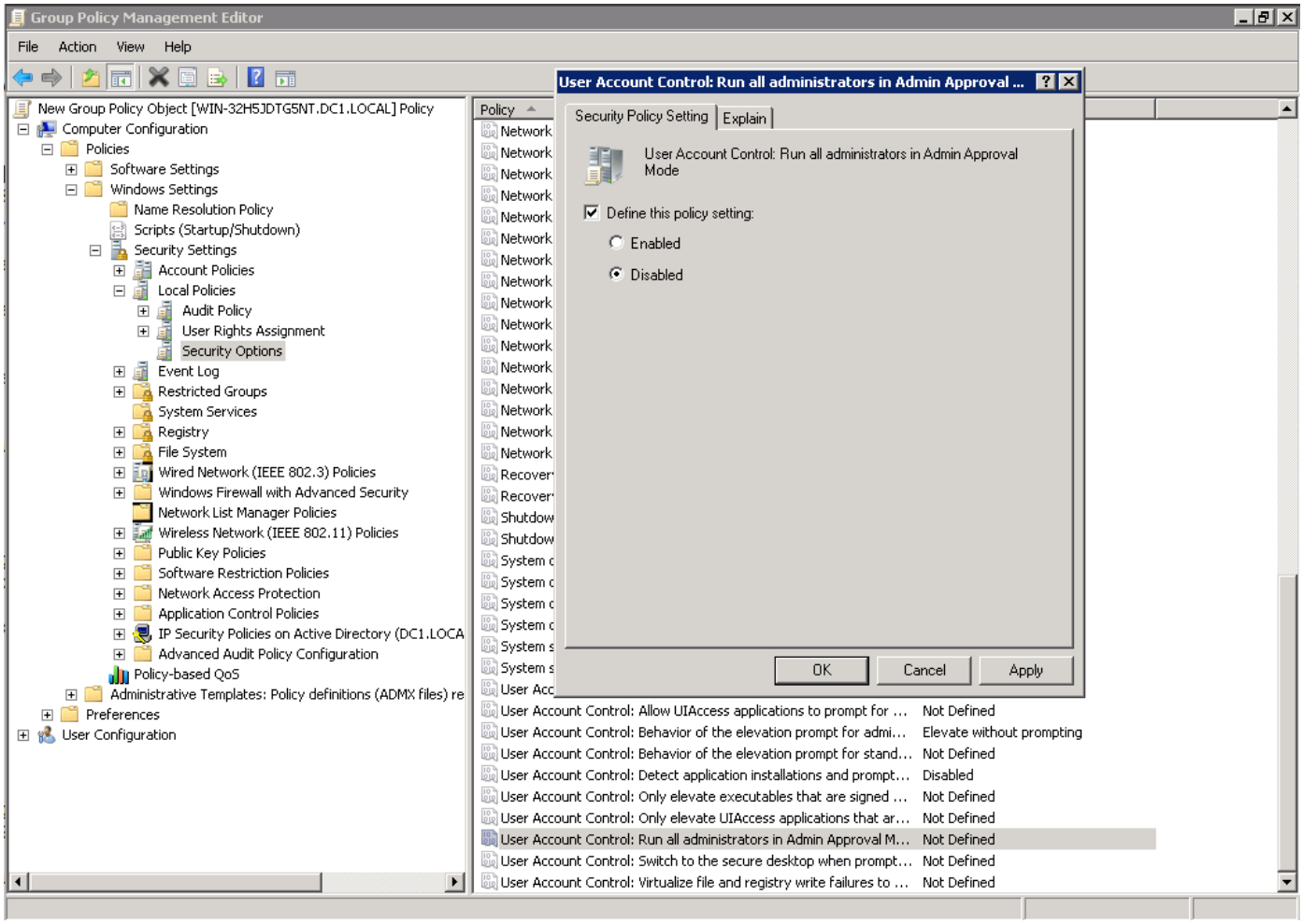
2.4.1. Установите политику **Контроль учетных записей пользователей: поведение запроса на повышение прав для администраторов в режиме одобрения администратором** на **Повышение без запроса**



2.4.2. Установите политику на **Контроль учетных записей: обнаружение установки приложений и запрос на повышение** на **Выключено**

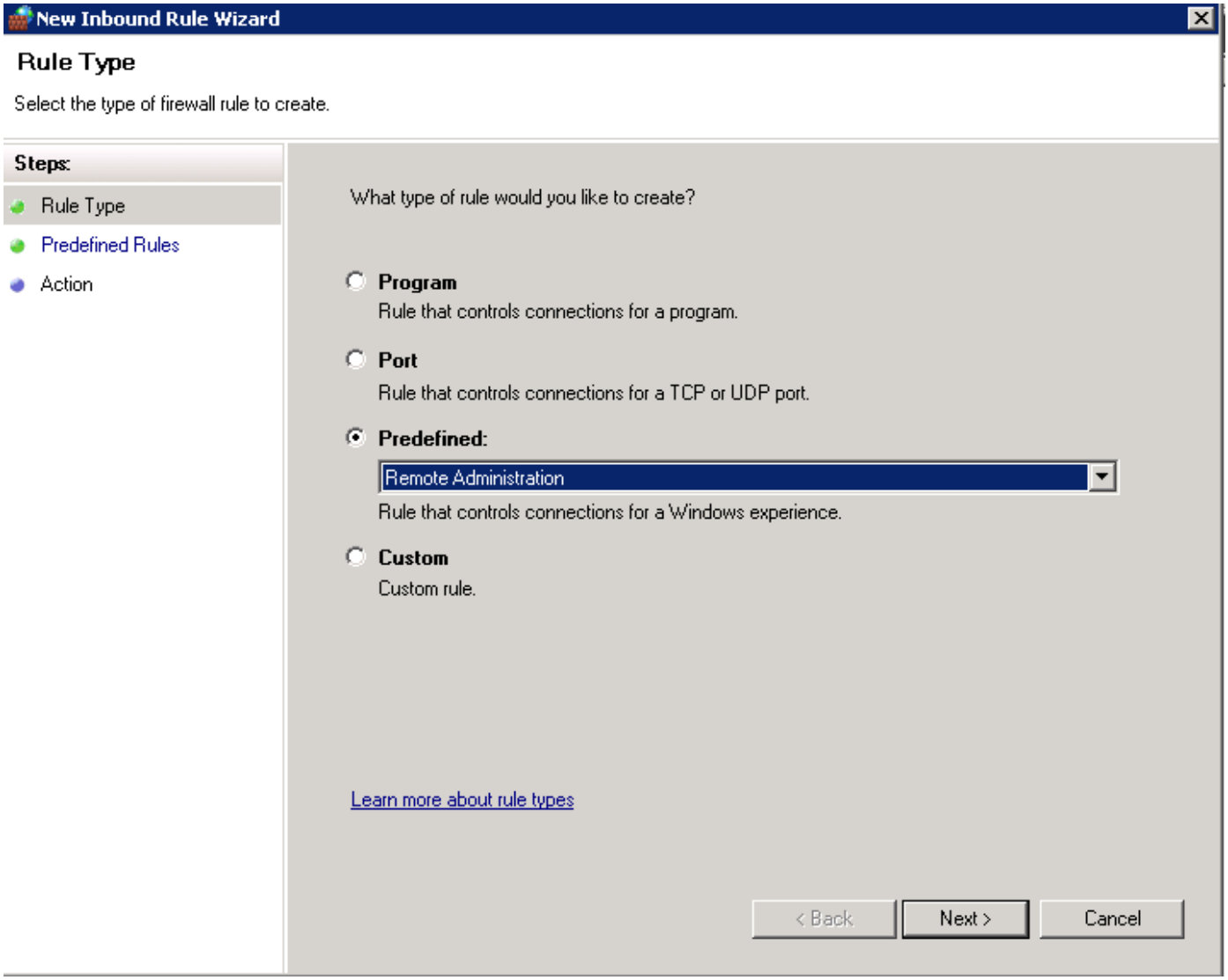


2.4.3. Установите политику **Управление учетными записями пользователей: все администраторы работают в режиме одобрения администратором** на **Выключено**



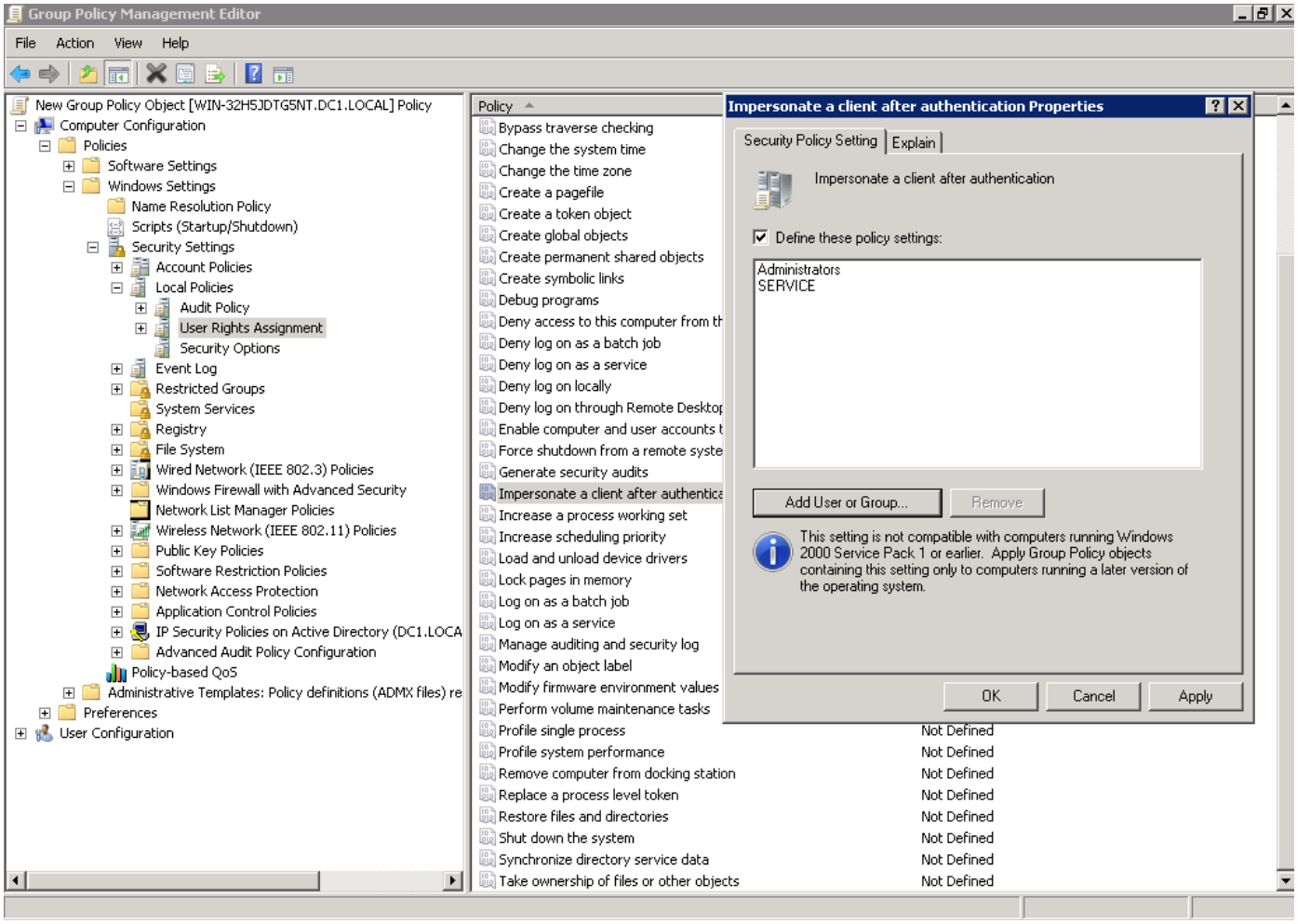
2.5 ВКЛЮЧИТЕ ДОСТУП WMI ЧЕРЕЗ БРАНДМАУЭР WINDOWS

Перенастройте **Брандмауэр Windows**, используя групповую политику для передачи запросов WMI:



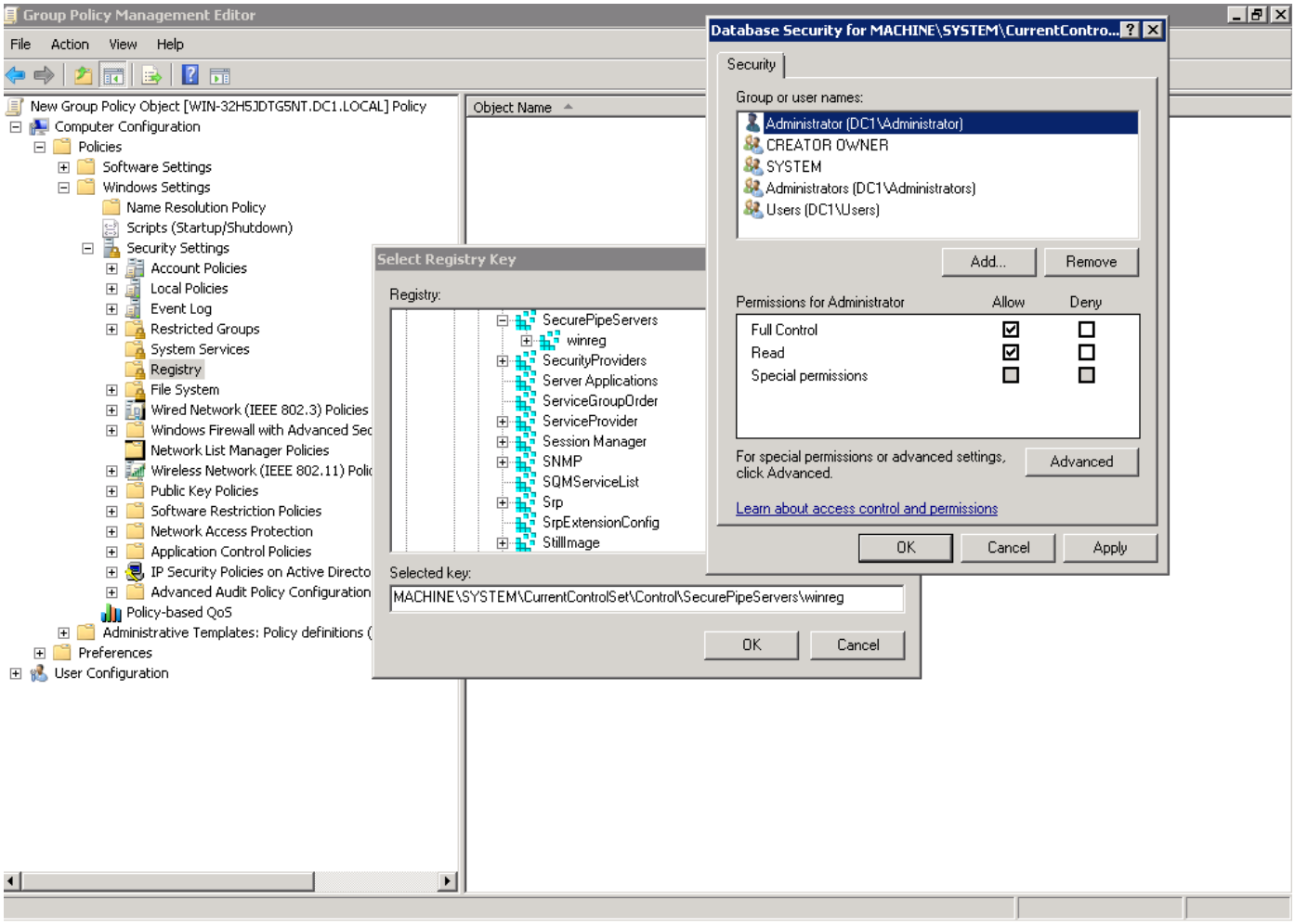
2.6 НАСТРОЙТЕ ИМПЕРСОНАЦИЮ WMI

Перейдите в **Настройки беопасности >Локальные политики**, затем выберите **Назначение прав пользователя**. Убедитесь, что политика **Имперсонация клиента после аутентификации** предоставляет права доступа учетной записи системы **СЛУЖБА**.



2.7 НАСТРОЙКА ПРАВ ДОСТУПА РЕЕСТРА

Предоставьте пользователю права доступа, которые будут использоваться для мониторинга прав доступа **Полный контроль** к ветви реестра **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurePipeServers\WinReg**.



2. Запуск конфигурационного скрипта WMI

Конфигурационный скрипт WMI делает работу по настройке массового конфигурирования WMI. Как только она заканчивается, вы можете подключить хосты WMI к мониторингу AtomMind.

Скрипт написан в PowerShell. Текст скрипта доступен в отдельной статье 677.

Инструкции для запуска:

1. Сохраните скрипт как файл `wmi_config.ps1`.
2. Скрипт можно запустить на машине **Windows 2008 Server** или **Windows 7** под учетной записью администратора домена.
3. Учетная запись пользователя, запустившего скрипт, должна быть включена в группу **Локальные администраторы** на каждой настраиваемой рабочей станции. Это можно сделать с помощью групповой политики: перейдите в **Конфигурация компьютера> Настройка Windows> Параметры безопасности> Группы с ограниченным доступом**, добавьте группу **Администраторы**, а затем добавьте домен, чьи привилегии должны быть переданы этой группе. Получите предупреждение о том, что пользователи домена, которые не были добавлены в новую созданную группу, будут удалены из локальных администраторов.
4. Необходимо выполнить скрипты PowerShell, используя команду `Set-ExecutionPolicy RemoteSigned`.
5. Нужно синхронизировать время всех рабочих станций, вовлеченных в мониторинг WMI.
6. Укажите параметр **$ComputerListFile** в скрипте к файлу со списком настраиваемых IP адресов хостов.
7. В параметре скрипта **$account** укажите имя учетной записи пользователя, чьи параметры доступа будут использоваться для мониторинга.
8. Запустите скрипт.

Нахождение хостов, готовых к настройке WMI

Чтобы создать файл со списком всех хостов в вашей сети, готовых для настройки WMI (например, с открытым портом RPC (135)), используйте другой скрипт PowerShell 680.

Автоматическое включение WMI

Скрипт, сканирующий хосты WMI, и конфигурационный скрипт WMI могут запускаться периодически с использованием планировщика задач Windows. Это позволит автоматически находить и автоматическое подключать все новые хосты WMI к AtomMind с помощью функции обнаружения сети AtomMind.

10.2.46.2.1 Конфигурационный скрипт WMI

Этот скрипт используется для массового конфигурирования удаленного доступа к WMI.

```
# Usage:
# 1. Edit computers.txt referred by $ComputerListFile parameter
# 2. Set $account parameter to the name of user those credentials will be used for monitoring
# 3. Run the script
# 4. Check the log file for results

$account = "administrator@dc1"
$ComputerListFile = "C:\wmi\computers.txt"
$LogFile = "C:\wmi\wmi_setup.log"

#--------------------------------------------- Function for print messages to $LogFile and screen
function print_message($File, [string]$Text)
{
  Write-Host $Text
  Add-Content $File $Text
}

#--------------------------------------------- Function for check open DCOM port (135) -----------
function check_open_port($ip, $port, $con_timeout)
{
    $tcpclient = new-object Net.Sockets.TcpClient
    $Connection = $tcpclient.BeginConnect($ip, $port, $null, $null)
    $TimeOut = $Connection.AsyncWaitHandle.WaitOne($con_timeout,$false)
    if(!$TimeOut)   {
        $TCPclient.Close()
        return 0
    } else {
        try {
            $TCPclient.EndConnect($Connection) | out-Null
            $TCPclient.Close()
            return 1
        } catch {
            ## Machine actively refused the connection. The port is not open but $TimeOut was stil
            return 0
        }
    }
}

#--------------------------------------------- Function for enable privilege to change CLSID\{76A6
function enable-privilege {
  param(
  ## The privilege to adjust. This set is taken from
  ## http://msdn.microsoft.com/en-us/library/bb530716(VS.85).aspx
  [ValidateSet(
   "SeAssignPrimaryTokenPrivilege", "SeAuditPrivilege", "SeBackupPrivilege",
   "SeChangeNotifyPrivilege", "SeCreateGlobalPrivilege", "SeCreatePagefilePrivilege",
   "SeCreatePermanentPrivilege", "SeCreateSymbolicLinkPrivilege", "SeCreateTokenPrivilege",
   "SeDebugPrivilege", "SeEnableDelegationPrivilege", "SeImpersonatePrivilege", "SeIncreaseBasePri
   "SeIncreaseQuotaPrivilege", "SeIncreaseWorkingSetPrivilege", "SeLoadDriverPrivilege",
   "SeLockMemoryPrivilege", "SeMachineAccountPrivilege", "SeManageVolumePrivilege",
   "SeProfileSingleProcessPrivilege", "SeRelabelPrivilege", "SeRemoteShutdownPrivilege",
   "SeRestorePrivilege", "SeSecurityPrivilege", "SeShutdownPrivilege", "SeSyncAgentPrivilege",
   "SeSystemEnvironmentPrivilege", "SeSystemProfilePrivilege", "SeSystemtimePrivilege",
   "SeTakeOwnershipPrivilege", "SeTcbPrivilege", "SeTimeZonePrivilege", "SeTrustedCredManAccessPri
   "SeUndockPrivilege", "SeUnsolicitedInputPrivilege")]
  $Privilege,
  ## The process on which to adjust the privilege. Defaults to the current process.
  $ProcessId = $pid,
  ## Switch to disable the privilege, rather than enable it.
  [Switch] $Disable
```

```
  )

  ## Taken from P/Invoke.NET with minor adjustments.
  $definition = @'
  using System;
  using System.Runtime.InteropServices;

  public class AdjPriv
  {
      [DllImport("advapi32.dll", ExactSpelling = true, SetLastError = true)]
      internal static extern bool AdjustTokenPrivileges(IntPtr htok, bool disall,
      ref TokPriv1Luid newst, int len, IntPtr prev, IntPtr relen);

      [DllImport("advapi32.dll", ExactSpelling = true, SetLastError = true)]
      internal static extern bool OpenProcessToken(IntPtr h, int acc, ref IntPtr phtok);
      [DllImport("advapi32.dll", SetLastError = true)]
      internal static extern bool LookupPrivilegeValue(string host, string name, ref long pluid);
      [StructLayout(LayoutKind.Sequential, Pack = 1)]
      internal struct TokPriv1Luid
      {
       public int Count;
       public long Luid;
       public int Attr;
      }

      internal const int SE_PRIVILEGE_ENABLED = 0x00000002;
      internal const int SE_PRIVILEGE_DISABLED = 0x00000000;
      internal const int TOKEN_QUERY = 0x00000008;
      internal const int TOKEN_ADJUST_PRIVILEGES = 0x00000020;
      public static bool EnablePrivilege(long processHandle, string privilege, bool disable)
      {
       bool retVal;
       TokPriv1Luid tp;
       IntPtr hproc = new IntPtr(processHandle);
       IntPtr htok = IntPtr.Zero;
       retVal = OpenProcessToken(hproc, TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY, ref htok);
       tp.Count = 1;
       tp.Luid = 0;
       if(disable)
       {
          tp.Attr = SE_PRIVILEGE_DISABLED;
       }
       else
       {
          tp.Attr = SE_PRIVILEGE_ENABLED;
       }
       retVal = LookupPrivilegeValue(null, privilege, ref tp.Luid);
       retVal = AdjustTokenPrivileges(htok, false, ref tp, 0, IntPtr.Zero, IntPtr.Zero);
       return retVal;
      }
   }
'@

  $processHandle = (Get-Process -id $ProcessId).Handle
  $type = Add-Type $definition -PassThru
  $type[0]::EnablePrivilege($processHandle, $Privilege, $Disable)
}

#------------------------------------------- First stage (get account sid, open file computers.t
function get-sid
{
    Param (
    $DSIdentity
    )
    $ID = new-object System.Security.Principal.NTAccount($DSIdentity)
    return $ID.Translate( [System.Security.Principal.SecurityIdentifier] ).toString()
}
$sid = get-sid $account
$SDDL = "A;;CCWP;;;$sid"
```

```
$DCOMSDDLDefaultLaunchPermission = "A;;CCDCLCSWRP;;;$sid"
$DCOMSDDLDefaultAccessPermission = "A;;CCDCLC;;;$sid"
$DefaultSDDLAccess = "O:BAG:BAD:(A;;CCDCLC;;;PS)(A;;CCDC;;;SY)(A;;CCDCLC;;;BA)"
$DefaultSDDLLaunch = "O:BAG:BAD:(A;;CCDCLCSWRP;;;BA)(A;;CCDCSW;;;IU)(A;;CCDCSW;;;SY)(A;;CCDCLCSWRP
$computers = Get-Content $ComputerListFile

$timeNow = get-date
print_message $LogFile $timeNow

#---------------------------------------------- Main loop ----------------------------------------
foreach ($strcomputer in $computers)
{
       print_message $LogFile $strcomputer

        $Port135Open = check_open_port $strcomputer "135" "1000"
        if ($Port135Open) {
            try{
                 $Reg = [WMIClass]"\\$strcomputer\root\default:StdRegProv"
            }
            catch {
               $message = "Error: " + [string]$Error[0]
                 print_message $LogFile $message
                 continue
            }
        } else {
            print_message $LogFile "Error! The RPC server is unavailable. No ping."
            continue
        }

        #-------------------------- apply group policy ------------------------------------------
       try{
           $Win32_OS = Get-WmiObject Win32_OperatingSystem -computer $strcomputer
        }
        catch {
           $message = "Error: " + [string]$Error[0]
            print_message $LogFile $message
            continue
        }
        print_message $LogFile $Win32_OS.caption
        $str_version = [regex]::Replace($Win32_OS.version, "(\d.\d).*", '$1'); #6.1.7601 -> 6.1
        $version = [decimal]$str_version
       print_message $LogFile "Applying group policies..."

        If ($version -ge 5.1) {
            Invoke-WmiMethod -ComputerName $strcomputer -Path win32_process -Name create -Argument
        } Else {
            Invoke-WmiMethod -ComputerName $strcomputer -Path win32_process -Name create вЂ“Argume
        }
       print_message $LogFile "Applying group policies completed"

       #---------------------------------------- If Windows 2008 or 7 set CLSID\{76A64158-CB41-11D1
       If ($version -ge 6.1) {
           print_message $LogFile "Set CLSID\{76A64158-CB41-11D1-8B02-00600806D9B6} permissions...
           $RemoteKey = [Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey([Microsoft.Win32.Registry

           enable-privilege SeTakeOwnershipPrivilege
           $key = $RemoteKey.OpenSubKey("CLSID\{76A64158-CB41-11D1-8B02-00600806D9B6}",[Microsoft.
           # You must get a blank acl for the key b/c you do not currently have access
           $acl = $key.GetAccessControl([System.Security.AccessControl.AccessControlSections]::Non
           $me = [System.Security.Principal.NTAccount]"BUILTIN\Administrators"
           $acl.SetOwner($me)
           $key.SetAccessControl($acl)
           $TrustedInstaller = [System.Security.Principal.NTAccount]"NT Service\TrustedInstaller"
           $acl.SetOwner($TrustedInstaller)

           # After you have set owner you need to get the acl with the perms so you can modify it.
           $acl = $key.GetAccessControl()
           $rule = New-Object System.Security.AccessControl.RegistryAccessRule ("BUILTIN\Administr
           $acl.SetAccessRule($rule)
```

```
            $key.SetAccessControl($acl)

            $key.Close()
            #http://social.technet.microsoft.com/Forums/en/winserverpowershell/thread/e718a560-2908
        }

         #-------------------------------------- Enable DCOM, Set authentication (to "Connect") an
         $Result = $Reg.SetStringValue(2147483650,"software\microsoft\ole","EnableDCOM","Y")
         if ($Result) {print_message $LogFile "EnableDCOM -> Y"}
         $Result = $Reg.SetDWORDValue(2147483650,"software\microsoft\ole","LegacyAuthenticationLeve
         if ($Result) {print_message $LogFile "LegacyAuthenticationLevel -> Connect"}
         $Result = $Reg.SetDWORDValue(2147483650,"software\microsoft\ole","LegacyImpersonationLevel
         if ($Result) {print_message $LogFile "LegacyImpersonationLevel -> Identify"}

        #-------------------------------------- Set wmi namespace root/CIMV2 permissions, set COM s
        $converter = new-object system.management.ManagementClass Win32_SecurityDescriptorHelper

         $security = Get-WmiObject -ComputerName $strcomputer -Namespace root/cimv2 -Class __System
         $binarySD = @($null)
         $result = $security.PsBase.InvokeMethod("GetSD",$binarySD)
         $outsddl = $converter.BinarySDToSDDL($binarySD[0])
        $newSDDL = $outsddl.SDDL += "(" + $SDDL + ")"
         $WMIbinarySD = $converter.SDDLToBinarySD($newSDDL)
         $WMIconvertedPermissions = ,$WMIbinarySD.BinarySD
         $result = $security.PsBase.InvokeMethod("SetSD",$WMIconvertedPermissions)
         print_message $LogFile "ROOT\cimv2 permissions should be granted"

        $COMAccess = $Reg.GetBinaryValue(2147483650,"software\microsoft\ole","DefaultAccessPermissi
        if ($COMAccess) {
                $outCOMAccessSDDL = $converter.BinarySDToSDDL($COMAccess)
               $newCOMAccessSDDL = $outCOMAccessSDDL.SDDL += "(" + $DCOMSDDLDefaultAccessPermission
        } else {
               $newCOMAccessSDDL = $DefaultSDDLAccess + "(" + $DCOMSDDLDefaultAccessPermission + ")
        }
        $COMAccessbinarySD = $converter.SDDLToBinarySD($newCOMAccessSDDL)
        $COMAccessconvertedPermissions = ,$COMAccessbinarySD.BinarySD
        $result = $Reg.SetBinaryValue(2147483650,"software\microsoft\ole","DefaultAccessPermission"
        if ($result.Returnvalue -eq 0) {print_message $LogFile "DefaultAccessPermission granted"}

        $COMLaunch = $Reg.GetBinaryValue(2147483650,"software\microsoft\ole","DefaultLaunchPermissi
        if ($COMLaunch) {
               $outCOMLaunchSDDL = $converter.BinarySDToSDDL($COMLaunch)
                $newCOMLaunchSDDL = $outCOMLaunchSDDL.SDDL += "(" + $DCOMSDDLDefaultLaunchPermissio
        } else {
                $newCOMLaunchSDDL = $DefaultSDDLLaunch + "(" + $DCOMSDDLDefaultLaunchPermission + "
        }
        $COMLaunchbinarySD = $converter.SDDLToBinarySD($newCOMLaunchSDDL)
        $COMLaunchconvertedPermissions = ,$COMLaunchbinarySD.BinarySD
        $result = $Reg.SetBinaryValue(2147483650,"software\microsoft\ole","DefaultLaunchPermission"
        if ($result.Returnvalue -eq 0) {print_message $LogFile "DefaultLaunchPermission granted"}
}
```

10.2.46.2.2 Сетевое сканирование WMI

Этот скрипт используется для сканирования подсети, поиска всех хостов, для которых открыт RPC порт (135), и записи их списка в файл.

```
# Usage:
# 1. Edit script, change $ip, $mask, $outpitFile, and $rewriteFile parameters
# 2. Start script
# 3. Check the output file

$ip = "192.168.1.1"
$mask = "255.255.255.0"
$outputFile = "C:\wmi\computers.txt"
$rewriteFile = 1
```

```
#-------------------------------------------- Function for print messages to $LogFile and screen
function print_message($File, [string]$Text)
{
  Write-Host $Text
  Add-Content $File $Text
}

#-----------------------------------------------------------------------------------------------
Function ConvertTo-DecimalIP {
  <#
    .Synopsis
      Converts a Decimal IP address into a 32-bit unsigned integer.
    .Description
      ConvertTo-DecimalIP takes a decimal IP, uses a shift-like operation on each octet and return
    .Parameter IPAddress
      An IP Address to convert.
  #>
  [CmdLetBinding()]
  Param(
    [Parameter(Mandatory = $True, Position = 0, ValueFromPipeline = $True)]
    [Net.IPAddress]$IPAddress
  )
  Process {
    $i = 3; $DecimalIP = 0;
    $IPAddress.GetAddressBytes() | ForEach-Object { $DecimalIP += $_ * [Math]::Pow(256, $i); $i--

    Return [UInt32]$DecimalIP
  }
}

Function ConvertTo-DottedDecimalIP {
  <#
    .Synopsis
      Returns a dotted decimal IP address from either an unsigned 32-bit integer or a dotted binar
    .Description
      ConvertTo-DottedDecimalIP uses a regular expression match on the input string to convert to
    .Parameter IPAddress
      A string representation of an IP address from either UInt32 or dotted binary.
  #>
  [CmdLetBinding()]
  Param(
    [Parameter(Mandatory = $True, Position = 0, ValueFromPipeline = $True)]
    [String]$IPAddress
  )
  Process {
    Switch -RegEx ($IPAddress) {
      "([01]{8}\.){3}[01]{8}" {
        Return [String]::Join('.', $( $IPAddress.Split('.') | ForEach-Object { [Convert]::ToUInt32
      }
      "\d" {
        $IPAddress = [UInt32]$IPAddress
        $DottedIP = $( For ($i = 3; $i -gt -1; $i--) {
          $Remainder = $IPAddress % [Math]::Pow(256, $i)
          ($IPAddress - $Remainder) / [Math]::Pow(256, $i)
          $IPAddress = $Remainder
         } )
        Return [String]::Join('.', $DottedIP)
      }
      default {
        Write-Error "Cannot convert this format"
      }
    }
  }
}

Function Get-NetworkAddress {
  <#
    .Synopsis
```

```
      Takes an IP address and subnet mask then calculates the network address for the range.
    .Description
      Get-NetworkAddress returns the network address for a subnet by performing a bitwise AND
      operation against the decimal forms of the IP address and subnet mask. Get-NetworkAddress
      expects both the IP address and subnet mask in dotted decimal format.
    .Parameter IPAddress
      Any IP address within the network range.
    .Parameter SubnetMask
      The subnet mask for the network.
  #>
  [CmdLetBinding()]
  Param(
    [Parameter(Mandatory = $True, Position = 0, ValueFromPipeline = $True)]
    [Net.IPAddress]$IPAddress,

    [Parameter(Mandatory = $True, Position = 1)]
    [Alias("Mask")]
    [Net.IPAddress]$SubnetMask
  )
  Process {
    Return ConvertTo-DottedDecimalIP ((ConvertTo-DecimalIP $IPAddress) -BAnd (ConvertTo-DecimalIP
  }
}

function check_open_port($ip, $port, $con_timeout)
{
    $tcpclient = new-object Net.Sockets.TcpClient
    $Connection = $tcpclient.BeginConnect($ip, $port, $null, $null)
    $TimeOut = $Connection.AsyncWaitHandle.WaitOne($con_timeout,$false)
    if(!$TimeOut)   {
        $TCPclient.Close()
        return 0
    } else {
        try {
            $TCPclient.EndConnect($Connection) | out-Null
            $TCPclient.Close()
            return 1
        } catch {
            ## Machine actively refused the connection. The port is not open but $TimeOut was stil
            return 0
        }
    }
}


if ($rewriteFile) {
    Remove-Item $outputFile
}
$mm = "255.255.255.255"
$dmm = ConvertTo-DecimalIP $mm
$first = Get-NetworkAddress $ip $mask
$dmask = ConvertTo-DecimalIP $mask
$dfirst = [long](ConvertTo-DecimalIP $first) + 1
$n = [long]$dmm - [long]$dmask

for ($i=0; $i -le ($n - 2); $i++) {
    $new = ConvertTo-DottedDecimalIP $dfirst
    $Port135Open = check_open_port $new "135" "1000"
    if ($Port135Open) {print_message $outputFile $new}
    $dfirst ++
}
```

10.2.47 XMPP (Расширяемый протокол обмена сообщениями

Драйвер устройства 518 Расширяемый протокол обмена сообщениями о присутствии (XMPP) позволяет AtomMind Server получать мгновенные сообщения, а также решает проблему взаимодействия между разнородными сетями. XMPP - спецификация передачи информации в режиме реального времени на базе IP технологии и Extensible Markup Language (XML).

Как следует из названия, драйвер XMPP обладает следующими характеристиками:

• Extensible (расширяемый): можно настроить драйвер под индивидуальные потребности пользователя.

• Messaging (обмен сообщениями): использует короткие сообщения как способ взаимодействия между клиентом (т.е. пользователем) и сервером.

• Presence (присутствие): драйвер реагирует на присутствие и статус пользователя.

• Protocol (протокол): это не язык, а открытая, постоянно дорабатываемая платформа. Обмен асинхронный.

Информация о драйвере

ID плагина 518 драйвера: com.tibbo.linkserver.plugin.device.xmpp

Общие настройки

Не определены.

Настройки уровня пользователя

Не определены.

Свойства Device

НАСТРОЙКИ ПОДКЛЮЧЕНИЯ

Настройки подключения определяют, как AtomMind Server взаимодействует с XMPP сервером. Доступны следующие свойства соединения:

Свойство	Описание
Протокол	Драйвер работает с двумя специфичными расширяемыми протоколами XMPP: • XEP-0325. Более подробно см. информацию по ссылке https://xmpp.org/extensions/xep-0325.html. • XEP-0323. Более подробно см. информацию по ссылке https://xmpp.org/extensions/xep-0323.html.
Адрес	IP-адрес или имя хоста, предоставляющего сервис XMPP.
Порт	Номер порта хоста, предоставляющего сервис XMPP.
Домен	Укажите XMPP домен (то, что следует за знаком '@' в XMPP-адресах (JIDs)).
Ресурс	укажите ресурс, на который будут направляться запросы с сервера.
Имя пользователя	Имя пользователя, которое будет использовано при авторизации. Имя пользователя - обычно локальная часть JID клиента. Однако, некоторые SASL механизмы или сервисы могут потребовать другой формат (например, полный JID) авторизационного имени.
Пароль	Пароль для авторизации.
Получатель	ID получателя. ID должен быть указан без доменного имени.
Данные для отправки	Определяет ваши данные для отправки. Они отправляются как строка по формату `name=value` и отделяются точкой с запятой (например, `name1=value1;name2=value2`). Свойство `value` может быть одного из следующих типов: Integer, Double или Boolean.
Таймаут	Определяет, как долго будет ждать сокет до установления TCP соединения (в миллисекундах).
Использовать TLS	Устанавливает использование режима безопасности TLS при установлении соединения. Режим по умолчанию - `SecurityMode.ifpossible`.

Активы Device

Драйвер не поддерживает активы.

Настройки Device

Драйвер XMPP создает одну переменную настроек Device. Переменная включает следующие поля:

Свойство	Описание
Успешно	Показывает успешные подключения драйвера устройств XMPP.
Время ответа, миллисекунд	Время ответа сервера.
Код состояния	Код состояния XMPP.
Ответ	Текст XMPP ответа.
Ошибка	Текст ошибки, либо NULL, если запрос был успешным.

Операции Device

ВЫПОЛНИТЬ XMPP-ЗАПРОС

Операция используется для выполнения запроса драйвером XMPP.

События Device

Драйвер не предоставляет событий.

Подключение

Драйвер переводит устройство в режим **онлайн**, при установлении успешного соединения с получателем.

10.3 Агенты

Агент AtomMind - одна из самых сложных и разносторонних концепций AtomMind. В этой части руководства мы попытаемся объяснить его назначение, функцию и важность.

Следует пояснить сам термин Агент. Смысл термина вдохновлен следующим значением слова:

Агент: кто-то представляющий кого-то, официальный представитель кого-либо в бизнесе. [Microsoft® Encarta® 2006. © 1993-2005 Microsoft Corporation.]

В нашей документации Agent имеет одно из следующих обозначений (в зависимости от контекста):

1. Особое приложение, взаимодействующее с сервером AtomMind Server по протоколу 2119 AtomMind. Это приложение конвертирует данные, собираемые с оборудования в формате AtomMind и/или конвертирует сторонние коммуникационный протоколы в протокол AtomMind. Это приложение может быть реализовано практически на любом языке программирования, в то время как ТВЭЛ предоставляет эталонные реализации на ТВЭЛ BASIC, Java (включая версию для Android) и .NET (включая .NET Compact).
2. Внешнее оборудование (программируемый контроллер или PC) с запущенным в нём приложением Agent.

Агент AtomMind - это программная библиотека, наследованная в прошивку устройства или отдельное небольшое приложение, работающее на модуле связи в составе устройства. Такое программно-аппаратное сочетание выступает посредником между "основной" прошивкой аппаратного устройства и AtomMind Server. Связь между AtomMind Server и агентом устанавливается по IP-сети. Агент осуществляет коммуникации с сервером и преобразует внутренние структуры данных устройства в формат единой модели данных 41.

Остановимся более подробно на второй из этих функций:

При использовании Агента конвертация данных происходит на стороне устройства. Агент конвертирует внутренние данные устройства и данные с его аппаратных интерфейсов ввода-вывода в протокол 2119 AtomMind, и отправляет их на AtomMind Server. При получении команд отAtomMind Server, Агент использует их для управления и настройки устройства. По сути, приложение Агент состоит из двух основных частей:

1. **Часть, специфичная для устройства:** это часть кода, специально написанная для определенного аппаратного устройства, подключенного к аппаратной платформе Агента. *Это единственная часть, которая должна модифицироваться, чтобы AtomMind работал с любым новым устройством.*
2. **Сетевая часть:** эта часть кода обрабатывает все коммуникации с AtomMind Server. Это общая часть. По сути, она взаимодействует с AtomMind Server, используя коммуникационный протокол AtomMind 2119. *Эта часть никогда не меняется для работы AtomMind с новым устройством.*

Можно рассматривать AtomMind в качестве аппаратного драйвера или конвертера протоколов. Если вы хотите подключить какое-либо устройство к AtomMind, чтобы воспользоваться всей мощью и преимуществами системы, все, что вам нужно - это включить библиотеку Агента с открытым исходным кодом в вашу прошивку/ПО, и все, ваше устройство полностью подключено!!

Агент предоставляет операторам AtomMind несколько способов контроля, мониторинга и настройки подключенных устройств:

- Он раскрывает AtomMind Server внутренние настройки вашего устройства таким образом, что настройки типа *Число Хранящихся Записей* (для таймера) или *Максимально Разрешенная Скорость* (для погрузчика) будут легко отображаться и редактироваться в любом пользовательском интерфейсе AtomMind Server (например, AtomMind Client 359).
- Он позволяет выполнять различные операции с устройством, например *Выполнить самотестирование* или *Отправить сообщение на LCD устройства*. Операторы могут указывать входные параметры для каждой операции и видеть статус их выполнения.
- Он помогает отслеживать события, сгенерированные устройством, например, *Пользователь вошёл в область* или *Пользователь покинул область* для терминала контроля доступа. AtomMind Server хранит все события устройств в постоянном хранилище и предоставляет доступ к истории событий системным операторам.

Драйвер устройства Agent

Agent имеет двойника на AtomMind Server - Драйвер Устройства 518, называемый Драйвер Агент 523. Этот драйвер отвечает за все коммуникации между AtomMind Server и Агентом.

Использование Агента в качестве центрального аппаратного компонента

Поскольку ТВЭЛ предоставляет полный спектр мощных программируемых модулей и контроллеров вместе с бесплатной средой IDE, другим вариантом будет создание вашей системы вокруг процессора ТВЭЛ с запущенным на нем приложением Agent.

В этом случае вы можете подключить периферийные компоненты вашей системы (LCD, клавиатуру и пр.) прямо к процессору ТВЭЛ и использовать его как ЦПУ. Затем вы пропишете логику своего устройства на языке ТВЭЛ BASIC или C. Таким образом вы получите новое устройство со стандартным функционалом AtomMind.

Настройка Агента для Ваших устройств

Приложение Агент нужно настроить для подключения любого уже существующего или вновь созданного устройства к AtomMind. Вам только нужно включить библиотеку Агента в ваше ПО/прошивку и предоставить:

- Определения настроек, операций и событий вашего устройства,
- Код, который реализует чтение и запись настроек устройства,
- Код для перенаправления команд на устройства,
- Код для опроса событий с ваших устройств или код для асинхронного получения событий и отправки их на сервер.

SDK Агента

Более подробно о реализации Агента как части приложения на ПК, основанного на технологиях Java или .NET, см. в разделе SDK Агента 1360.

10.3.1 Общая информация об Агенте

Приложение Agent может использоваться для:

- Представления одного или нескольких устройств в AtomMind (например, кассовый аппарат на основе ПК с приложением на .NET, использующий библиотеку Агента для связи с AtomMind Serverом, предоставляя статистику поставок и удалённую обработку операций).

- Для взаимодействия с одним или более внешних устройств по параллельному порту, Wi-Fi, GPRS или других типов связи, делая его доступным для AtomMind (например, программируемый модуль, собирающий данные с сенсоров температуры, влажности и давления и делая эти данные доступными внутри AtomMind).

НЕОБХОДИМЫЕ ЗНАНИЯ

Для подключения Ваших устройств к AtomMind через Агента, Вам необходимо быть знакомым с принципами работы системы такими, как представление устройств как контекстов, содержащих набор переменных, функций и событий.

Мы советуем прочесть следующие основные темы перед тем, как начинать делать собственные изменения:

- Обзор платформы 38
- Подключение и сбор данных 494
- Устройства 497
- Общая информация об Агентах 684
- Драйвер Агента 523

Крайне рекомендуется изучить внутреннее представление данных устройства в системе:

- Контексты 41
- Переменные 61
- Функции 70
- События 73
- Таблицы данных 49

Также Вам было бы полезно взглянуть на две статьи, посвящённые связи между AtomMind Serverом и Агентом:

- Протокол связи AtomMind 2119
- Кодировка таблицы данных 2123

10.3.2 Ключевые принципы Агента

Наиболее важные принципы реализованы в приложении Agent и соответствующего ему драйверу AtomMind Servera 523:

- Приложение Agent устанавливает соответствие данные аппаратуры **переменным**, **функциям** и **событиям** для одного контекста 41 (контекста Агента).
- AtomMind Server создает контекст устройства 1494 для имитации переменных, функций и событий контекста, предоставляемого приложением Агента.
- Все переменные (настройки) предоставляемые Agent читаются и кэшируются на стороне сервера (см. синхронизация 514).
- Все запросы управления (вызовы функций) к AtomMind Server'ному контексту устройства перенаправляются агенту для исполнения. Ответы отправляются обратно AtomMind Servery.
- AtomMind Server получает и сохраняет все события, полученные от Агента.

Запуск Агента

1. Когда программируемый контроллер с выполняющимся на нем Агентом включается, или приложение Агента стартует в ПК, оно пытается открыть соединение с AtomMind Serverом.
2. Если соединение установлено, он выполняет последовательность входа посылая некоторые команды на сервер и проверяя ответы сервера. Последовательность включает проверку пароля AtomMind, хранящегося на стороне сервера у учётной записи устройства.
3. После этого AtomMind Server начинает полную синхронизацию 525. Он читает информацию о контексте, предоставляемую Агентом вместе с **определениями** его переменных, функций и событий, синхронизирует часы реального времени Агента с серверным временем и переключается в "рабочий режим".

В рабочем режиме AtomMind Server:

- Периодически **синхронизирует значение переменных** между контекстами сервера и агента;
- **Перенаправляет запросы** в Agent и получает результаты выполнения;
- **Получает события устройства** от Агента обрабатывает/сохраняет их.

Нормальный режим связи с AtomMind Server

В этом режиме Agent получает команды от AtomMind и обрабатывает их. Он также может асинхронно генерировать некоторые события и отправлять их на сервер.

Элементы поддержки Протокола Связи с AtomMind:

Входящие Сообщения (от сервера):

- **Стартовое Сообщение**. Обрабатывается автоматически модулем взаимодействия AtomMind Servera.
- **Операционное Сообщение**. Разбирается и обрабатывается модулем взаимодействия AtomMind Server.

Исходящие Сообщения (к серверу):

- **Сообщение События**. Отправляется на сервер асинхронно модулем взаимодействия AtomMind Servera когда специфичный для устройства код решает сгенерировать сообщение.

Ответы:

- **Успех**
- **Отказ**
- **Ошибка**

Все ответы генерируются модулем взаимодействия AtomMind Servera. Ответ Отказ может возвращаться только в случае, если версия Агента несовместима с версией AtomMind Servera. В этом случае Agent будет неспособен взаимодействовать с сервером.

Операции:

- **Получить значение переменной.**
- **Установить значение переменной.**
- **Вызвать функцию.**

Они автоматически обрабатываются модулем взаимодействия AtomMind Servera. Если операция связана с переменной или функцией, объявленной в модуле взаимодействия AtomMind Servera (например, синхронизация часов реального времени между сервером и Агентом, соответствующая команда автоматически обрабатывается этим модулем. В противном случае исполнение команды доверяется модулю устройства, который может:

- Получить данные из подключенного устройства, EEPROM, flash-диск, память или другой источник и отправляет их на сервер (как описывается ниже) для **получения значения переменной**.
- Отправить данные оборудованию или сохранить их, **установив значение переменной**.
- Выполнить обработку данных или отправить команды подключенному оборудованию, **вызвав функцию**.

10.3.3 Представление устройств в Агенте

Данные устройства представляются в Агенте как будто они представлены в самом AtomMind Servere. Агент предоставляет "нормализованное" представление для каждого устройства для других частей AtomMind. Это представление называется контекст 41. В каждом контексте имеются переменные 61 (настройки устройства), функции 70 (команды, допустимые для этого устройства) и события 73, которые это устройство может производить.

Значения переменных, параметры функций ввода/вывода и данные, связанные с экземплярами событий также представлены в "нормализованной" форме, в виде Таблицы Данных 49. Каждая таблица данных имеет **формат**, определяющий дозволенное число записей в таблице, её имя, тип и прочие параметры для каждой из колонок вместе с остальной необходимой информацией.

Каждый контекст (т.е. устройство) обязан предоставлять **определения** своих переменных, функций и событий. Определение содержит метаданные переменной, функции или события, такие как *описание*, формат и др. информацию.

10.3.4 Возникновение события

Большинство устройств генерируют разнообразные *события*. Обычно эти события отражают изменения статуса устройств или условий его работы. Приложение Agent может использоваться для опроса событий с внешнего оборудования точно так же, как может и генерировать эти события самостоятельно. Эти события представлены как события контекста 73 в контексте, предоставляемом Агентом.

Подтверждение события

Некоторые агенты требуют надежный способ доставки событий. Такие агенты описывают функцию confirmEvent() 689, которая будет вызываться сервером каждый раз при её подключении и успешной обработке несистемного события из Agent.

Асинхронные обновления серверного кэша

Иногда необходимо "протолкнуть" новое значение из какой-то изменённой Агентом переменной в AtomMind Server и обновить кэш настроек 502 без ожидания цикла синхронизации 514. В этом случае, Agent должен сгенерировать событие Обновление переменной 84 для изменённой переменной. Это событие содержит имя и новое значение переменной. AtomMind Server положит это значение в серверный кэш в момент получения **обновлённого** события если настройки синхронизации для этой переменной не блокируют синхронизацию между устройством и сервером.

10.3.5 Контексты Агента

Эта секция описывает специфичные для Agent переменные 61, функции 70 и события 73 доступные в контексте, предоставляемом Agent.

Открытые переменные (свойства)

ДАТА

Эта переменная должна быть описана вручную в контексте Agent, если AtomMind Server выполняет синхронизацию часов с Агентом в режиме реального времени. Если Agent описывает переменную **Date**, сервер записывает её с текущей датой / временем при каждом соединении между Agent и AtomMind Server.

Имя переменной: date

Записи: 1

Доступность: корневой контекст

Формат 49 записи:

Имя поля	Описание поля	Тип поля	Примечания
date	Значение часов реального времени платы программируемого контроллера или ПК с запущенным на нем Агентом.	Дата	

ВРЕМЯ МОДИФИКАЦИИ

Эта переменная содержит метки времени последней модификации для каждой из настроек устройства (переменной) в контексте устройства Agent. Эти метки обычно хранятся в EEPROM для сохранения состояния синхронизации между перезагрузками устройства.

После каждой синхронизации настроек устройства между AtomMind Serverом и Agent сервер обновляет эти метки времени, сохранённые в Agent. Когда какая-либо настройка устройства была изменена "изнутри" (напр., через клавиши или сенсорный экран устройства), Agent должен обновить время модификации переменной чтобы не допустить её затирание старым значением при последующей синхронизации с кэшом сервера.

В некоторых случаях Agent может не поддерживать отметки времени для некоторых или всех настроек устройства. В таких случаях переменная **modtime** не должна содержать метки времени для них или метки времени должны быть установлены в NULL.

Имя переменной: date

Записи: 0...неограничено

Доступность: Контекст устройства

Формат 49 записи:

Имя поля	Описание поля	Тип поля	Примечания

variable	Имя настройки устройства	Строка	
modtime	Дата/время последнего изменения значения этой переменной.	Дата	Может быть равно Null

Открытые функции [?70]

СИНХРОНИЗИРОВАНО

Сообщает Агенту, что синхронизация с AtomMind Server завершена, и он может начать отправку событий687.

Имя функции: synchronized

Входные записи: 0

Формат49 ввода: нет

Выходные записи: 0

Формат49 вывода: нет

ПОДТВЕРДИТЬ СОБЫТИЕ

Если эта функция определяется Агентом, AtomMind Server вызывает её каждый раз при её получении и успешной обработке несистемного события.

Имя функции: confirmEvent

Записи ввода: 1

Формат50 ввода:

Имя	Тип	Описание
id	Длинное	Идентификатор Агента - сгенерированное событие, которое было успешно получено, сохранено и обработано сервером

Записи вывода: 0

Формат50 вывода: нет

ПОЛУЧИТЬ ИСТОРИЮ

Если эта функция определяется Агентом, AtomMind Server вызывает её в начале каждой синхронизации. Он возвращает список исторических значений, помещённых в буфер Агентом, когда соединение с сервером было недоступно. Если функция вернула хотя бы одну запись, она будет снова вызвана, позволяя вернуть исторические значения в предпочитаемом Вами размере.

Значения отдельной переменной должны быть представлены в хронологическом порядке как в пределах одной таблицы результатов getHistory(), так и между последующими вызовами getHistory(). Несоблюдение этого вызовет ошибки в статистических каналах718 AtomMind Server .

Имя функции: getHistory

Записи ввода: 0

Формат50 ввода: нет

Записи вывода: 0... не ограничено

Формат50 вывода:

Имя	Тип	Описание
variable	Строка	Имя переменной, чьё историческое значение представлено записью.

timestamp	Дата	Временная метка исторического значения, т.е. дата/время, когда оно было получено или сгенерировано самим Агентом.
value	Таблица данных	Историческое значение переменной.

10.3.6 Взаимодействие Агента и AtomMind Servera

В этой статье описывается, как Agent взаимодействует с AtomMind Serverом по протоколу взаимодействия с AtomMind 2119. Взаимодействие начинается после того, как Agent подключается к AtomMind Server и производит вход. В этот момент управление Agent передаётся драйверу AtomMind Агента 523. Затем AtomMind Server начинает отправку команд по протоколу взаимодействия с AtomMind чтобы выяснить, какие данные предоставляются Agent.

Покомандная схема взаимодействия

A. ВЫЯСНЕНИЕ КОНТЕКСТА AGENT

Сначала AtomMind Server получает информацию о контексте, объявленном в Agent. Посылаются следующие команды:

1. Получить переменную 2122 info 63
2. Получить переменную children 63
3. Получить переменную variables 64
4. Получить переменную functions 64
5. Получить переменную events 65

B. СИНХРОНИЗАЦИЯ ЧАСОВ РЕАЛЬНОГО ВРЕМЕНИ

1. Установить переменную 2122 дата 688 контекста Агента в текущее время в соответствии с временной зоной 511 устройства.

C. СИНХРОНИЗАЦИЯ НАСТРОЕК УСТРОЙСТВА (ЗНАЧЕНИЙ ПЕРЕМЕННЫХ КОНТЕКСТА УСТРОЙСТВА)

Теперь AtomMind Server синхронизирует значения всех переменных устройства между своим кэшем и Agent. Более подробную информацию о синхронизации и кэшировании настроек устройства можно прочесть в статье о драйвере AtomMind Агента 525

1. Получить переменную modtime 688. Сервер получает временные метки всех настроек для определения верного направления синхронизации (от устройства к серверу или наоборот) для каждой из них.
2. Для каждой переменной выполняются команды Получить переменную или Установить переменную в зависимости от направления синхронизации. Эти команды используются для установки значений переменных в контексте устройства Agent (для синхронизации от сервера к устройству) или в серверном кэше (при синхронизации от устройства к серверу).
3. Установить переменную modtime 688. Сервер устанавливает временные метки модификации Agent.

D. ОКОНЧАНИЕ СИНХРОНИЗАЦИИ

1. Вызвать функцию synchronized 689 из контекста Агента.

Команды, инициируемые Агентом

Когда Agent подключен к AtomMind Server, он может посылать команды событий 2121 к серверу, когда события будут сгенерированы 687.

AtomMind Server никогда явно не начинает или оканчивает прослушивание событий от Агента посылкой команд Add Event Listener 2124 или Remove Event Listener 2125 протокола взаимодействия с AtomMind. он прослушивает события без уведомления об этом Агента. Поэтому Agent может начать отправку событий любого типа сразу после окончания синхронизации, например после того, как сервер вызвал функцию synchronized 689 из корневого контекста Агента.

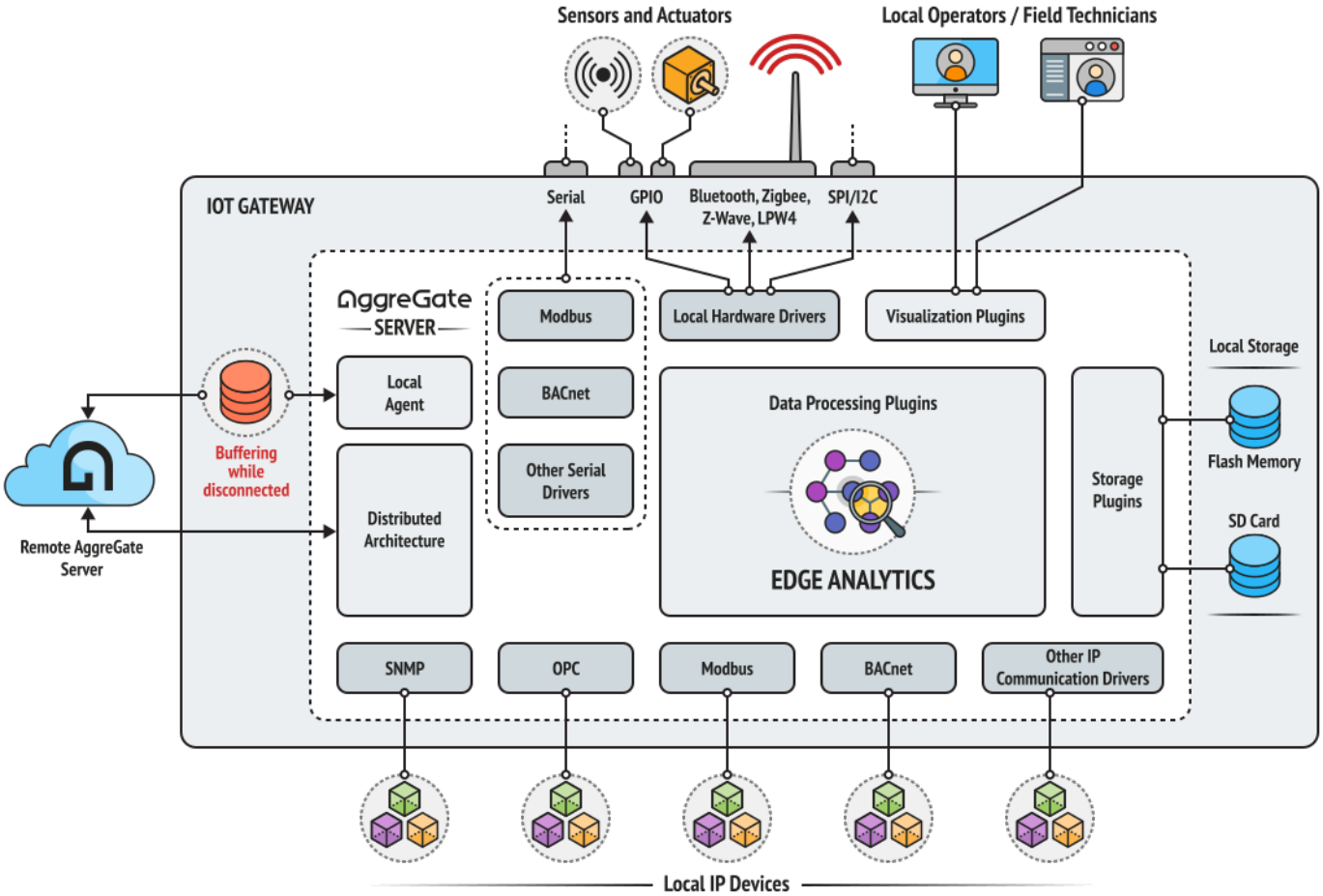
10.4 Пограничные шлюзы

AtomMind переопределяет парадигму периферийных вычислений, позволяя использовать одно и то же программное обеспечение как на конечных устройствах, так и на облачных серверах.

AtomMind Server может быть запущен на одноплатных ПК, IoT-шлюзах, промышленных ПК, сенсорных панелях, ПЛК и практически на любых устройствах с операционными системами, поддерживающими Java (Linux, Windows, и т.д.). С точки зрения функционала не будет никакой разницы, и единственное, что надо учитывать - это ограничения масштабируемости конечных IoT устройств.

Встроенный AtomMind Server становится сердцем IoT-шлюза. Он отвечает за сбор, хранение, обработку и визуализацию информации, используя всю мощь инструментария AtomMind.

Такой IoT-шлюз может передавать информацию на сервера AtomMind более высокого уровня через так называемый локальный агент 574, а также может быть объединён с другими серверами через распределенную архитектуру 1332, обеспечивая доставку агрегированных и обработанных данных в облако.



IoT-шлюзы идеально подходят для задач, в рамках которых требуется организовать мониторинг, управление и оценить возможности развития большого количества удалённых объектов, таких как телекоммуникационные вышки, насосные станции или электрические трансформаторы. Система, основанная на развёрнутых на конечных устройствах серверах AtomMind, предоставляет полностью централизованное управление, лёгкую модернизацию и продвинутое управление изменениями для тысяч объектов.

Поскольку нет разницы между пограничной и облачной версиями AtomMind Server, в настоящей документации нет отдельного большого раздел, посвященного Edge. Все возможности AtomMind доступны и на периферии - испытайте их!

11 Хранение данных

Эта глава описывает хранение и агрегирование данных с устройств, а также хранение настроек системы, событий и других типов данных.

AtomMind Server использует разные типы *хранения данных* для постоянных данных:

- **Настройка,** т.е. свойства устройств и объектов системы
- Хронологические **события** (включая хронологические значения переменных, которые сохранены в форме *событий изменения*)
- **Бинарные данные**, такие как звуки, изображения, документы и т.д.
- **Статистика**, такая как средние значения каждую минуту или максимальные и минимальные ежемесячные показатели изменений значения.
- **Топологии,** представленные в виде графов физических/логических объектов и их связей

11.1 Хранилища данных

Сервер может использовать разные типы хранения для разных типов данных. Ниже представлено сравнение доступных типов хранения и их возможности:

Тип хранения	Применяется для	Плюсы	Минусы
Реляционная база данных 697	Настройки, Событий, Бинарных данных	Работает из коробки. Данные доступны для сторонних приложений. Поддерживается нативная и отказоустойчивая кластеризация для AtomMind Server.	Низкая производительность вставки и обновления данных.
База данных Key-Value 697	Настройки, Бинарных данных	Очень высокая производительность вставки и обновления. Поддерживается отказоустойчивая кластеризация.	Данные недоступны для сторонних приложений.
База данных NoSQL 693	Событий	Высокая производительность вставки и обновления. Расширяемость производительности хранения посредством кластеризации хранения. Поддерживается отказоустойчивая кластеризация.	Данные недоступны для сторонних приложений.
Хранение файлов 717	Настройки, Бинарных данных	Очень низкие затраты производительности системы.	Не поддерживается отказоустойчивая кластеризация.
Отключено	Событий, Бинарных данных	Нет затрат производительности.	Данные не хранятся постоянно. Все хронологические значения не сохраняются.
Кольцевая база данных 718	Статистики (совокупных временных рядов)	Постоянное отслеживание данных. Высочайшая производительность получения вставки/обновления.	Могут храниться только показатели временных рядов числовых значений.

Настройка хранения по умолчанию

По умолчанию AtomMind Server настроен, чтобы сохранять данные в Базе данных NoSQL 693 и Базе данных Key-Value 697.

Раздел базы данных 182 глобальной настройки сервера предоставляет возможность для независимого типа хранения изменений для каждого вида сохраненных данных.

11.1.1 База данных NoSQL

Этот раздел описывает, как AtomMind Server использует базу данных NoSQL для хранения данных.

База данных NoSQL - это движок БД Apache Cassandra, который работает либо внутри сервера, либо как самостоятельное приложение (кластеризованное или некластеризованное). Cassandra является хранилищем больших массивов данных, которые обеспечивают очень высокую скорость вставки/обновления данных.

11.1.1.1 Схема и взаимодействие базы данных NoSQL

Схема базы данных AtomMind Server полностью динамическая, сервер создает и удаляет новые таблицы по ходу работы, чтобы отражать создание, удаление и перенастройку ресурсов системы.

Однако, несколько базовых таблиц присутствуют в любой инсталляции:

Таблица	Описание
context_directory	Содержит идентификаторы (УУИД) контекстов 41, которые используются для преобразования и извлечения данных из других таблиц. Поля: • **key** - полный путь контекста 41 • **column1** - идентификатор (УУИД) контекста 41, генерируется автоматически • **value** - время создания в миллисекундах, закодированное в типе данных BLOB
ag_properties	Содержит значения всех постоянных переменных контекста 61 сервера. AtomMind Server генерирует относительно небольшое количество запросов типа ДОБАВИТЬ и УДАЛИТЬ к этой таблице, когда создаются или удаляются объекты системы. Количество запросов ВЫБРАТЬ также относительно невелико из-за кэширования в памяти сервера. Однако, таблица ag_properties получает огромное количество запросов ОБНОВИТЬ, так как значения свойств постоянных ресурсов меняются, а сервер сохраняет значения изменений в базе данных. Например, устройства 497, использующие постоянную переменную кэш 502, вызывают большое количество обновлений в таблице ag_properties во время синхронизации 514. Поля: • **key** - идентификатор (УУИД) контекста 41, значение чьих переменных хранится в строке таблицы • **property** - имя переменной • **value** - таблица данных 49, представляющая значение переменной, закодированной как string 2123, с использованием невидимых разделителей
ag_events	Содержит все события контекста 73 за исключением тех, которые хранятся в других (пользовательских) таблицах. Подробнее об этом см. в разделе **Хранение событий контекста** ниже. Поля: • **key** - Имеет формат идентификатор (УУИД):имя, где первая часть - это идентификатор контекста 41, в котором возникло событие, а вторая - имя события (т.е. тип события) • **column1** - УУИД с привязкой ко времени, содержит закодированную временную метку, когда произошло событие, и идентификатор события • **value** - закодированная информация о событии, которая содержит:

	▪ **expirationtime -** временная метка в будущем, соответствующая планируемому окончанию события ▪ **level -** уровень события ▪ **permissions -** настроенные права доступа, необходимые для доступа к экземпляру события, либо null, если должны использоваться права доступа, указанные в определении события ▪ **count -** количество дубликатов зарегистрированных событий ▪ **ack -** закодированный список подтверждений события ▪ **enrichments -** закодированный список обогащений события ▪ **format -** событие в закодированном формате, либо null, если должен использоваться формат, указанный в определении события ▪ **data -** таблица данных, в которой представлены данные события, закодированные в Byte array
ag_data	Содержит блоки 52 бинарных данных, встроенные в значения переменных контекста 61. Эти блоки извлекаются в данную отдельную таблицу, когда значение переменной записывается в таблицу ag_properties. Однако, бинарные блоки не загружаются сразу при загрузке значения переменной из ag_properties. Блоки загружаются только по необходимости, т.е. через конкретные запросы от модулей AtomMind Server, клиентов или внешних приложений. Поля: • **key** - идентификатор (УУИД) контекста 41, чьи данные хранятся в строке таблицы • **value** - бинарные данные

Хранение событий контекста

Большая инсталляция AtomMind может хранить миллиарды постоянных событий контекста 73 в базе данных. Эти события динамически распределяются между разными таблицами, чтобы повысить производительность сохранения и извлечения.

Формат всех таблиц событий соответствует формату таблицы ag_events, описанному выше.

Доступ к таблицам событий обычно включает множество операций ДОБАВИТЬ. Новая запись добавляется каждый раз при регистрации постоянного события сервера. Команды УДАЛИТЬ применяются к таблице событий в двух случаях:

- Если ресурс системы со всеми соответствующими событиями удален
- Если просроченные события удалены в рамках периодической задачи планировщика 823

Количество запросов ВЫБРАТЬ, обращенных к таблице событий, относительно мало. События загружаются в следующих случаях:

- Если открыт Журнал событий 398, что вызвало активацию фильтра событий 762
- Если события загружены напрямую при помощи функции get() 1516 контекста **События**
- Если история переменной напрямую загружена через функцию variableHistory() 1613 контекста **Утилиты**
- Во время начальной визуализации графика виджета 1051, настроенного так, чтобы включать исторические события или значения переменных
- В некоторых других подобных случаях

Несмотря на то, что события загружаются нечасто, количество событий, загружаемых одним запросом, не ограничено платформой, т.к. для некоторых инсталляций необходимо загружать миллионы событий за раз. Системные архитекторы, разрабатывающие решения на базе AtomMind, должны помнить об этом и продумать максимальное количество загружаемых событий при создании конвейера обработки данных на стороне сервера, а также инструментальных панелей визуализации.

Попытка загрузить слишком много событий сразу приведет к переполнению памяти AtomMind Server, что проявится в резком снижении производительности из-за чистки памяти виртуальной машины Java. Это может вызвать полное зависание сервера или ошибки внутренней памяти.

Список таблиц событий, приведенный в документации, не является исчерпывающим. Разные модули и плагины AtomMind Server могут запросить создание других особых таблиц событий, не упомянутых здесь.

Все события, для которых нет явных настроек использовать пользовательскую таблицу, записываются в таблицу ag_events. Ниже представлен список таблиц БД, которые обычно создаются для хранения событий разного типа.

Структура таблиц такая же, как у таблицы ag_events.

Таблица	Описание
ag_info	Содержит события Информация 77.
ag_alert	Содержит события Тревоги 790.
ag_*xxx*_change	Содержит все события Изменение 84, показывая изменения переменных в одном контексте устройства 1494 (соответствует устройству *xxx*). Обратите внимание, что имя таблицы может быть сокращено из-за ограничения длины имени таблицы БД на самом сервере.

Прямой доступ к базе данных AtomMind Server

Прямой доступ сторонних приложений к базе данных AtomMind Server крайне нежелателен. AtomMind - мощная платформа, предлагающая полноценный комплект разработчика и различные виды API для локального и удаленного доступа ко всем элементам данных, содержащимся в БД сервера. Более того, доступ через API всегда будет наследовать надлежащие блокировки, проверку прав доступа 477 и оптимизацию производительности.

Прямое изменение данных в БД AtomMind Server в большинстве случаев приведет к некорректному поведению системы. Однако, в редких случаях допускается использование операций прямого чтения (ВЫБРАТЬ).

Пожалуйста, обратитесь в ТВЭЛ, чтобы узнать, как избежать прямого доступа к БД.

11.1.1.2 Настройка хранилища NoSQL

Настройка хранилища NoSQL

Отдельная группа настроек в разделе базы данных 182 глобальных настроек сервера позволяет конфигурировать опции производительности и кластеризации для хранилища NoSQL.

Чтобы активировать базу данных NoSQL, измените свойство **Хранилище событий** на **Хранилище NoSQL**. Обычно не требуются дополнительные изменения конфигурации, если единственная база данных NoSQL используется без отказоустойчивого кластера.

Чтобы использовать базу данных NoSQL при наличии отказоустойчивого кластера, нужно:

- Настроить два или более экземпляров AtomMind Server
- Активировать в них хранилище NoSQL
- Установить **Базы данных кластера NoSQL** на каждый сервер, чтобы указывать на другие серверы в кластере
- Перезагрузить все серверы

Папка хранилища NoSQL

По умолчанию база данных NoSQL хранит данные в подпапке `/nosql_data` папки инсталляции AtomMind Server.

Поэтому чтобы передать все постоянные события из одного устройства на другое, необходимо будет скопировать папку `/nosql_data` в целевой сервер, заменяющий существующую папку.

11.1.1.3 Работа с внешней базой данных Cassandra

Внедрение высокопроизводительного внешнего экземпляра Cassandra

1. Установите Java 8 или JDK 8.
2. Установите (распакуйте) самую последнюю версию Cassandra.
3. Редактируйте `config/cassandra.yaml`
   - Измените **каталог данных** и **каталог журнала регистрации** (предпочтительно на разных устройствах).

- Укажите IP-адрес сервера Cassandra в **seeds**, **listen_address** и **rpc_address**.
- Установите **start_rpc** на true.

4. Установите переменную среды `JAVA_HOME` в корневую папку JRE/JDK.
5. Установите переменную среды `CASSANDRA_HOME` в папку инсталляции Cassandra.
6. Редактируйте `bin/cassandra`
   - Измените настройки **Xms** и **Xmx** на самое высокое значение, оставив 2-4 Гб памяти для операционной системы.
7. Запустите `bin/cassandra` и убедитесь, что Cassandra начинает слушать входящие соединения.

Оптимизация производительности Cassandra для загрузки с высокой интенсивностью записи

Загрузка с высокой интенсивностью записи - это множество операций вставки. Чем быстрее вы вставляете данные, тем быстрее вам нужно сжать их таким образом, чтобы обеспечить устойчивый отсчет. Поэтому вам нужно редактировать следующие параметры в `cassandra.yaml`:

- Увеличить количество **concurrent_compactors**.
- Увеличить значение **compaction_throughput_mb_per_sec** или отключить регулирование путем установки этого параметра на значение ноль.

Также вам может понадобиться изменить стратегию сжатия определенных таблиц. Могут потребоваться изменения следующих параметров:

- **compaction** - предпочтительная стратегия здесь - это **TimeWindowCompactionStrategy.** Создана специально для временных рядов и истекающих рабочих загрузок TTL. Для индивидуализации необходимо только изменить **compaction_window_unit** и **compaction_window_size**. Также **unchecked_tombstone_compaction** необходимо установить на `true` , чтобы заставить Cassandra сбросить истекшие *SSTables* в режиме реального времени.
- **gc_grace_seconds** - должно быть снижено или установлено на ноль (в случае, если вы используете кластер с одним узлом).

Чтобы изменить параметры уплотнения, нужно выполнить запрос, используя интерактивный терминал CQL. Запустите `bin/cqlsh`, чтобы подключить текущий узел Cassandra:

```
USE aggregate;
ALTER TABLE <table_name>
  WITH compaction = {'class' : 'TimeWindowCompactionStrategy', 'compaction_window_unit' : 'HOURS',
  AND gc_grace_seconds = 0;
```

Перед настройкой вашей базы данных, используя перечисленные параметры, проверьте документацию Apache Cassandra. Настройка базы данных может отличаться в отдельных случаях.

Другие параметры, на которые следует обратить внимание

НАСТРОЙКИ ТАЙМАУТА СЕТИ

- range_request_timeout_in_ms
- read_request_timeout_in_ms
- write_request_timeout_in_ms
- request_timeout_in_ms
- и т.д.

Учитывайте изменения этих настроек, когда вы планируете произвести операции, требующие временных затрат. Иначе вы можете получить **TimedOutException** в течение работы с базой данных.

11.1.2 Хранилище «ключ-значение»

Хранилище "ключ-значение" управляется СУБД Berkeley, встроенной в AtomMind Server и работающей внутри сервера. Она идеально подходит для сохранения конфигураций системы и устройств.

Конфигурация хранилища "ключ-значение"

Отдельная группа настроек в разделе база данных 182 общих настроек сервера, позволяет настраивать параметры производительности и отказоустойчивой кластеризации для хранилища "ключ-значение".

Для активации хранилища "ключ-значение", измените свойство **Configuration Storage** на **Key-Value Storage**. Как правило, дополнительных изменений конфигурации не требуется, если используется одна база данных без отказоустойчивого кластера.

Чтобы хранилище "ключ-значение" в сценарии отказоустойчивого кластера:

- Настройте два или более экземпляра AtomMind Server
- Запустите на них хранилище "ключ-значение"
- Заполните параметр **Key-Value Cluster Databases** на каждом сервере, в это параметре должны быть указаны адреса других элементов кластера
- Перезагрузите все серверы

Папка хранилища "ключ-значение"

Хранилище "ключ-значение" хранит данные в подпапке `/key_value_data` папки установки AtomMind Server.

Таким образом, для копирования конфигураций сервера с одной машины на другую необходимо скопировать папку `/key_value_data` на целевой сервер, заменив существующую папку.

11.1.3 Реляционная база данных

Этот раздел описывает, как AtomMind Server использует классическую реляционную базу данных для хранения данных.

AtomMind Server общается с реляционной базой данных, используя драйвер *JDBC*, поэтому он может работать с большинством современных систем управления базами данных.

JDBC это Java DataBase Connectivity API, стандартизированный способ взаимодействия Java приложений с разнообразными базами данных и их источниками.

JDBC отличается от ODBC (Microsoft's Open DataBase Connectivity) главным образом в том моменте, что JDBC пишут на языке Java, поэтому его можно использовать без изменений в кросс-платформенной среде. В добавление к этому, учитывая, что ODBC является сложным стандартом, он технически устаревает, в то время как JDBC является современным, стирает технические требования для доступа БД разных производителей.

Встроенная база данных

Дистрибутив AtomMind Server включает встроенную базу данных (Apache Derby), которая используется по умолчанию. Эта база данных предназначена для ознакомительных целей.

Встроенная база данных не подходит для долговременной промышленной эксплуатации! Использование встроенной базы данных в эксплутационных установках может привести к высокой загрузке памяти и проблемам производительности. Необходимо переключиться на любую базу данных уровня предприятия, если система используется промышленно.

Обратитесь к разделу переключение на другой движок баз данных 698 для получения дополнительной информации как изменить базу данных, используемую сервером.

Отдельная группа настроек общей конфигурации сервера 182 управляет взаимодействием AtomMind Server с движком баз данных.

Данные встроенной базы данных расположены в подпапке `/database` в папке установки AtomMind Server.

Некоторые плагины (Classes, CMDB, NetFlow) требуют для работы БД SQL. Для них будет запущен движок встроенной БД (Apache Derby). Если вы хотите полностью деактивировать движок встроенной БД, пожалуйста, удалите эти плагины из папки AtomMind Server.

ПЕРЕКЛЮЧЕНИЕ НА ВНЕШНИЙ ДВИЖОК РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

Если AtomMind Server установлен с базой данных Apache Derby вместо прилагаемой базы данных MySQL, вам следует учитывать переключение на внешнюю реляционную базу данных.

Все данные, которые хранятся в базе данных Apache Derby AtomMind Server, загружены в память в течение запуска сервера. Если ваш AtomMind Server потребляет много памяти с относительно маленьким количеством устройств, проверьте размер директории `/db` в каталоге инсталляции AtomMind Server. Если занимает больше, чем 100-200 Мб, вам необходимо обратить внимание на переключение на сервер внешней реляционной базы данных.

Обратитесь к статье Движок переключения на другу БД для получения информации относительно переключения реляционной базы данных.

У документации сервара БД обычно есть собственный раздел Оптимизации производительности. Обратитесь к руководству пользователя сервера базы данных для администратора за информацией о том, как максимизировать производительность базы данных в вашей среде.

Прилагаемый пакет MySQL

Некоторые комплекты поставки AtomMind Server включают в себя преднастроенный пакет MySQL. MySQL - это мощный и бесплатный сервер баз данных, который позволяет AtomMind работать с тысячами устройств.

Если во время установки выбран прилагаемый пакет MySQL, он будет загружен с сайта ТВЭЛ и установлен в подпапке `/mysql` папки установки AtomMind Server. Он уже настроен для использования с AtomMind Server, поэтому дополнительных настроек в большинстве случаев не требуется.

УПРАВЛЕНИЕ УСТАНОВКОЙ ПРИЛАГАЕМОГО MYSQL

Обычно прилагаемый сервер MySQL запускается автоматически при запуске Windows. Для его запуска/остановки вручную:

- Перейдите к **Start Menu > Control Panel > Administration > Services**
- Найдите сервис `MySQL for AtomMind`
- Щелкните по нему правой кнопкой мыши, чтобы увидеть список функций управления в контекстном меню

Если сервис MySQL service не запускается, просмотрите файл журнала ошибок (`mysql/error.log`) для выяснения деталей ошибки.

11.1.3.1 Переключение на другой движок баз данных

Этот раздел описывает, как переключить AtomMind Server на использование другого движка БД. По умолчанию AtomMind Server использует встроенный движок БД, который подходит только для небольших инсталляций (до 20-30 устройств). Для крупных инсталляций рекомендуется использовать более мощный сервер баз данных, например MySQL, Oracle, Microsoft SQL Server, PosrgreSQL, Firebird и другие.

Некоторые версии AtomMind Server содержат в качестве опционального пакета сервер *MySQL*. Если этот пакет выбирается при установке, сервер автоматически настраивается на использование MySQL. В этом случае переключение на другой движок баз данных, скорее всего, не потребуется.

Чтобы переключить AtomMind Server на другой движок баз данных, необходимо:

1. Выполнить резервное копирование конфигурационного файла AtomMind Server (`server.xml` в установочной папке AtomMind Server), скопировав его в ту же папку под именем `server_old.xml`.
2. Установить и настроить базу данных, если это еще не было сделано.
3. Создать новую базу данных, чтобы AtomMind Server мог ее использовать для хранения данных.
4. На движке БД создать учетную запись пользователя, от имени которого AtomMind Server будет осуществлять доступ к данным.
5. Выдать необходимые права новой учетной записи базы данных AtomMind Server. Потребуются права на создание и редактирование структуры таблиц (**CREATE**, **ALTER, DROP**). Также необходимо предоставить все права на вставку, обновление, удаление и выбор данных из всех таблиц (**INSERT**, **UPDATE**, **DELETE** and **SELECT)**.
6. Остановить AtomMind Server, если он запущен.
7. Положить соответствующий JDBC-драйвер в подпапку `/ jar` установочной папки AtomMind Server. Файл драйвера обычно имеет расширение **.jar**. Например, JDBC-драйвер для MySQL имеет имя `MySQL-connector-`

java-X.X.XX-ga-bin.jar, где *X.XX.XX* i- номер версии. Драйвер JDBC может поставляться как производителем сервера базы данных (Oracle, и т.д.), так и сторонними авторами.

Существует также альтернативный метод, как сделать Ваш JDBC-драйвер доступным для AtomMind Server:

- Создайте новый файл server.vmoptions в Файле Свойств Запускающего Модуля 158 в папке инсталляции AtomMind Server и откройте его для редактирования в текстовом редакторе.
- Наберите -classpath/a*<JDBC_Driver_File_Path>* в первой и единственной строке этого файла*JDBC_Driver_File_Path* - это полный путь к файлу JDBC-драйвера базы данных (или имя этого файла, если он находится в директории инсталляции AtomMind Server).

8. Запустите утилиту Конфигуратор сервера 177 или AtomMind Server и запустите действие Конфигурировать сервер 1559. Перейдите во вкладку **База Данных** в диалоговом окне конфигурации сервера.

9. Измените настройку **Драйвер БД** в Настройках общей конфигурации 182 на имя класса Java, представляющего драйвер БД. Это имя определяется производителем драйвера JDBC. Например, класс драйвера БД MySQL называется "com.MySQL.jdbc.Driver". Дополнительную информацию о Общих Настройках Конфигурации можно найти здесь 179.

10. В Настройках Общей Конфигурации укажите **URL** Вашей **базы данных**. Ее формат также задан производителем JDBC-драйвера. Для MySQL формат имеет следующий вид: jdbc:MySQL://[host][:port]/[database], где [host] - IP-адрес или DNS-имя сервера MySQL (может быть также пустой строкой или localhost), [port] - номер порта, на котором запущен сервер MySQL (опционально - если он не задан, используется порт по умолчанию), и [database], являющееся именем базы данных AtomMind Server'a. В качестве имени БД Вы можете использовать "AtomMind Server". Например, если Ваш сервер MySQL запущен на 192.168.0.1 с портом по умолчанию, используйте следующий URL: jdbc:MySQL://192.168.0.1/AtomMind Server.

11. Задайте правильное **имя пользователя** и **пароль к БД** в Общих Настройках Конфигурации. Они нужны для входа в базу, обозначенную в настройках **URL Базы Данных**.

12. Выберите **диалект базы данных** в Общих Настройках Конфигурации, соответствующий типу Вашей БД. См. общие настройки конфигурации для Диалекта БД 184.

13. Сохраните новые настройки. Не перезапускайте сервер, даже если появится соответствующее приглашение.

14. Остановите конфигуратор сервера (или AtomMind Server).

15. Запустите утилиту конвертер базы данных 706, которая находится в папке инсталляции AtomMind Server, чтобы переместить данные из старой базы в новую.

16. Дождитесь, когда конвертер закончит работу. В зависимости от размера БД на это может уйти от нескольких секунд до нескольких часов.

17. Запустите AtomMind Server.

18. Процесс закончен, и теперь Вы можете использовать новую БД.

Максимальное количество параллельных соединений для Вашего сервера БД должно быть больше значения Максимального размера пула подключений 185 в Общих Настройках AtomMind Servera. В большинстве случаев количество параллельных соединений БД по умолчанию меньше размера пула подключений AtomMind Server, поэтому должно быть увеличено. О том, как увеличить лимит соединений, прочитайте в документации к Вашему серверу БД.

11.1.3.1.1 Переключение базы данных на MySQL

Некоторые версии дистрибутива AtomMind Server содержат пакет MySQL сервера. Если этот пакет был выбран для установки, сервер будет автоматически использовать базу данных MySQL, и выполнять ниже описанные этапы Вам не нужно.

Если пакет MySQL был установлен, то его файлы находятся в подпапке mysql/ инсталляционной директории AtomMind Server. База данных находится в подпапке mysql/data/.

Инсталлятор настраивает MySQL для автозапуска во время загрузки ОС. Конфигурации AtomMind Server и MySQL по умолчанию оптимизируются для получения максимальной производительности.

Имейте в виду,что пакет MySQL предварительно оптимизирован для производительности и требования к памяти у него довольно высокие. Вам, возможно, потребуется перенастроить инсталляцию MySQL, если ваш сервер имеет менее 4 Гб оперативной памяти.

Чтобы переключиться на использование MySQL в качестве сервера баз данных, следуйте общим инструкциям из параграфа Переключение на другой движок базы данных 698. Ниже приводятся специфичные для этого SQL-сервера шаги.

1. JDBC-драйвер MySQL (**MySQL-connector** для **Java**) уже включен в дистрибутив AtomMind Server. Файл драйвера называется `MySQL-connector-java-5.X.XX-ga-bin.jar`, он находится в подпапке `/lib` в папке для инсталляции AtomMind Server. Чтобы добиться лучшей производительности и совместимости с поздними версиями MySQL, можно обновить JDBC-драйвер MySQL до самой последней версии. На момент написания документации, эта версия доступна на http://www.mysql.com/products/connector/j/.
2. В Настройках Общей Конфигурации установите значение переменной AtomMind Server'a `Database Driver` в значение `com.mysql.jdbc.Driver`. Это имя Java-класса для драйвера MySQL.
3. Формат переменной `Database URL` для MySQL имеет следующий вид: `jdbc:mysql://[host][:port][/database]`, где `host` - IP-адрес или DNS-имя сервера MySQL (может быть пустая строка или `localhost`), `port` - это номер порта, на котором запущен сервер MySQL (пропустите эту часть URL чтобы использовать значение по умолчанию), `database` - имя базы данных, содержащей данные AtomMind Server'a. В качестве имени базы данных можно использовать `server`. Например, если Ваш MySQLSQL запущен на машине 192.168.0.1 с портом по умолчанию, используйте следующую строку в качестве URL: `jdbc:mysql://192.168.0.1/server`.
4. Установите значение переменной `Database Dialect` в MySQL 5 (`MySQL5InnoDBDialect`), если Вы используете MySQL 5 или MySQL (`MySQLInnoDBDialect`) для старых версий.

Если вы хотите использовать символы не из Юникода, необходимо указать параметр characterEncoding. Это можно сделать двумя способами:

- Через URL базы данных: jdbc:mysql://localhost:3306/test?useUnicode=true&characterEncoding=UTF-8
- Через Дополнительные свойства: useUnicode = true, characterEncoding = UTF-8

Если вы используете MySQL 8, необходимо указать следующий параметр: `useSSL=false` в URL вашей базы данных или через Дополнительные свойства.

Установка MySQL

Для использования MySQL с AtomMind требуется сделать несколько изменений в настройках MySQL. Обычно эти настройки производятся изменением конфигурационного файла MySQL (по умолчанию `my.ini`).

1. Добавьте параметр `max_allowed_packet=100M`, чтобы разрешить большие запросы.

2. Установите значение параметра `max_connections` больше, чем значение общих настроек переменной AtomMind Server'a Минимальный размер пула соединений 185 (например, `max_connections=1000`).

3. Установите значение параметра `innodb_buffer_pool_size` в максимальное значение размера RAM, которое сервер MySQL может использовать. В случае выделенного сервера баз данных это значение стоит поставить в 80% от всей доступной в системе памяти.

4. Установите значение параметра `innodb_log_file_size` в 25-50% от `innodb_buffer_pool_size`.

5. Добавьте строки `character-set-server = utf8` и `collation-server = utf8_unicode_ci` к конфигурации.

КОНФИГУРАЦИОННЫЙ ФАЙЛ MYSQL

Ниже приведен пример конфигурационного файла MySQL 5.7, подходящего для больших инсталляций AtomMind.

```
[mysqld]
port  = 3306
# Вы можете использовать настройки сокета из созданного автоматически файла конфигурации.
socket =/tmp/mysql.sock
default-storage-engine = INNODB
datadir =/data/mysql
```

```
max_connections = 1000
max_allowed_packet = 100M

innodb_flush_log_at_trx_commit = 0
innodb_buffer_pool_size =2G
innodb_log_file_size =500M
innodb_log_buffer_size = 8M
innodb_lock_wait_timeout = 1200
innodb_file_per_table = 1

sort_buffer_size = 20M
query_cache_size = 100M

table_open_cache = 10000
table_definition_cache = 10000

thread_cache_size = 32
innodb_thread_concurrency = 0

character-set-server = utf8
collation-server = utf8_unicode_ci
```

Необходимо изменить */data/mysql* на подлинный каталог данных для вашей инсталляции сервера MySQL. В Linux путь каталога файлов MySQL /var/lib/mysql.

Изменение рамеров буфера

Инсталляция MySQL в пакете с AtomMind Server для Windows конфигурируется для серверов высокой производительности. Если сервер установлен на низкопроизводителном устройстве (например, на ноутбуке), сервер MySQL может не запуститься из-за проблемы выделенной буферной памяти. В этом случае необходимо уменьшить размеры буфера:

- Проверьте, что `MySQL для AtomMind` сервиса/процесса не запущен на сетевом компьютере или остановите его, если он запущен (вы можете контролировать сервис из `Control Panel > Administration > Services`)
- Поправьте файл `mysql/my.ini`, расположенный в дистрибутивной папке AtomMind Server'a
- Уменьшите `innodb_buffer_pool_size`, например, установите его на `500M`
- Уменьшите `innodb_log_file_size` на 25-50% от `innodb_buffer_pool_size`, например, установите его на `200M`
- Удалите файл регистрации MySQL (`mysql/ib_logfile0` и `mysql/ib_logfile1`)
- Снова запустите `MySQL для сервиса AtomMind`

11.1.3.1.2 Переключение базы данных на Microsoft SQL Server

Чтобы переключиться на использование Microsoft SQL Server в качестве сервера баз данных, следуйте общим инструкциям из параграфа Переключение на другой движок базы данных 698. Ниже приводятся специфичные для этого SQL-сервера шаги.

1. Скачайте и установите последнюю версию **JDBC-**драйвера для **Microsoft SQL Server**. В момент написания этого документа драйвер можно взять на странице http://msdn.microsoft.com/data/jdbc/.
2. Скопируйте файл `sqljdbc4.jar` в подпапку `/jar` установочной папки AtomMind Server.

3. В Настройках Общей Конфигурации установите значение переменной AtomMind Server'a `Database Driver` на `com.microsoft.sqlserver.jdbc.SQLServerDriver`. Это имя Java-класса для драйвера Microsoft SQL Server.
4. Формат переменной `Database URL` для Microsoft SQL Server имеет следующий вид: `jdbc:sqlserver://[serverName[\instanceName][:port]][;property=value[;property=value]]`, где `jdbc:sqlserver://` является константой и описывает подпротокол, `serverName` - DNS-имя или IP-адрес сервера (может быть `locahost`), `instanceName` - имя экземпляра сервера на удаленной машине (если не указано, то используется экземпляр по умолчанию), `portNumber` - номер порта для подключения (1433 по умолчанию). Например, если Ваш Microsoft SQL Server запущен на 192.168.0.1, на порту по умолчанию, используйте следующий URL для подключения к экземпляру по умолчанию: `jdbc:sqlserver://192.168.0.1:1433`. Можно также пропустить порт по умолчанию и использовать в качестве URL такую строку: `jdbc:sqlserver://192.168.0.1`.
5. Установите значение переменной `Database Dialect` в `SQLServerDialect`.

Настройка собственной аутентификации

В некоторых случаях вам следует использовать Собственную аутентификацию. Следуйте инструкциям для настройки:

1. Загрузите и установите последнюю версию библиотеки `sqljdbc_auth.dll` с сайта Microsoft.
2. Поместите `sqljdbc_auth.dll` в папку, путь которой определен опцией `java.library.path` JVM.
3. Используйте предыдущий пример URL, просто добавив параметр `IntegratedSecurity`: `IntegratedSecurity=true`.

Пример: `jdbc:sqlserver://192.168.0.1:1433;IntegratedSecurity=true`

Настройка аутентификации домена

Драйвер JDBC сервера Microsoft SQL не позволяет использовать аутентификацию доменов. Для того, чтобы редактировать подключения сервера Microsoft SQL, используя ваши доменные регистрационные данные, следуйте следующим инструкциям:

1. Загрузите последнюю версию **jTDS JDBC Driver**. В процессе написания, она доступна на сайте http://jtds.sourceforge.net/.
2. Поместите `jtds-X.X.X.jar` в подпапку `/jar` папки установки AtomMind Server.
3. Установите `Database Driver` в Настройки общей конфигурации AtomMind Server на `net.sourceforge.jtds.jdbc.Driver`.
4. Настройка `Database URL` имеет следующий формат: `jdbc:jtds:sqlserver://[serverName[\instanceName][:port]];domain=domainValue;user=userValue;password=passwordValue[;property=value[;property=value]]`, где часть `jdbc:jtds:sqlserver://` известна как подпротокол и является постоянной, `serverName` является именем DNS или IP-адресом сервера для подключения (может быть `locahost`), `instanceName` является экземпляром для подключения к `serverName` (если не определен, совершается подключение к экземпляру по умолчанию), `portNumber` является портом для подключения к `serverName` (По умолчанию – 1433).

Пример: `jdbc:jtds:sqlserver://192.168.0.1:1433;domain=COM;user=admin;password=pass`

Для успешной аутентификации с использованием jTDS драйвера, AtomMind Server должен работать на Windows-машине, включенной в домен.

Настройка сервера Microsoft SQL

Для поддержания производительности сервера Microsoft SQL на определенном уровне необходимо:

- Определить фактор заполнения для индексов
- Настроить сжатие таблицы и индекса

Рекомендуется определить значение фактора заполнения – 70. Для этого откройте Microsoft SQL Server Management Studio и:

1. Нажмите на подключение к базе данных.

2. Выберите **Свойства** в конце списка.

3. Нажмите на страницу **Настройки Базы данных** слева под **Свойства сервера**.

4. Укажите 70 в качестве значения **фактора заполнения индекса по умолчанию**.

Необходимо сделать это перед первым запуском AtomMind Server, потому что конфигурация, описанная выше, применяется только к новым индексам (т.е. только что созданным таблицам).

Однако, есть возможность изменения опции фактора заполнения для уже созданных индексов. Сделать это можно несколькими способами:

http://technet.microsoft.com/en-us/library/ms177459(v=sql.110).aspx

Все советы производят один и тот же эффект – реорганизацию индексов с новым значением фактора заполнения.

Фактор заполнения помогает уменьшать коэффициент фрагментации индекса. Но при этом необходимо отслеживать уровень фрагментации и регулярно реорганизовывать/перестраивать индексы.

Используйте следующий скрипт для поддержания фрагментации индекса на нужном уровне. Например, создайте Агент сервера Microsoft SQL для запуска скрипта планировщиком.

```
USE ag;
GO

SET NOCOUNT ON;
DECLARE @objectid int;
DECLARE @indexid int;
DECLARE @partitioncount bigint;
DECLARE @schemaname nvarchar(130);
DECLARE @objectname nvarchar(130);
DECLARE @indexname nvarchar(130);
DECLARE @partitionnum bigint;
DECLARE @partitions bigint;
DECLARE @frag float;
DECLARE @command nvarchar(4000);
DECLARE @dbid smallint;

-- Choose indexes with fragmentation level more than 10%
-- Define current database

SET @dbid = DB_ID();
SELECT
    [object_id] AS objectid,
    index_id AS indexid,
    partition_number AS partitionnum,
    avg_fragmentation_in_percent AS frag, page_count
INTO #work_to_do
FROM sys.dm_db_index_physical_stats (@dbid, NULL, NULL , NULL, N'LIMITED')
WHERE avg_fragmentation_in_percent > 10.0
AND index_id > 0 -- игнорируем heap

-- Declare a cursor for a processing partitions' list.

DECLARE partitions CURSOR FOR SELECT objectid,indexid, partitionnum,frag FROM #work_to_do;

OPEN partitions;

-- Loop by partition
WHILE (1=1)
BEGIN
FETCH NEXT
FROM partitions
INTO @objectid, @indexid, @partitionnum, @frag;
IF @@FETCH_STATUS < 0 BREAK;
SELECT @objectname = QUOTENAME(o.name), @schemaname = QUOTENAME(s.name)
FROM sys.objects AS o
JOIN sys.schemas AS s ON s.schema_id = o.schema_id
```

```
WHERE o.object_id = @objectid;

SELECT @indexname = QUOTENAME(name)
FROM sys.indexes
WHERE object_id = @objectid AND index_id = @indexid;
SELECT @partitioncount = count (*)
FROM sys.partitions
WHERE object_id = @objectid AND index_id = @indexid;

-- 30% is limit for index rebuilding
IF @frag < 30.0
    SET @command = N'ALTER INDEX ' + @indexname + N' ON ' + @schemaname + N'.' + @objectname + N'
IF @frag >= 30.0
    SET @command = N'ALTER INDEX ' + @indexname + N' ON ' + @schemaname + N'.' + @objectname + N'
IF @partitioncount > 1
    SET @command = @command + N' PARTITION=' + CAST(@partitionnum AS nvarchar(10));

EXEC (@command);
PRINT N'Выполнено: ' + @command;
END;

CLOSE partitions;
DEALLOCATE partitions;

-- delete temporary table
DROP TABLE #work_to_do;
GO
```

Сжатие таблицы и индекса позволяет уменьшить количество логических прочтений.

Также рекомендуется включить сжатие страницы для таблицы `ag_properties`. Для дополнительной информации смотрите http://msdn.microsoft.com/en-us/library/hh710070.aspx.

11.1.3.1.3 Переключение базы данных на Oracle

Чтобы переключиться на использование Oracle в качестве сервера баз данных AtomMind Server'a, следуйте общим инструкциям из параграфа Переключение на другой движок базы данных 698. Ниже приводятся специфичные для этого SQL-сервера шаги.

1. Скачайте версию **JDBC-**драйвера из Java JDK 1.6 для **Oracle**. В момент написания этой статьи драйвер доступен для скачивания на странице http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/.
2. Скопируйте файл `ojdbc.jar` в папку `/lib` установочной папки AtomMind Server'a.
3. В Настройках Общей Конфигурации установите значение переменной AtomMind Server'a `Database Driver` в значение `oracle.jdbc.OracleDriver`. Это имя Java-класса для драйвера Oracle.
4. Формат переменной `Database URL` для Oracle имеет следующий вид: `jdbc:oracle:thin:@host[:port]:database`, где `host` - IP-адрес или DNS-имя сервера Oracle (может быть пустым или `localhost`), `port` - порт, на котором запущен сервер (пропустите эту часть URL чтобы использовать значение по умолчанию), и `database` в качестве имени базы данных, содержащей данные AtomMind Server'a. В качестве имени базы данных можно использовать `linkserver`. Например, если Ваш сервер Oracle запущен на машине 192.168.0.1 на порту по умолчанию, используйте следующую строку в качестве URL: `jdbc:oracle:thin://192.168.0.1:linkserver`.
5. Установите значение переменной `Database Dialect` в `Oracle 10g/11g`.

Настройка Oracle

AtomMind требует внесения изменений в конфигурацию Oracle. Данные изменения обычно применяются на консоли Oracle.

```
alter system set processes=<process_count>   scope=spfile;
alter system set sessions=<session_count  > scope=spfile;
alter system set transactions=<transaction_count  > scope=spfile;
```

Установите `process_count` в значение выше значения Максимального размера пула соединений 185 в общих настройках конфигурации AtomMind Servera , напр. **Maximum Connection Pool Size * 1.05**.

Установите `session_count` в **(1.1 * process_count) + 5**.

Установите `transaction_count` в **1.1 * session_count**.

Максимальная длина ID (имя таблицы или колонок) в БД Oracle - это 30 символов.

В Oracle есть дескриптор "cluster". Пожалуйста, не используйте его для наименования класса или столбца внутри классов 885 AtomMind.

Плагин CMDB несовместим с базой данных Oracle. Пожалуйста, удалите его из AtomMind\плагинов\контекста\папки перед стартом сервера.

11.1.3.1.4 Переключение базы данных на PostgreSQL

Чтобы переключиться на использование PostgreSQL в качестве сервера баз данных AtomMind Server'a, следуйте общим инструкциям из параграфа Переключение на другой движок базы данных 698. Ниже приводятся специфичные для этого SQL-сервера шаги.

1. Скачайте последнюю версию **JDBC4**-драйвера для **PostgreSQL**. В момент написания этой статьи он доступен на странице http://jdbc.postgresql.org/.

2. Скопируйте файл `postgresql-X.X-XXX.jdbc4.jar` в подпапку `/jar` инсталляционной папки AtomMind Server.

3. В Настройках Общей Конфигурации установите значение переменной `Database Driver` в значение `org.postgresql.Driver`. Это имя Java-класса для драйвера PostgreSQL.

4. Формат переменной `Database URL` для PostgreSQL имеет следующий вид: `jdbc:postgresql://host[:port]/database`, где `host` - IP-адрес или DNS-имя сервера, на котором запущен PostgreSQL (может быть пустой строкой или `localhost`), `port` - номер порта (опустите эту часть URL чтобы использовать значение по умолчанию), `database` - имя базы данных, содержащей данные AtomMind Server'a. В качестве имени базы данных можно использовать `linkserver`. Например, если Ваш PostgreSQL запущен на машине 192.168.0.1 с портом по умолчанию, используйте следующую строку в качестве URL: `jdbc:postgresql://192.168.0.1/linkserver`.

5. Установите значение переменной `Database Dialect` в `PostgreSQLDialect`.

11.1.3.1.5 Переключение базы данных на Firebird

Чтобы переключиться на использование Firebird в качестве сервера баз данных AtomMind Server'a, следуйте общим инструкциям из параграфа Переключение на другой движок базы данных 698. Ниже приводятся специфичные для этого SQL-сервера шаги.

1. Скачайте драйвер **Jaybird JCA/JDBC**. В момент написания этой статьи он доступен на странице http://www.firebirdsql.org/index.php?op=devel&sub=jdbc.

2. Скопируйте файл `jaybird-full-X.X.X.jar` в подпапку `/lib` инсталляционной папки AtomMind Server'a.

3. В Общих Настройках Конфигурации установите значение переменной `Database Driver` в значение `org.firebirdsql.jdbc.FBDriver`. Это имя Java-класса для драйвера Firebird.

4. Формат переменной `Database URL` для Firebird имеет следующий вид: `jdbc:firebirdsql:host[/port]:/path/to/db.fdb`, где `host` - IP-адрес или DNS-имя сервера, на котором запущен Firebird (может быть пустой строкой или `localhost`), `port` - номер порта (опустите эту часть URL чтобы использовать значение по умолчанию), `/path/to/db.fdb` - путь к базе данных, содержащей данные AtomMind Server'a. В качестве имени базы данных можно использовать `linkserver`. Например, если Ваш Firebird запущен на localhost с портом по 3050 и базой данных в C:\db\linkserver.fdb, используйте следующую строку в качестве URL: `jdbc:firebirdsql:localhost/3050:C:\\db\\linkserver.fdb`.

5. Установите значение переменной `Database Dialect` в `FirebirdDialect`.

6. Установите значение переменной `Batch Size` в нуль. Это необходимо, поскольку JDBC-драйвер Firebird не поддерживает пакетные вставки объектов, содержащих BLOB-поля.

11.1.3.2 Конвертор базы данных

Утилита конвертора базы данных (db_converter) находится в инсталляционной папке AtomMind Server. Она используется для переноса данных из базы данных 692 AtomMind Servera в другую.

Файл с расширением db_converter принимает два аргумента: путь старого конфигурационного файла AtomMind Server (содержащий конфигурацию старой базы данных) и путь нового конфигурационного файла. Итак, команда на исполнение выглядит следующим образом: db_converter server_old.xml server.xml.

Конвертирование базы данных может занять значительное время.

11.1.3.3 Схема и взаимодействие базы данных

Схема базы данных AtomMind Server полностью динамическая, новые таблицы оперативно создаются и удаляются сервером, таким образом отражая создание, удаление и обновление системных ресурсов.

Тем не менее, в каждой инсталляции существует несколько основных таблиц:

Таблица	Описание
ag_properties	Содержит значения всех постоянных переменных контекста 61 сервера. AtomMind Server генерирует в этой таблице относительно малое количество запросов ВСТАВИТЬ и УДАЛИТЬ, когда создаются или разрушаются системные объекты. Количество запросов ВЫБРАТЬ также относительно мало из-за серверного кэширования памяти. При этом таблица ag_properties часто ОБНОВЛЯЕТСЯ, потому что значения постоянных свойств ресурса меняются, и сервер сохраняет измененные значения в базе данных. Например, устройства 497, использующие постоянное значение кэша 502, вызывают большое количество обновлений ag_properties в период синхронизации 514. Поля: • **контекст** - полный путь контекста 41 значений этих переменных содержится в строке таблицы • **свойство** - имя переменной • **данные** - таблица данных 49, представляющая значение переменной, закодированной в виде строки 2123 с использованием невидимых разделителей
ag_events	Содержит все события контекста 73 сервера, кроме тех, которые хранятся в других (настраиваемых) таблицах. Внизу приведен подробный список **Событий Контекста в Памяти**. Поля: • **контекст** - полный путь контекста 41, в котором было запущено событие • **имя** - имя события (т.е. тип события) • **время создания** - временная отметка появления события (или последнее появление события, если было зарегестрировано множество его дубликатов) • **срок действия** - временная отметка запланированного срока действия события • **уровень** - уровень события • **права доступа** - пользовательские права, необходимые для доступа к экземпляру события, или ноль, если должны использоваться права, обозначенные в определении события • **число** - количество зарегистрированных дубликатов 196 события • **подтверждение** - закодированный список подтверждений 76 появления события • **обогащение** - закодированный список обогащений 76 события • **формат** - закодированный формат события или ноль, если должен использоваться формат, обозначенный в определении события • **данные** - таблица данных 49, представляющая данные события 74, закодированные в виде строки 2123 с использованием невидимых разделителей
ag_data	Содержит двоичные блоки данных 52, встроенные в значения переменных контекста 61. Эти блоки извлекаются в отдельную таблицу, когда значение переменной сохраняется в таблице ag_properties. Однако двоичные блоки не загружаются сразу после того, как загружается

	значение переменной из ag_properties. Блоки загружаются только по запросу, т.е. при помощи явного запроса модулей AtomMind Servera, клиентов внешних приложений.

Память Событий Контекста

Большая инсталляция AtomMind может хранить миллиарды событий контекста 73 в базе данных. Эти события динамически распределяются между различными таблицами, чтобы оптимизировать память и производительность запроса.

Формат всех таблиц событий соответствует формату таблицы ag_events, описанной выше.

Доступ к таблицам событий обычно состоит из множества операций ВСТАВИТЬ. Новая запись вставляется каждый раз, когда регистрируется постоянное событие сервера. Сообщение УДАЛЕНО передается поверх таблицы событий в двух случаях:

- Если системный ресурс со всеми соответствующими событиями удален
- Если просроченные события стираются периодической задачей-планировщиком 823

Количество запросов ВЫБРАТЬ в таблицах событий относительно мало. События загружаются в следующих случаях:

- Если Журнал Событий 398 открыт, вызывая последующую активацию фильтра событий 762
- Если события напрямую загружаются при помощи функции получить () 1516 контекста **Событий**
- Если история переменных загружается при помощи функции ИсторияПеременных() 1613 контекста **Утилитов**
- В период начальной визуализации диаграммы виджетов 1051, которая сконфигурирована так, чтобы включать в себя хронологические события или значения переменных
- В нескольких других подобных случаях

Несмотря на то, что события загружаются редко, количество загружаемых за один запрос событий не ограничено платформой, поскольку определенные инсталляции требуют миллионов событий, загружаемых за раз. Архитекторы системы, разрабатывающие решения на основе AtomMind, должны помнить о максимальном количестве загружаемых событий при выстраивании серверных цепочек обработки данных и визуализирующих панелей инструментов.

Попытка загрузить слишком большое количество событий за раз вызовет у AtomMind Servera исчерпывание памяти в виде значительного снижения производительности, вызванного сборкой мусора Виртуальной Машины Java, и может привести к полному зависанию сервера или внутренним ошибкам памяти.

Список таблиц событий, представленный в этой документации, не полон. Различные модули и плагины AtomMind Servera могут запросить создание других выделенных таблиц событий, которые здесь не упомянуты.

Все события, которые не были явно сконфигурированы для использования настраиваемой таблицы, записываются в таблице ag_events. Приведенный ниже список описывает таблицы базы данных, которые обычно создаются для хранения событий различных типов.

Таблица	Описание
ag_info	Содержит события категории Инфо 77.
ag_alerts	Содержит события категории Тревоги 790.
ag_*xxx*_change	Содержит все Изменения 84 событий, представляя изменения переменных в одном контексте устройства 1494 (соответствующие устройству *xxx*). Обратите внимание, что имя таблицы может быть усечено из-за ограничения длины имени таблицы, отражающей базу данных всего сервера.

Прямой доступ к базе данных AtomMind Servera

Прямой доступ сторонних приложений к базе данных AtomMind Servera не приветствуется. AtomMind - мощная платформа, предлагающая полноценный пакет разработки программ (SDK) и различные виды программных интерфейсов приложения (API) для локального и удаленного доступа ко всем элементам данных, содержащимся в базе данных сервера. При этом доступ на основе программных интерфейсов приложения будет всегда наследовать надлежащие блокировки, проверку прав доступа 477 и оптимизацию производительности.

Прямая модификация данных, содержащихся в базе данных AtomMind Servera, в большинстве случае вызовет некорректное поведение системы. Тем не менее, прямое чтение (ВЫБРАТЬ) операций можно использовать в редких случаях.

Пожалуйста, свяжитесь с ТВЭЛом для получения информации о том, как избежать прямого доступа к базе данных.

11.1.3.4 Настройка производительности базы данных

Максимальная производительность базы данных AtomMind Servera - ключевой фактор высокой производительности всей инсталляции AtomMind. Вот краткий список советов по оптимизации общей производительности базы данных:

1. AtomMind Server использует множество запросов ВСТАВИТЬ и ОБНОВИТЬ, поэтому следуйте советам производителя базы данных по повышению производительности запросов ВСТАВИТЬ/ОБНОВИТЬ.
2. Количество разрешенных параллельных соединений должно быть больше настройки общей конфигурации Максимальный Размер Пула Соединений 185.
3. Когда значения переменных 61, содержащие большие двоичные блоки (звуки, изображения, файлы прошивок), хранятся в базе данных, AtomMind Server сохраняет их, используя одну транзакцию ВСТАВИТЬ/ОБНОВИТЬ. Таким образом, максимальный размер данных, переводимых за одну транзакцию, будет не ограничен или ограничен достаточно большим значением (рекомендуем ограничение в 100 Мб).
4. Для больших инсталляций мы рекомендуем отключение синхронного сброса данных ВСТАВИТЬ и ОБНОВИТЬ на диск при помощи подтверждения транзакции. Это принесет значительное повышение производительности.
5. Если сервер базы данных запущен на том же устройстве, что и AtomMind Server, рекомендуем позволить ему использование 25-40% оперативной памяти, оставив остальную часть AtomMindy. Также избегайте замен.

Настройка производительности пулов соединений базы данных

Дополнительный раздел определяет, как AtomMind Server управляет пулом соединений базы данных.

AtomMind Server использует библиотеку пулов соединений Java под названием c3p0, чтобы управлять соединениями базы данных. Настройки по умолчанию библиотеки **c3p0** могут быть переопределены добавлением новых значений к файлу `hibernate.properties`, расположенному в установочной папке AtomMind Servera.

Вот пример содержимого файла `hibernate.properties`:

```
# Удостоверьтесь в правильном подходе к решению пролемы базы данных
hibernate.c3p0.checkoutTimeout=30000

# Слишком большое время простоя вызывает ошибки из-за разъединения с базой данных
hibernate.c3p0.maxIdleTime=300
```

Настройки конфигурации библиотеки **c3p0** описаны здесь: https://www.mchange.com/projects/c3p0/. Все свойства c3p0 должны иметь префикс "`hibernate.`".

РЕГУЛИРОВКА РАЗМЕРА ПУЛА СОЕДИНЕНИЯ С БД

В настройках БД 182 AtomMind Server есть две опции, которые в значительной степени воздействуют на производительность системы:

- **Минимальный размер пула соединений**
- **Максимальный размер пула соединений**

Пул соединений - это кэш соединений БД, поддерживаемый базой данных так, чтобы соединения можно было повторно использовать, когда БД получит очередные запросы данных. Пулы соединений используются для увеличения производительности выполняемых команд на БД.

Существует несколько правил для регулировки размера пула:

- Максимальный размер пула соединений должен быть больше количества одновременно подключенных устройств с высокой скоростью передачи данных (т.е. частые синхронизации или большое количество генерируемых устройством событий). Например, если отслеживаются 2000 устройств, которые опрашиваются

один раз в час, можно назначить для максимального размера пула значение, равное 200, но если у Вас 500 устройств, синхронизируемых с сервером каждые 5 минут, лучше увеличить максимальный размер пула до 500.

- Максимальный размер пула должен быть ниже общего количества соединений, разрешенных БД для того, чтобы избежать генерируемых базой данных ошибок, таких как "слишком много соединений".
- Для крупных инсталляций, где *пиковая* активность значительно выше *средней* активности, рекомендуется увеличить минимальный размер пула до 20-50% от максимального размера пула.

Очистка базы данных

База данных AtomMind Server содержит преимущественно конфигурации и события. В то время как обычно объем данных конфигурации небольшой, и не увеличивается с течением времени, число событий (включая события изменения значений) может увеличиваться достаточно быстро, в следствии чего размер базы данных может составлять несколько гигабайтов.

Корректировка периодов хранения событий окажет влияние с течением времени, поэтому в некоторых случаях очистка базы данных вручную имеет смысл.

Чтобы установить объем и структуру базы данных, запустите действие Посмотреть статистику базы данных 1559 из корневого контекста. Это действие покажет **Статистику события**, т.е. количество событий в каждой таблице контейнеров событий и их распределение по типам и контекстам 41 источника, а также общее число событий в каждой таблице.

Основная концепция очистки базы данных заключается в следующем:

Очистка любой таблицы базы данных AtomMind Server, которая содержит события в то время как сервер не работает, не помешает нормальной работе сервера.

Это делает возможным очистку почти любой таблицы в базе данных AtomMind Server кроме **ag_properties и других** "специальных" таблиц, описанных в разделе Схема и взаимодействие базы данных 706. Настоятельно рекомендуем остановить сервер и сделать резервную копию базы данных перед очисткой любой таблицы событий. Еще одной опцией являются таблицы очистки с помощью действия Выполнить прямой запрос к СУБД 1548 без остановки сервера, т.е. применив `DELETE * FROM ag_xxx_change` или похожий запрос обновления.

ОЧИСТКА POSTGRESQL

Рекомендуемый цикл очистки для БД PostgreSQL DB - это выполнение *vacuumlo()* каждый час и *vacuumdb()* каждый день. Помогает сохранить оптимальный размер БД PostgreSQL:

```
#!/bin/bash
export PATH=$PATH:/usr/pgsql-9.4/bin/ export PATH=$PATH:/usr/pgsql-9.4/bin/
TIME=`date +'%Y%m%d-%H%M'`
echo $TIME" executing vacuumlo"
vacuumlo -U postgres -v -w password
TIME=`date +'%Y%m%d-%H%M'`
echo $TIME" stop vacuumlo"
```

```
#!/bin/bash
export PATH=$PATH:/usr/pgsql-9.4/bin/ export PATH=$PATH:/usr/pgsql-9.4/bin/
TIME=`date +'%Y%m%d-%H%M'`
echo $TIME" executing vacuumdb"
vacuumdb -fav -w
TIME=`date +'%Y%m%d-%H%M'`
echo $TIME" stop vacuumdb"
```

ОЧИСТКА БЛОБА POSTGRESQL

Есть два варианта удаления устаревших блоб данных из базы данных.

Первый вариант - использовать комманду:

```
vacuumlo -WUSER DATABASE
```

Заметьте, что в процессе очистки показатели производительности значительно упадут.

Второй вариант применяется к системам 24/7. Необходимо создать триггерную функцию и набор триггеров для таблиц:

```
СОЗДАЙТЕ ИЛИ ПЕРЕМЕСТИТЕ ФУНКЦИЮ "BlobDel"()
  ВОЗВРАЩАЕТ trigger КАК
```

```
$BODY$
BEGIN
удалите из pg_catalog.pg_largeobject_metadata
где pg_catalog.pg_largeobject_metadata.oid = OLD.ag_data;
удалите из pg_catalog.pg_largeobject
где pg_catalog.pg_largeobject.loid = OLD.ag_data;
возвращает OLD;
END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION "BlobDel"()
  OWNER TO postgres;
```

Который будет активирован триггером:

```
СОЗДАЙТЕ ТРИГГЕР ag_info
  ПЕРЕД УДАЛЕНИЕМ
  ИЗ ag_info
  ДЛЯ КАЖДОГО РЯДА
  ВЫПОЛНИТЕ ПРОЦЕДУРУ "BlobDel"();
```

Эта триггерная функция удалит все потерянные блобы из таблиц **pg.largeobject_metadata** и **pg.largeobject** после операции очистки в таблице **ag_info**. Есть возможность добавления таких триггеров как этот в другие таблицы вашей базы данных.

11.1.3.5 Отказоустойчивость базы данных

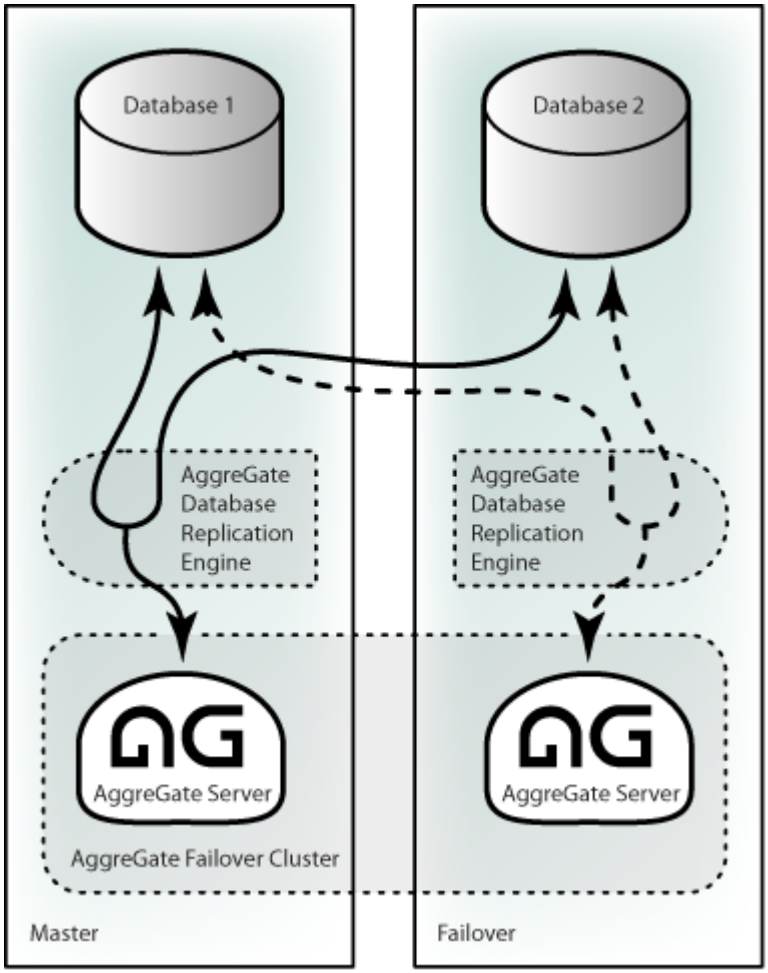
Главный и дублирующие серверы 1326 AtomMind связаны с одной и той же базой данных 692, которая используется для хранения данных о конфигурации и событиях в системе. Эту базу данных следует защищать от неполадок посредством реплицирования данных в несколько физических местоположений.

Существуют три способа настройки базы данных для отказоустойчивого кластера:

- С помощью репликационного движка AtomMind для БД 713
- "Родная" репликация базы данных 716
- Нереплицированная база данных 717 (**не рекомендуется**)

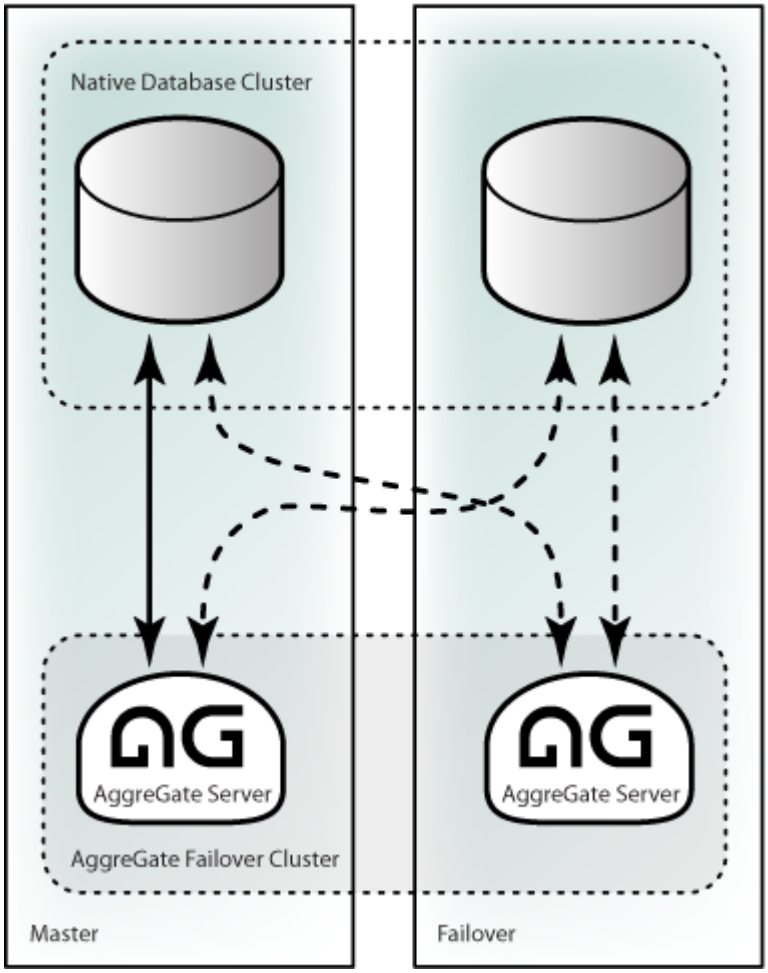
Репликационный движок БД средствами AtomMind

Обычный способ настройки репликации базы данных -- использование репликационного движка базы данных, интегрированного в AtomMind Server 713 для записи данных во множественные независимые базы данных и распределения нагрузки операций чтения.



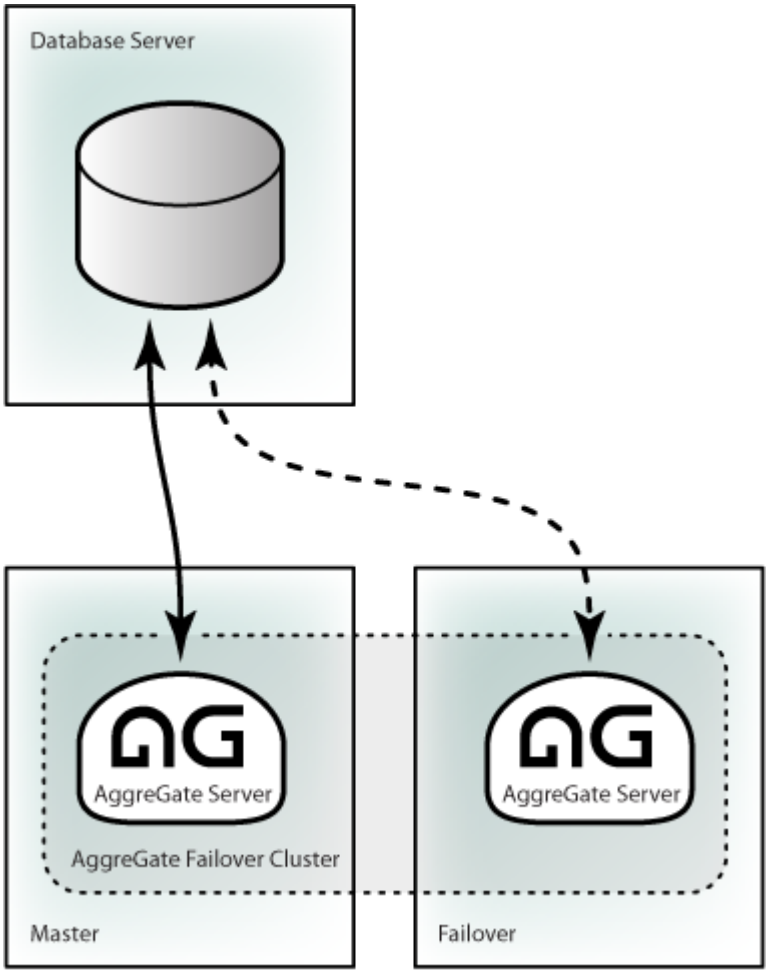
"Родная" репликация базы данных

Второй способ репликации базы данных AtomMind Server -- использование технологии "родной" репликации[716] сервера базы данных для построения отказоустойчивого кластера базы данных.



Нереплицированная база данных

Некоторые инсталляции AtomMind Server могут также быть связаны с одной и той же некластеризированной базой данных[717] как на Главном, так и на Дублирующих серверах. Однако этот метод не защищает всю систему от сбоев в работе сервера базы данных.



Предпочтительный метод осуществления отказоустойчивости базы данных

В большинстве случаев для кластеризации отказоустойчивой базы данных должен использоваться встроенный в AtomMind метод репликации данных 713. Его конфигурация столь же проста, как установка двух или более отдельных экземпляров серверов БД с редактированием нескольких опций общей конфигурации. Однако этот метод является гарантом целостности данных БД и оптимального функционирования.

11.1.3.5.1 Репликация базы данных средствами AtomMind

AtomMind Server имеет встроенный **кластерный движок высокой доступности**, который может выполнять репликацию базы данных и балансировку нагрузки 710, независимо от базы данных. Этот движок обычно имеет ту же конфигурацию, что главный и дублирующий сервера AtomMind. Он настроен на то, чтобы:

- Отслеживать соединения с несколькими серверами БД.
- Записывать любые данные, сгенерированные AtomMind Server во все БД.
- Выбирать менее загруженную базу данных для чтения данных, запрашиваемых AtomMind Server. Это гарантирует балансировку нагрузки между серверами БД.
- После восстановления неработающей БД движок синхронизирует данные между неисправной БД и другими (активными) базами данных.

Преимущества и недостатки

Преимущества:

- Настройка проста и не зависит от базы данных.
- Для обеспечения безопасности данные реплицируются во все узлы кластера БД.
- Балансировка нагрузки серверов БД во время операций чтения.

Недостатки:

- Если при записи в одну из БД кластера возникла ошибка, синхронизация баз данных осуществляется AtomMind Server, по сути останавливая все остальные операции сервера. Такие перерывы в работе могут занимать довольно много времени (несколько минут), если одна из баз данных кластера была недоступна длительное время (например, несколько часов или дней).
- Небольшие накладные расходы главного сервера AtomMind при операциях записи.
- Для всех узлов базы данных должны использоваться одни и те же данные для авторизации.

Создание кластера БД с использованием встроенной технологии AtomMind Server обеспечивает зеркалирование данных и, как результат, повышает надежность системы. Это также повышает производительность системы благодаря распределению нагрузки операций чтения. Однако, если узел кластера БД недоступен, AtomMind Server производит синхронизацию узла БД, чтобы обеспечить целостность данных. Эта операция по сути останавливает все другие операции сервера, поэтому использование встроенной кластеризации БД не рекомендовано для систем высокой надежности, т.е. систем, для которых несколько минут простоя недопустимы.

Пожалуйста, используйте технологию внутренней кластеризации ядра выбранной вами реляционной БД или технологию отказоустойчивой кластеризации БД NoSQL, встроенную в AtomMind Server , чтобы обеспечить сохранность данных при высокой надежности системы.

Настройка

Ниже приведен список изменений в конфигурации, необходимых для настройки репликации БД средствами AtomMind.

СЕРВЕРЫ БАЗЫ ДАННЫХ

- Настройка двух и более идентичных серверов БД.
- Настройка всех серверов на принятие соединений с IP-адресов или хостов главного и дублирующего серверов AtomMind.

ГЛАВНЫЙ СЕРВЕР ATOMMIND

- Запустите Конфигуратор сервера 177.
- Включите опцию Кластеризация базы данных 183 во вкладке **База данных**.
- Добавьте запись в таблицу Базы данных кластера 184 для каждого сервера БД. Убедитесь, что Вы указали правильный URL для каждой БД.
- Убедитесь, что **Имя пользователя БД**, **пароль**, и настройки **диалекта** во вкладке БД соответствуют Вашим БД.
- Сохраните общие настройки и запустите сервер. Теперь он должен соединиться с кластером БД.

Все базы данных в кластере должны использовать одни и те же имя пользователя/пароль для подключения AtomMind Server.

ОТКАЗОУСТОЙЧИВЫЙ СЕРВЕР ATOMMIND

- Выполните изменения, аналогичные изменениям на главном сервере, т.е. включите Кластеризацию БД и добавьте записи в таблицу баз данных кластера.

Решение ситуации сбоя в базе данных

Движок кластеризации выполняет периодические проверки доступности и целостности каждой БД. Если возникает неисправность базы данных, она отключается от кластера и помечается как **Not Active (**не активная**)**. Сервер не будет пытаться выполнить операции чтения/записи с этой БД.

Активация и синхронизация БД

Сервер периодически проверяет доступность всех неактивных БД. Если неактивная БД становится доступна, сервер автоматически реактивирует ее. В процессе реактивации происходит синхронизация данных между вновь

активированной базой и всеми остальными базами данных в кластере, что может потребовать большого количества времени и значительных ресурсов сервера.

Просмотр статуса кластера БД

Чтобы просмотреть статус отдельных баз данных, включенных в кластер базы данных, откройте таблицу Базы данных кластера 184 в общей настройке AtomMind Server. Поле **Активен** в этой таблице указывает, использует ли кластер базу данных в текущий момент.

Хранение информации о статусе баз данных кластера

AtomMind хранит в памяти информацию об активности баз данных кластера и удерживает эту информацию при перезапуске сервера. Данные хранятся независимо от ОС, например, в системном реестре (для систем Windows) или в файле (для систем Linux).

Чтобы конфигурационный файл кластера всегда соответствовал сохраненному в памяти статусу баз данных, удаляйте кластерные базы данных только при помощи утилиты конфигуратор сервера 177 или другими 177 способами конфигурации AtomMind Server. **Не** удаляйте учетные записи БД из конфигурационного файла кластера БД 715, редактируя этот файл напрямую.

11.1.3.5.1.1 Файл конфигурации баз данных кластера

Файл конфигурации кластеризованных баз данных называется `database.cluster.xml`. Он располагается в установочной директории AtomMind Server.

В этом файле определяется:

- Список баз данных, хранящих данные AtomMind Server
- Правила балансировки нагрузки серверов БД
- Стратегия синхронизации для восстановленных кластерных узлов БД

В большинстве случаев этот файл не должен редактироваться вручную. Используйте таблицу Баз данных кластера 184 через утилиту Конфигурация сервера 177, чтобы сделать изменения в настройках кластера базы данных.

Настройка файла кластеризации БД

Приведем список настроек, необходимых для установки репликации БД вручную:

- Установите атрибут `dialect` тега `<cluster>` согласно типу Вашей БД:

Тип БД	**Значение атрибута**
Apache Derby	derby
Firebird, InterBase	firebird
H2	h2
HSQLDB	hsqldb
IBM DB2	db2
Ingres	ingres
Mckoi	mckoi
MySQL	mysql
MySQL MaxDB	maxdb
Oracle	oracle
PostgreSQL	postgresql
Sybase	sybase
Standard (SQL-92 compliant)	standard

- Убедитесь, что число блоков <database/> соответствует числу баз данных в отказоустойчивом кластере. Удалите или добавьте блоки при необходимости.
- Присвойте каждой БД уникальный идентификатор, отредактировав атрибут id в каждом блоке <database/>.
- Задайте атрибут location в каждом блоке <database>. Возможно, URL будут отличаться во всех БД в кластере, так как URL обычно включает адрес сервера БД. Более подробно см. Адрес базы данных (URL) 183 и заметки, специфичные для БД 698.
- Установите правильное значение тэгов <username> и <password> для каждой БД.

Все базы данных кластера должны использовать одинаковый логин и пароль для подключения к ним AtomMind Server'a.

Если PostgreSQL 705 используется в качестве движка БД AtomMind Server, вам необходимо добавить следующие свойства к Адресу базы данных (URL) 182:

"tcpKeepAlive=true&networkTimeout=60&socketTimeout=60&loginTimeout=60&connectTimeout=60&cancelSignalTimeout=60".

Таймауты задаются в секундах.

11.1.3.5.2 Репликация средствами базы данных

Большинство БД уровня предприятий имеют "родную" поддержку для кластеризации и репликации данных. AtomMind Server может получить преимущество из кластеризации "родной" БД при автопереключении с главного на дублирующие узлы, когда сервер БД выполняет репликацию между узлами.

Следующие движки БД совместимы с технологией поддержки отказоустойчивой кластеризации AtomMind Server:

- Oracle
- MySQL
- Сервер Microsoft SQL
- PostgreSQL

Преимущества и недостатки

Преимущества:

- Для обеспечения надежности данные реплицируются во все узлы кластера БД.
- "Родная" репликация гарантирует быстрое и эффективное реплицирование между узлами.
- При временном сбое узла кластера БД последующая синхронизация данных между неисправным и остальными узлами выполняется движком БД с минимальными потерями.

Недостатки:

- Построение "родного" кластера БД может оказаться затруднительным.
- Драйвер JDBS AtomMind Server'a должен быть настроен на автопереключение на дублирующие сервера БД.

Настройка

Ниже следует список изменений в конфигурации, необходимых для настройки "родной" кластеризации БД.

СЕРВЕРЫ БАЗЫ ДАННЫХ

- Дополнительную информацию о том, как построить кластер, можно найти в документации к Вашему серверу БД.

ГЛАВНЫЙ СЕРВЕР ATOMMIND

- Если Ваш главный сервер еще не настроен на работу с кластеризированной БД, переключите 698 его на сервер базы данных.
- Дополнительную информацию о том, как настроить адреса для дублирующих серверов, можно найти в документации к Вашему JDBC драйверу. В большинстве случаев адреса дублирующих серверов указываются в URL базы данных 183.

ДУБЛИРУЮЩИЙ СЕРВЕР ATOMMIND

- Настраивайте БД аналогично конфигурации главного сервера.

11.1.3.5.3 Использование общей базы данных

И основная, и дублирующая инсталляция AtomMind Server могут делить между собой одну некластеризованную БД, которая может находится на:

- Отдельном компьютере
- Том же самом компьютере, на котором запущен главный сервер или один из отказоустойчивых серверов.

Достоинства и недостатки

Этот метод конфигурирования БД в кластерной среде является самым простым и не требует дополнительных настроек. Однако в этом случае БД не реплицируется, и все данные хранятся в одном единственном месте. Таким образом, БД является уким местом для всего отказоустойчивого кластера. Повреждения данных или сбой в работе ПО сервера БД может привести к прерыванию сервиса. Поэтому этот вариант довольно ограничен в применении.

Конфигурация

Ниже приводится список необходимых изменений для настройки основного и дублирующего узлов при использовании некластеризованной БД.

СЕРВЕР БАЗ ДАННЫХ

- Разрешите Вашему серверу БД принимать входящие соединения с IP-адресов главного и дублирующего узлов.
- Позвольте фаерволу на сервере БД принимать соединения для порта сервера БД (например, для порта 3306 для сервера MySQL) с IP-адресов обоих узлов.

ГЛАВНЫЙ СЕРВЕР ATOMMIND

- Если сервер уже настроен для подключения к упомянутому выше серверу БД, не трогайте его настройки.
- В ином случае переключите 698 Ваш главный сервер на означенную выше БД.

ДУБЛИРУЮЩИЙ СЕРВЕР ATOMMIND

- Используйте те же самые настройки подключения БД, как и на основном сервере.

Если Ваша БД работает на том же компьютере, где запущен основной сервер, Вам может понадобиться изменить настройку адреса БД в URL базы данных 183 с `localhost` на IP-адрес или имя хоста головного узла.

11.1.4 Файловое хранилище

Файловое хранилище позволяет AtomMind Server хранить отдельные элементы данных, такие как элементы конфигурации и двоичные блоки в отдельных файлах внутри локальной файловой системы сервера.

Так как конфигурация файлов и хранение двоичных данных создает минимальную нагрузку на систему, хранение файлов подходит для установок на системы с низкой производительностью, такие как Raspberry Pi, BeagleBone Black и другие одноплатные ПК на базе платформы ARM, работающие на операционной системе Linux.

Файловое хранилище не поддерживает любые дополнительные параметры конфигурации.

Файловое хранилище не подходит для длительной промышленной эксплуатации! Использование файлового хранилища в промышленных установках с высокой нагрузкой может привести к потере данных. Как только система переключена на продуктивный режим, необходимо переключиться на любую БД уровня предприятия. Неожиданная остановка AtomMind Server может вызвать нарушение БД.

11.2 Статистика

Платформа AtomMind может хранить долгосрочные серии данных в *Циклической Базе Данных* (RRD). Модуль, отвечающий за сбор данных, их хранение и обработку называется *Модуль Контроля Системы Статистики*, или просто *Статистика*.

Циклическая БД (RRD) нацелена управлять временными сериями данных, таких как пропускная способность сети, температуры, загрузка ЦП и т.д. Данные сохраняются в цикличном буфере, таким образом, размер хранимых данных остается неизменным со временем. Для более подробной информации см. статью Технология RRD 727.

БД RRD имеет два важных преимущества для хранения долговременных статистических данных:

- Небольшой и постоянный размер БД
- Очень быстрый доступ к данным истории за любой период времени

Каналы статистики

В AtomMind, Данные статистике на основе RRD представлены *Каналами статистики*. Каждый канал обрабатывает значение одной переменной контекста 61. Однако, так как Основные точки данных 727 канала рассчитываются при помощи выражений 112, исходные значения канала могут основываться на любых данных, проходящих внутри системы.

Каждый канал определяет:

- Выражение для определения значений Точек первичных данных
- Тип хранилища (файл или память)
- Тип 725 канала (индикатор, счетчик и т.д.)
- Активные функции агрегирования (среднее, минимальное и т.д.)
- Периоды хранения для различных интервалов агрегирования (почасовой, ежедневный и т.д.).
- И другие опции 719

В зависимости от типа хранилища статистики обеспечивается разный уровень целостности данных. Например, если используется файловый тип хранилища, в случае отключения электроэнерегии при написании сегмента статистики, файл может быть поврежден.

Применение статистики

На данный момент, следующие средства обработки данных AtomMind позволяют использовать статистику:

- Устройства 497. Можно создавать от одного и более каналов для хранения истории каждой переменной настройки устройства
- Датчики 2181. Каждый датчик имеет заранее определенный канал статистики для хранения изменений в истории.

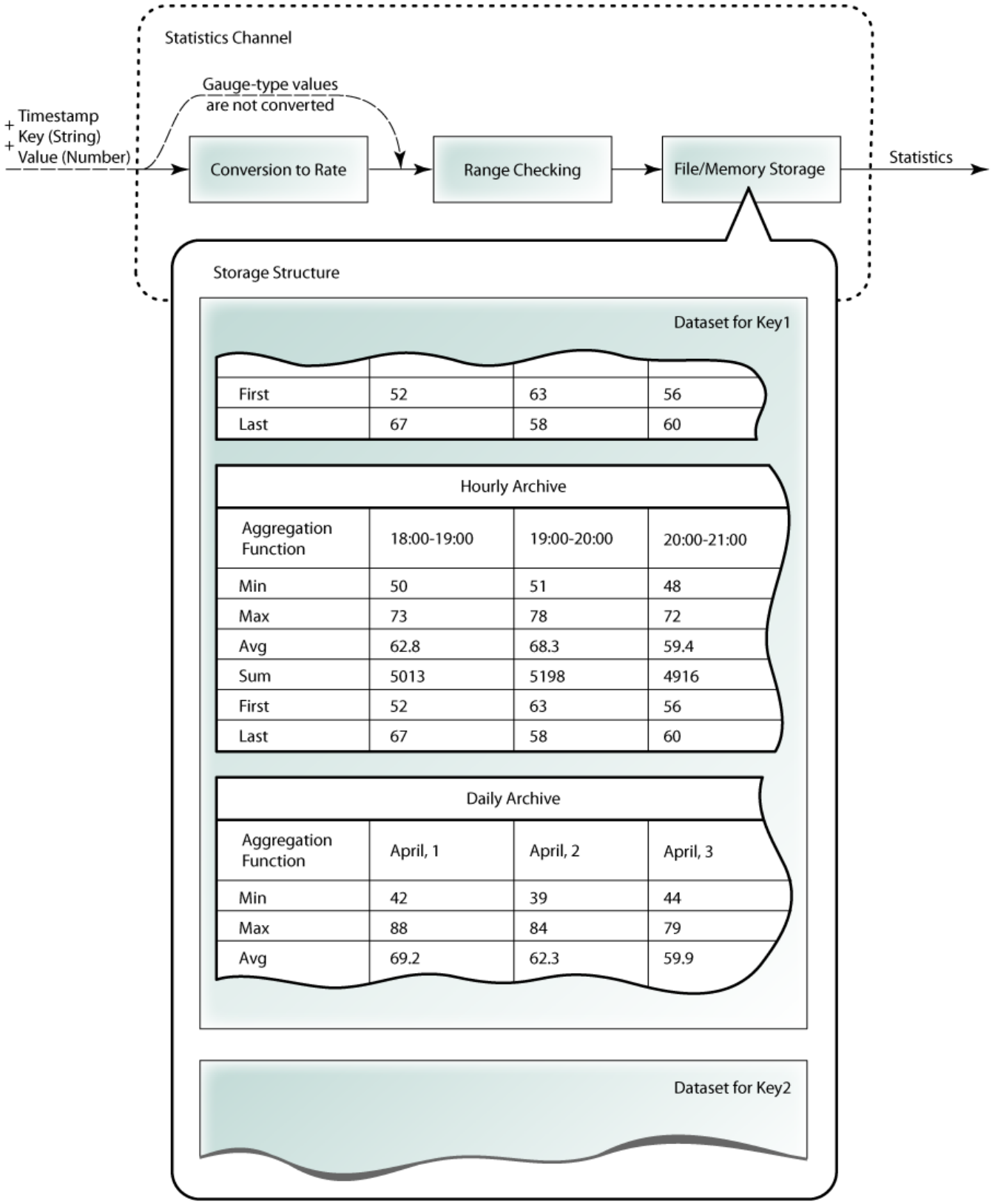
Более подробную информацию см. здесь:

- Статистика настроек устройства 507
- Статистика датчиков 2184

Элементы каналов

Каждый канал статистики состоит из следующих элементов:

- **Массивы данных**. Массивы данных позволяют хранить множество статистик для одной контекстной переменной. Простые переменные соответствуют одному массиву данных, в то время как каждый ряд в табличной переменной может идентифицироваться уникальным ключом для создания имени массива данных. Для более подробной информации см. Работа с ключом 724.
- **Архивы**. Каждый массив данных включает несколько Архивов, которые содержат данные, агрегированные за разные периоды времени. Например, массив может включать один Архив со среднемесячными данными, другой - максимальными почасовыми и т.д. Доступные типы Архивов: Поминутный, Почасовой, Ежедневный, Недельный, Месячный, Годовой.



11.2.1 Свойства статистического канала

Данная глава посвящена описанию свойств одного канала статистики.

Исходные данные

ВЫРАЖЕНИЕ

Свойство **Выражение** определяет, какие данные являются основой статистики. Оно вычисляется во время каждого изменения переменной, на которой построен канал.

Выражение должно иметь в результате выполнения число (целое или с плавающей запятой). Если значение выражения имеет любой другой тип данных, система пытается сконвертировать его в число или пропускает значение Если результат выражения NULL, система применяет метод **Управление значениями NULL** (см. далее).

Среда вычисления 112 выражения канала статистики:	
Контекст по умолчанию 119	Контекст, в котором определен канал.
Таблица данных по умолчанию 120	Текущее значение переменной, на которой основан канал.
Ряд по умолчанию 119	0, если свойство **Использовать Ключевое поле** отключено. Номер обрабатываемого на данный момент ряда, если свойство **Использовать Ключевое поле** включено.
Переменные среды 123	Только стандартные 123 переменные.

Ключевые поля

Канал может включать один и более Массивов данных для одной контекстной переменной. Для табличных переменных можно создать множество Массивов данных. В таком случае, каждый ряд в переменной распознается уникальным строковым ключом (имя Массива данных). Обратитесь к разделу Работа с ключом 724 для дальнейшего объяснения.

Обработка значений

Канал статистики предоставляет несколько опций для обработки значений, полученных **Выражением** канала:

- Обработка значений NULL
- Обработка значений, вышедших за пределы диапазона
- Минимальное значение
- Максимальное значение

ОБРАБОТКА ЗНАЧЕНИЙ NULL

Данная настройка определяет, что должен делать канал, когда его **Выражение** вернуло NULL. Возможны два варианта:

- **Исключать**. Значение будет просто игнорировано. Если не будет получено никаких других значений, общее значение канала станет НЕ ОПРЕДЕЛЕН для данного периода.
- **Приводить к нулю**. Значения NULL будут конвертированы в нули.

ОБРАБОКА ЗНАЧЕНИЙ, ВЫШЕДШИХ ЗА ПРЕДЕЛЫ ДИАПАЗОНА

Определяет, как обрабатывать значения, которые меньше **Минимального значения** или больше **Максимального**. Доступны два варианта:

- **Ничего не делать**. Просто отключает минимальные/максимальные значения для канала.
- **Отклонить**. Игнорирует значения меньше минимального или больше максимального. Если не будет получено никаких других значений, общее значение канала станет НЕ ОПРЕДЕЛЕН для данного периода.
- **Нормализовать**. Значения меньше минимального или больше максимального конвертируются в минимальное и максимальное соответственно.

МИНИМАЛЬНОЕ И МАКСИМАЛЬНОЕ ЗНАЧЕНИЕ

Эти числа определяют пороги отклонения/нормализации, как обусловлено опцией **Обработка данных, вышедших за пределы диапазона**.

Включенные для каналов типа 725 "счетчик" (Counter, Derive и Absolute), эти числа определяют минимальную и максимальную **величину в секунду**.

Настройка **минимального** и **максимального значений** очень полезна для работы со сбрасыванием или ошибочным уменьшением показателей счетчика. Если ваш счетчик может сбросить показатели до нуля или уменьшить их в любой момент (например, во время перезапуска устройства), у AtomMind Server нет способа определить, сбросились показатели или переполнились. Однако в случае сброса или уменьшения показатели, рассчитываемые в секунду, будут очень высоки. Например, если значение счетчика, измеряемое в 18:00:00, было равно 100 000, а следующий замер в 18:01:00 имел значение 1000, модуль статистики сочтет, что счетчик увеличился до 2^32 (4.2 миллиарда), переполнился и увеличился до 100 за эту минуту. Таким образом, рассчитанный в одну секунду показатель будет, конечно, выше, чем 1 000 000.

Для исключения подобных высоких показателей, вызванных сбрасыванием счетчика из вашей статистики, просто установите **максимальное значение** на нечто выше, чем ваш максимально ожидаемый показатель в секунду, но ниже, чем ненормальный "показатель сброса счетчика". Например, если максимально ожидаемый вами показатель в секунду равен 10 000, установите максимальное значение на 100 000 для исключения некорректных показателей и избежания захламления статистики.

Хранение статистики

По существу, данные статистики могут храниться в файлах и памяти. Хранение в файле постоянное, а данные каналов, основанных на памяти, сбрасываются при перезапуске сервера.

ХРАНЕНИЕ

Свойство Хранилище определяет тип хранения статистики. Три опции:

- Стандартный файл (Низкое использование памяти, медленее)
- Отображаемый файл (более высокое использование памяти, быстрее)
- Память

Хранение в памяти гарантирует максимальную производительность, но не сохраняет данные между перезапусками сервера, следовательно, его использование ограничено.

- NoSQL хранилище

Важно, что NoSQL хранилище работает только с хранилищем событий NoSQL, активированным в настройках базы данных.

Далее приведены некоторые советы по выбору между стандартными и отображаемыми в памяти файлами:

- Отображаемый в памяти файл кэширует данные канала в памяти (идущие не из JVM, а напрямую из операционной системы), время как данные обычного файла сохраняются на диске.
- Выполнение хранения на основе обоих файлов зависит от ОС, таким образом, в большинстве случаев необходимо провести какие-либо тесты, чтобы выбрать один из них.

Тип канала

Настройка **Тип** канала определяет, как обрабатываются исходные данные для формирования Точек первичных данных 727 для канала. Более подробно об этом см. в разделе Типы каналов 725.

Агрегирование данных

Модуль статистики использует Функции Агрегирования для конвертирования Точек первичных данных в Точки консолидированных данных (для более подробной информации обратитесь к разделу Технология RRD 727). Существуют функции агрегирования размера, которые можно включить/отключить для канала в отдельности:

- Среднее
- Минимальное
- Максимальное
- Общее
- Первое
- Последнее

Функции агрегирования данных применяются ко всем Архивам[718], включенным для канала. Например, если включена функция агрегирования **Среднее**, система сохраняет средние значения, такие как **Поминутный**, **Почасовой** и т.д.

СРЕДНЕЕ

Функция рассчитывает среднее значение Точек первичных данных для образования Точки консолидированных данных. Является средним самих значений (для канала Индикатор)[725] или величиной изменений значений (для Счетчика[725] и других типов каналов).

МИНИМАЛЬНОЕ

Данная функция берет минимальную Точку первичных данных и использует ее в качестве Точки консолидированных данных. Является самим минимальным значением (для канала Индиктор)[725] или минимальным значением величины изменений значения (для Счетчика[725] и других типов каналов).

МАКСИМАЛЬНОЕ

Приблизительно то же, что и **Минимальное**.

СУММА

Данная функция суммирует все Точки первичных данных для формирования Точки консолидированных данных. Является суммой самих значений (для канала Индикатор)[725] или величиной изменений значений (для Счетчика[725] и других типов каналов).

ПЕРВОЕ

Данная функция берет первую Точку первичных данных в качестве значения для Точки консолидированных данных.

ПОСЛЕДНЕЕ

Данная функция берет последнюю Точку первичных данных в качестве значения для Точки консолидированных данных.

Архивы

Каждый канал может иметь включенными все или несколько следующих Архивов[718]:

- Поминутный
- Почасовой
- Ежедневный (фиксированный 24-часовой период, см. примечание ниже)
- Еженедельный
- Месячный (фиксированный период, см. примечание ниже)
- Годовой (фиксированный период, см. примечание ниже)

Важная заметка: БД RRD и модуль статистики всегда имеют дело с фиксированными временными периодами. Это приводит к следующим последствиям:

1. Ежедневные архивы создаются на сутки. Это приводит к неверным результатам в несколько дней при переключении с летнего времени на обычное время и обратно.

2. Ежемесячные архивы создаются для средней длины месяца в течение любых четырех последовательных лет (три обычных года и один високосный год), что составляет около 30,43 дней. Это влияет на точность при расчете статистических значений за месяц и год для всех месяцев/лет.

Для получения более точных месячных и годовых результатов необходимо основывать вашу статистику на суточных образцах и использовать дополнительную агрегацию на последующих этапах обработки, например, агрегирование внутренних данных графика, которое будет группировать суточные образцы по «реальным» месяцам/годам.

Чтобы получить результаты с максимально возможным уровнем точности, основывайте свои расчеты на почасовых образцах статистики и используйте агрегацию более высокого уровня, чтобы сгруппировать результаты по дням, неделям, месяцам и годам.

Длину каждого архива, т.е. количество точек консолидированных данных, можно конфигурировать.

Чтобы полностью отключить архив, установите его длину на NULL (<*Не задано*>).

Системные администраторы определяют длину архива в "человеческих" временных единицах, в то время как число Точек консолидированных данных рассчитывается автоматически. Например, если длина Поминутного архива установлена на 1 day, он будет содержать 1440 CDP (т.е. 1440 **Средних**, 1440 **Минимальных**, 1440 **Максимальных** и 1440 **Общих**, при всех включенных функциях агрегирования).

Контрольный интервал

Максимальный период времени, который может пройти между двумя исходными значениями, полученными каналом, до того, как статистическое значение за определенный период будет считаться *неизвестным*.

Нулевое значение означает автоматический контрольный интервал.

Автоматический контрольный интервал рассчитывается следующим образом:

- Если статистический канал привязан к одной из настроек удаленных устройств 501 (перечислены в разделе параметры синхронизации настроек 502), контрольный интервал рассчитывается как ожидаемый период синхронизации для настройки (минимальные периоды полной и частичной синхронизации), умноженный на 2.
- Во всех остальных случаях невозможно автоматически определить период обновления значения, поэтому контрольный интервал постоянно устанавливается как 1 минута, умноженная на 2 (120 секунд).

Автоматический контрольный интервал рассчитан, в основном, на работу в статистических каналах, привязанных к настройкам удаленных устройств. В остальных случаях рекомендуется вручную устанавливать контрольный интервал, равный ожидаемому периоду обновления переменной канала, умноженному на 2.

ВЗАИМОДЕЙСТВИЕ МЕЖДУ ШАГОМ И КОНТРОЛЬНЫМ ИНТЕРВАЛОМ

Статистический канал получает исходные значения в произвольные моменты времени. Из них он строит Очки Первичных Данных (PDP) в определенное время каждого интервала Шаг. PDP затем собираются в CDP. См. Технология RRD 727 для разъяснения значений PDP и CDP.

Контрольный интервал определяет максимальный приемлемый интервал между образцами. Если интервал между образцами меньше контрольного интервала, то рассчитывается средний показатель и применяется для этого интервала. Если интервал между образцами длинее, чем контрольный интервал, то весь этот интервал считается *неизвестным*. Обратите внимание, что есть и другие факторы, которые могут сделать интервал образца *неизвестным*, такие как ограничения скорости превышения или даже *неизвестный* (NULL) вводный образец.

Известные показатели во время интервала Шаг PDP используются для расчета среднего показателя данного PDP. Кроме того, если общее *неизвестное* время в течение интервала Шаг превышает контрольный интервал, весь PDP помечается как *неизвестный*. Это означает, что совмещение известного и *неизвестного* времени образца в одном Шаге PDP может добавлять или не добавлять достаточно *неизвестного* времени, чтобы превысить контрольный интервал и, следовательно, отметить весь PDP как *неизвестный*. Итак, контрольный интервал - это не только максимально допустимый интервал между образцами, но и максимально допустимое количество *неизвестного* времени в PDP (очевидно, это имеет значение, только если контрольный интервал меньше Шага).

Контрольный интервал может быть коротким (необычным) или длинным (типичным) по отношению к интервалу Шаг между PDP. Короткий контрольный интервал означает, что требуется несколько образцов в PDP, и если этого нет, PDP помечается как неизвестный. Длинный контрольный интервал может занимать несколько Шагов, а это означает, что допустимо иметь несколько PDP, рассчитываемых из одного образца. Примером этого может быть Шаг в 5 минут и контрольный интервал в один день, и в этом случае один образец "каждый день" приведет к тому, что все PDP этого полного дня будут настроены на тот же средний показатель.

Выражение временной метки

Это выражение позволяет переопределить временную метку для каждого образца. Выражение должно возвращать данные типа дата. Каждый раз при появлении нового образца, выражение вычисляется и образец приобретает метку времени. Когда **выражение временной метки** не задано, текущее время принимается за временную метку образца.

Защитить статистику

Это полезный флаг для защиты канала статистики от удаления. При изменении одного из следующих двух параметров (**Шаг** и **XFiles фактор**), канал перенастраивается, и его данные теряются. Таким образом, если необходимо изменить защищенные параметры, следует сначала снять флаг **Защитить статистику**.

Шаг

Интервал грануляции для канала, более подробно об этом см. в Технологии RRD 727. В большинстве случаев **Шаг** лучше не менять.

Изменение **Шага** приведет к сбросу статистики канала.

Поскольку шаг по умолчанию равен 60 секундам, **Среднее**, **Минимальное** и **Максимальное** статистические значения для периода **Последняя минута** будут равны.

XFiles фактор

XFiles фактор определяет, какая часть интервала консолидации может быть сформирована из *неопределенных* точек данных, в то время как консолидированное значение все еще считается как известное. Записывается как скалярное отношение допустимого объема *неопределенных* данных к общему числу данных для конкретного временного интервала, в виде десятичной дроби в диапазоне от 0 до 1 (включительно).

Изменение **XFiles фактора** приведет к сбросу статистики канала.

Значение по умолчанию - 0.9, т.е. 90% исходных образцов могут быть потеряны.

Для увеличения надежности значений статистики, уменьшите это значение. Типичной настройкой является 0.5.

Показать в статусе

Определяет, будет ли отображаться краткая статистика канала в статусе контейнера (т.е. устройства или датчика).

11.2.1.1 Работа с ключами

Если исходная переменная канала является табличной, он может работать в двух режимах путем создания одного или множества *Массивов данных* RRD для одной переменной источника. Эти режимы переключаются с помощью флажка **Использовать ключевое поле**. О том, как использовать оба режима, см. в Примерах конфигурации статистики 726.

ИСПОЛЬЗОВАТЬ КЛЮЧЕВОЕ ПОЛЕ

Канал может работать в двух режимах:

- Если **Ключевое поле** отключено (настройка по умолчанию), для канала создается один Массив данных 718 (т.е. массив данных статистики).
- Если **Ключевое поле** включено, создается один Массив данных для каждой записи в значении исходной переменной. Эта опция имеет смысл, только для каналов, основанных на табличных переменных. Имя массива данных определяется уникальным ключом записи, который строится согласно свойству **Ключевое поле**.

Далее приеден пример действий системы при обновлении исходной переменной канала:

1) При отключенном **Ключевом поле Выражение** канала вычисляется один раз. При вычислении используется ряд по умолчанию 119. Результат оценки конвертируется в число и обновляется в наборе данных.

2) При включенном **ключевом поле**, **выражение** канала вычисляется один раз для каждого ряда, обнаруженного в исходной таблице. Для каждого ряда:

   a) Во-первых, система определяет, к какому массиву данных относится этот ряд. Это происходит с использованием **Ключевого поля** или значения первого поля, помеченного как ключевое 49 в формате переменной.

b) Во-вторых, **Выражение** канала вычисляется для текущего ряда, установленного в качестве ряда по умолчанию 119.

c) В-третьих, обновляется массив данных, определенный на шаге (a).

КЛЮЧЕВОЕ ПОЛЕ

Данное свойство определяет, какое поле исходной переменной канала используется для определения имен массива данных RRD, когда **Использовать ключевое поле** включен. Имя массива данных строится из записи Таблицы данных следующим образом:

- Если **Ключевое поле** не является NULL, значение этого поля используется в качестве имени массива данных;
- Если **Ключевое поле** является NULL, значение первого ключевого поля 49, определенного в табличном формате, будет использоваться в качестве имени массива данных;
- Если **Ключевое поле** является NULL и нет ключевых полей, определенных в табличном формате, имя массива данных будет совпадать с номером текущей записи.

11.2.1.2 Типы каналов

Настройка **Тип** канала полностью меняет стиль того, как сырые значения обрабатываются для расчета Точек первичных данных 727 канала. Существует четыре типа канала:

- Индикатор
- Счетчик с переполнением
- Счетчик без переполнения
- Сбрасываемый счетчик

Индикатор

Значение Точки первичных данных равно результату Выражения канала, т.е. единственной конвертацией является применение правил **Обработка значений NULL** и **Обработка значений вне диапазона.**

Данный тип подходит для простых измерений, таких как температура, количество людей в комнате или курс акций.

Счетчик с переполнением

Подходит для продолжительных увеличивающихся счетчиков. Источник данных для этого канала предполагает, что счетчик никогда не убавляется, за исключением, когда он переполнен. Функция обновления статистики принимает во внимание переполнение счетчика. Счетчик сохраняется в качестве величины в секунду. При переполнении счетчика, канал проверяет, произошло ли оно на 32 или 64-битной границе и действует соответственно, добавляя подходящее значение к результату.

Проще говоря, Счетчик с переполнением рассматривает различие между предыдущим значением и текущим (дельту). Примером послужит одометр. Точка первичных данных рассчитывается как разница показаний счетчика, деленная на разницу по времени.

Примеры счетчиков:

- Путь пробега транспортного средства (количество метров, пройденных устройством)

- Счетчик ошибок (количество ошибок, случившихся с момента запуска устройства)

- Трафик интерфейса (количество входящих/исходящих байтов интерфейса с момента включения сетевого устройства).

- Нагрузка процессора, вызванная процессом (количество миллисекунд времени процессора, потребляемого процессом с момента его запуска)

Счетчик без переполнения

Канал этого типа может измерять как увеличения, так и уменьшения. Его применение удобно для индикаторов, например, при измерении количества людей, входящих или покидающих комнату. Счетчик без переполнения работает так же, как и счетчик с переполнением, но без проверок переполнения. Поэтому, если ваш счетчик не сбрасывает значения на 32 или 64 битах, вы можете использовать Счетчик без переполнения и совместить его с **Минимальным значением**, равным нулю.

Эти два счетчика действительно похожи друг на друга, но счетчик без переполнения может идти в обратном направлении. Примером может послужить устройство, контролирующее реверсивный насос. Итоговая величина насоса может быть как отрицательной, так и положительной.

Сбрасываемый счетчик

Каналы этого типа используются для счетчиков, которые сбрасываются при чтении. Применяется для счетчиков, которые склонны к переполнению. Поэтому, вы сбрасываете их значения после каждого прочтения, чтобы быть уверенными, у вас есть максимальное количество времени до следующего переполнения. Другим применением может послужить счет, например, сообщений после последнего обновления.

Канал этого типа также похож на одометр, но здесь счетчик сбрасывается каждый раз при его чтении. Точка первичных данных рассчитывается как значение, деленное на разницу по времени.

11.2.2 Последовательность обработки данных

Данная статья описывает полную последовательность шагов, предпринимаемых каналом статистики для конвертирования значений полей переменной[61] AtomMind Server в точки данных статистики. Эта последовательность выполняется каждый раз при обнаружении изменения переменной.

1. Сначала, рассчитывается **Выражение** канала.
2. Применяется правило **Обработка значений NULL** для отбрасывания значений NULL или их конвертации в ноль.
3. Полученное значение, не являющееся NULL, конвертируется в число самым оптимальным способом. Значения, сконвертированные в числа, не могут быть отброшены.
4. Правило **Обработка значений вне диапазона** применяется для уверенности, что числовое значение находится в пределе диапазона, в противном случае оно будет отклонено/нормализовано.
5. Значение обрабатывается согласно **Типу** канала для формирования новой точки первичных данных канала. Для **Типа** Индикатор, значение остается, как оно есть. Для Счетчиков, итоговое значение будет представлять исходное значение величины изменений в секунду.
6. Значение Точки первичных данных используется для обновления значения Точки (Точек) консолидированных данных канала согласно включенным функциям агрегирования (**Среднее**, **Минимальное** и т.д.). Выполняется для каждого Архива (**Поминутного**, **Почасового** и т.д.).
7. Значение(я) Точки консолидированных данных сохраняются в **Хранилище** (в файле/памяти).

11.2.3 Построение графиков на основе статистики

Модуль Статистики предназначен для тесного взаимодействия с графиками[1051]. Можно построить на графике статистические данные по времени так же, как и сырые данные истории.

Чтобы построить график на основе статистики переменной, создайте График, основанный на переменной[1061] и используйте специальную ссылку **{statistics/}** в выражении серий данных для ссылки на данные статистики.

Для более подробной информации обратитесь к разделу Серии данных, основанные на статистике[1063].

11.2.4 Примеры конфигурации Статистики

Данный раздел предоставляет несколько примеров конфигурации каналов статистики.

Пример 1: Температура

Предположим, у нас есть одна переменная, которая представляет текущую температуру в градусах Цельсия. Эта переменная имеет только одну запись с одним полем **temp**. Однако, мы хотим, чтобы наша статистика основывалась на шкале измерений по Фаренгейту.

Свойства канала[719] нужно настроить следующим образом:

- **Выражение** канала должно быть `{temp} * 9 / 5 + 32`. Это приведет к конвертации температуры.
- Так как канал основан на простой (не табличной) переменной, настройка **Использовать ключевое поле** будет отключена.

Пример 2: Использование дисков

Данный пример показывает, как отследить использование дисков для случая, когда статистика сохраняется в отдельной таблице.

Предположим, мы имеем переменную со следующим значением:

Метка (Ключевое поле)	**Диск**	**Использовано**	**С у м м а**
00DFFF	C:	150	3 0 0
45CABC	D:	500	7 0 0
CBBC99	E:	10	3 0

Поле Метка является ключевым согласно формату вышеуказанной таблицы.

Свойства канала[719] нужно настроить следующим образом:

- Выражение нашего канала будет {**Used**}
- Так как нам нужно создать один массив данных RRD для каждого отслеживаемого диска, флажок **Использовать Ключевое поле** будет включен.
- Если мы оставим свойство **Имя ключевого поля** по умолчанию (NULL), система будет использовать значение поля **Label** (т.е. метки дисков) в качестве имен массивов данных, т.к. **Label** является ключевым полем. Однако, это неудобно для применения, поэтому мы зададим **Ключевое поле** на `Disk`. Это приведет к тому, что имена дисков будут использоваться для имен массивов данных RRD.

11.2.5 Технология RRD

RRD (Циклическая база данных) принимает данные, изменяющиеся во времени в интервалах определенной длины. Данные интервалы называются **шагами**. Так как данные не всегда могут быть доступны в нужное время, RRD автоматически интерполирует любые предлагаемые данные для заполнения внутренних временных шагов.

Значение для определенного шага, которое было интерполировано, называется Точкой первичных данных (PDP). Множество PDP может быть консолидировано согласно функции агрегирования (AG) для создания Точки консолидированных данных (CDP). Типичными функциями консолидации являются: **среднее**, **минимальное**, **максимальное** и **общее**.

После консолидации данных полученная CDP сохраняется в циклических архивах (RRA). Циклический архив хранит фиксированное количество CDP и определяет, сколько PDP должны быть консолидированы в одну CDP и какую AF использовать. Общее количество охватываемого времени рассчитывается следующим образом:

Time Covered = (#CDPs stored) * (#PDPs per CDP) * step

После этого архив будет перезаписываться: следующая вставка перезапишет старую запись. Данное поведение является циклическим, отсюда и название технологии.

Чем RRD отличается от Классической БД

1. В случае линейных БД, новые данные добавляются в конец таблицы БД. Таким образом ее размер увеличивается, в то время как размер RRD определяется в момент создания. Представьте RRD, как периметр окружности. Данные добавляются по периметру. Когда новые данные доходят до начальной точки, они перезаписывают уже существующие. Таким образом, размер циклической БД всегда постоянный.
2. Другие БД хранят значение в том виде, в каком они их получили. RRD может быть настроена на вычисление величины изменений между предыдущим и текущим значением и сохранять эту информацию.
3. Другие БД обновляются по мере поступления новых данных. RRD построена так, что ей нужны данные в заранее определенные интервалы времени. Если она не получает новое значение за этот интервал, она сохраняет *НЕИЗВЕСТНОЕ* значение для этого интервала.

11.2.5.1 Скорость изменения, нормализация и консолидация

Модуль статистики сохраняет *скорость изменения (скорость)* в течение временных интервалов. Эти временные интервалы находятся в хорошо определенных во времени границах. Однако ваш ввод не всегда является *скоростью изменения* и, скорее всего, не войдет в эти границы. Это значит, что ваш ввод должен меняться. Данный раздел объясняет, как это работает.

Следует выделить пару других этапов:

- Преобразование в *скорость*
- Нормализация интервала
- Консолидация интервалов в более большие

Это не мешает вам и не причиняет вред вашим данным. Это то, как работает модуль SPC в соответствии с проектом.

Каждый этап применяется для всего ввода, без каких-либо исключений. После преобразования вашего ввода в *скорость* происходит нормализация. После нормализации происходит консолидация. Все три этапа могут пропускаться, если вы тщательно настроили свою базу данных, но наличие одного пропущенного этапа не означает, что другие этапы будут пропускаться.

Если вы используете Индикатор 725, ввод уже является *скоростью*, но все же подлежит нормализации. Если вы вводите данные точно на границе нормализации, ваш ввод все равно подлежит консолидации.

Трансформация в скорость изменения

Все обрабатывается как *скорость*. Это не означает, что вы не можете работать с температурой, просто запомните, что она обрабатывается так, как если бы она так же была *скоростью*.

Существует несколько способов получения скорости из ввода модуля SPC (в зависимости от типа канала 725):

Индикатор:

Сохраняется его в "исходном виде". Ввод уже является *скоростью*. Примером может служить спидометр. Это тоже тип, используемый для отслеживания температуры и подобных показателей.

Сохранение в "исходном виде" не означает, что нормализация и консолидация пропускаются! Только сам этап.

Счетчик с переполнением:

Оценивается разница между предыдущим и текущим значением (дельта). Примером может служить одометр. Скорость рассчитывается так: дельта(счетчик с переполнением) разделенная на дельту (время).

Сбрасываемый счетчик:

Как одометр, но в этом случае счетчик переустанавливается при каждом чтении. Рассчитывается так: значение делится на дельту (время).

Счетчик без переполнения:

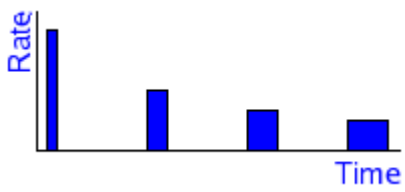
Как счетчик, но также может идти в обратную сторону. Примером может служить мониторинг двунаправленного насоса. Результат может быть как положительным, так и отрицательным.

В каждом из этих четырех случаев результатом будет *скорость,* которая действительна между предыдущим обновлением статистического канала и текущим. Модуль SPC не должен ничего знать о вводе, у него есть начало, конец и *скорость*.

Так завершается этап 1. Данные становятся *скоростью*, независимо от того, какой источник данных вы используете. С этого момента модуль SPC не знает и не проверяет, какой источник данных вы используете.

О скорости и времени

Если вы передаете что-то на скорости 60 байт в секунду в течение 1 секунды, вы можете передавать такое же количество данных на скорости 30 байт в секунду в течение 2 секунд, 20 байт в секунду в течение 3 секунд или 15 байт в секунду в течение 4 секунд, и так далее.

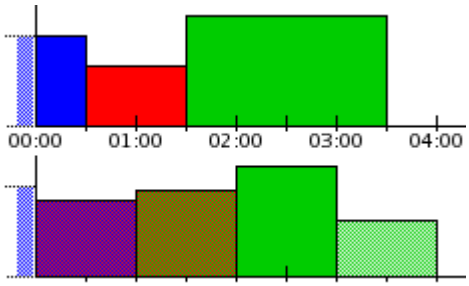


Данные числа все разные, но имеют одно общее: *скорость*, умноженная на время является константой. На этом рисунке важна сама поверхность, не ширина или высота, поскольку мы смотрим на количество данных, не их *скорость* или время. Далее объясняется, почему это важно.

Интервалы нормализации

Ввод теперь является *скоростью*, но он еще не находится в хорошо определенных границах времени. В этом случае нужна нормализация. Предположим, вы смотрите на счетчик каждую минуту. Что вы знаете? Вы знаете значения счетчика в ЧЧ:ММ:СС. Вы не знаете, увеличил ли счетчик свои показатели на высокой *скорости* в течение небольшого промежутка времени (1 секунда на скорости 60 байт в секунду) или в течение длительного времени на небольшой *скорости* (60 секунд на скорости 1 байт в секунду). Посмотрите на рисунок выше еще раз, каждый ЧЧ:ММ:СС будет находиться где-то в белых областях.

Это означает, что *скорость*, которую, как вы думаете, вы знаете, вообще не является реальной *скоростью*! В этом примере вы только знаете, что передали 60 байт за 60 секунд, где-то между ММ:СС и следующим ММ:СС. Данная полученная *скорость* будет равняться 1 байту в секунду в течение каждого интервала в 60 секунд. Подчеркнем следующее: **вы не знаете реальную *скорость***, только приблизительную.



Теперь посмотрите на другое изображение, показывающее замеренные интервалы и скорость. Образцы брались каждые 30 секунд в минуту, каждая закрашенная область представляет новый замер. Есть четыре закрашенных интервала, у последнего нулевая скорость, но известная (т.е. последнее обновление случилось в 04:30). Одно ожидаемое обновление, в 02:30, не произошло. Модуль SPC может идеально справиться с этим, если вы позволите. Обновление просто происходит в 03:30 и действительно до 01:30.

Нижняя часть изображения является результатом, полученным после нормализации. Она показывает, что каждый интервал использует часть каждого вводимого интервала. Первый интервал выстраивается из синего интервала (который начался до 00:00) и красного интервала (измеренного между 00:30 и 01:30). Используется только синяя часть, выпадающая в интервале 00:00 - 01:00, используется только красная часть, выпадающая в интервале 00:00 - 01:00. Подобное же происходит с другими интервалами. Заметим, что это только области, которые важны в данном случае. Используется хорошо определяемая часть синей области (в данном примере ровно половина) и хорошо определяемая часть красной области. Обе представляют байты, переданные за интервал. В этом примере используется половина каждого интервала, таким образом, мы получаем количество ранвое половине переданных байтов. Новый интервал, созданный в процессе нормализации, представляет собой сумму этих двух количеств. Его время известно, это фиксированное количество времени, размер шага, указанный вами для вашей базы данных. Его скорость - это область, поделенная на данное количество времени.

Если вы думаете, что неправильно перемещать данные по кругу таким образом, подумайте еще раз. Посмотрите на красный интервал. Вы знаете, что что-то произошло между 00:30 и 01:30. Вы знаете количество переданных данных, но **вы не знает, когда**. Возможно, что все они были переданы в первой половине этого интервала. Также возможно, что все они были переданы во второй половине этого интервала. В обоих случаях реальная скорость могла быть в два раза выше измеренной вами! Есть смысл разбить передачу так, как мы сделали. Вы все еще не знаете, правильно это или нет. В долгосрочной перспективе разницы нет, данные передаются, и мы об этом знаем.

Теперь скорости нормализованы. Это те скорости, с которыми работает модуль статистики. Заметим, что вторая и четвертая нормализованная скорость (сочетание красной и зеленой, сочетание зеленой и белой) ниже, чем зеленая скорость. Это важно, когда вы смотрите на максимальные скорости. Но поскольку красная и зеленая скорости являются сами по себе средними, их сочетание действительно так же, как и его источники.

Каждая нормализованная скорость действительна в течение фиксированного количества времени. Вместе они называются первичными данными - Primary Data Points (PDPs). Каждая PDP действительна в течение размера шага. Модуль SPC не знает и не обращает внимания на ввод, который вы ему предоставили. На этом завершается этап 2, и модуль SPC забывает весь оригинальный ввод.

Даже если нормализация является пустой (если вы убедились, что ваши временные метки уже находятся в хорошо определенных границах), все равно применяется консолидация.

Консолидация интервалов

Предположим, вы собираетесь представить свои данные в виде изображения. Вы хотите увидеть десять дней данных на одном графике. Если каждый PDP имеет в ширину одну минуту, вам нужно 10*24*60 PDP (10 дней 24 часов 60 минут). 14400 PDP - это много, особенно если ваше изображение будет иметь только 360 пикселей в ширину. Есть один способ показать свою информацию, а именно собрать вместе несколько PDP и показывать их

как один пиксель-столбец. В этом случае вам нужно 40 PDP за раз для каждого из 360 столбцов, чтобы получить в общей сложности десять дней. Совмещение этих 40 PDP называется консолидацией, что может быть сделано несколькими способами:

Средний:

Рассчитывает среднее значение каждой *скорости* (из этих 40)

Минимальный:

Берет минимально зафиксированную *скорость* (из этих 40)

Максимальный:

Берет максимально зафиксированную *скорость* (из этих 40)

Суммарный:

Берет сумму этих 40 *скоростей*

Первый:

Берет первую зафиксированную *скорость* (из этих 40)

Последний:

Берет последнюю зафиксированную *скорость* (из этих 40)

То, какую функцию вы будете использовать, зависит от вашей задачи. Иногда вам нужно видеть средние значения, так что вы можете использовать ее, чтобы видеть количество переданных данных. Иногда вам нужно видеть максимальные значения, чтобы определить периоды перегрузки и т.д.

Какую бы функцию вы не выбрали, потребуется время для вычисления результатов. 40 на 360 - это немного, но представьте, что произойдет в случае с большими периодами времени (такими как несколько лет). Это значит, что вам нужно будет подождать, пока сгенерируется изображение.

Эту операцию также можно осуществить при помощи модуля SPC, но для этого потребуется планирование наперед. В этом примере вам нужно будет использовать 40 PDP одновременно. В других случаях каждый раз потребуется иное количество PDP, но вы можете заранее знать, какими будет это количество. Вместо произведения расчетов в момент использования, модуль SPC может производить их в период мониторинга. Каждый раз, когда известны серии 40 PDP, он консолидирует их и сохраняет как консодидированные данные - Consolidated Data Point (CDP). Эти CPD сохраняются в базе данных. Даже если консолидация не потребуется, вы сможете "консолидировать" один PDP в один CDP.

11.2.6 Агрегирование сервера и устройств

Многие устройства, которые подключаются к AtomMind (через Agent или драйверы устройств 518), предоставляют различные значения типа датчика или счетчика, такие как показатели температуры, одометра или потребления топлива. Эти значения обычно группируются (агрегируются) по временным периодам, чтобы можно было использовать консолидированные значения (средние, минимальные и максимальные) в отчетах 928, графиках 1051 и других средствах обработки данных.

Разработчики систем мониторинга и контроля устройств часто сталкиваются с необходимостью выбора между агрегированием данных в устройстве и передачей необработанных значений в AtomMind Server. Эта статья разъясняет все "за" и "против" каждого метода.

Агрегирование данных внутри устройства

В этом случае устройство само рассчитывает консолидированные значения (такие как ежедневные средние значения и ежемесячные минимальные). Затем оно предоставляет серверу заранее рассчитанные значения в форме табличных переменных, таких как:

Месяц	Среднее потребление топилива
Июнь 2012	124
Июль 2012	131
Август 2012	119

"За" этого метода:

- Устройство может продолжить агрегирование данных, даже если связь с сервером недоступна длительное время (однако некоторые протоколы, такие как протокол AtomMind, позволяют произвести буферизацию устройства)
- Агрегированная статистика доступна внутри устройства, например, для отображения на ЖК устройства или внутреннем веб-интерфейсе

"Против" этого метода:

- Реализация агрегирования данных внутри устройства довольно сложна
- Агрегированная статистика займет место в памяти устройства
- Если потребуется много типов агрегации (например, для построения графиков), таких как средние/минимальные/максимальные показатели в минуту/час/день/неделю/месяц/год, устройство должно обеспечить множество таблиц агрегации для одного счетчика
- Таблицы агрегации будут полностью заново прочитаны из сервера во время обновления, частичное чтение невозможно
- Данные агрегации будут дублированы в устройстве и базе данных сервера
- Графики не смогут автоматически переключаться между различными типами агрегирования (например, с ежедневных на ежемесячные образцы)

Агрегирование данных на сервере с использованием статистических каналов

В этом случае устройство предоставляет серверу необработанные значения в форме переменных. Сервер опрашивает эти переменные и использует статистические каналы 718 для расчета и хранения агрегированных точек данных.

"За" этого метода:

- Только необработанные значения отправляются на сервер и, таким образом, трафик устройство-сервер минимизируется
- Ресурсы устройства не используются для расчета, хранения и предоставления агрегированных данных
- Сервер обеспечивает централизованный контроль агрегированных параметров
- Агрегированные точки данных из множества устройств могут легко совмещаться в отчетах и графиках сервера

"Против" этого метода:

- Кроме тех случаев, когда используется буферизация, сервер не получает необработанные значения от отключенных устройств, что приводит к пропускам в детализированной статистике (поминутной, почасовой) и снижению точности менее детализированной статистики (по дням, месяцам, годам)
- Само устройство не сможет воспользоваться консолидированными данными, например, отобразить их на веб-интерфейсе

Использование обоих методов

В некоторых системах возможно совместить оба метода, например, агрегировать данные как на устройстве, так и на сервере. В этом случае некоторые средства анализа данных сервера будут использовать заранее созданные таблицы консолидации устройства, а другие будут ссылаться на статистические каналы сервера.

11.3 Грануляция

Движок грануляции предназначен для агрегирования данных временных рядов. Для этого же предназначена подсистема Статистика в AtomMind, но движок грануляции обладает более широким набором вариантов агрегации, чем могут предложить статистические каналы. Это достигается ценой более низкой производительности и большего потребления памяти/места на диске.

Основные особенности движка грануляции:

- Позволяет вычислять и виртуально хранить любые пользовательские данные с любой необходимой структурой данных, не только простые числа. При этом, конечно, поддерживаются стандартные функции агрегирования. Например, функция Среднее можно реализовать путем сбора количества образцов и их суммы.
- В отличие от статистических каналов, движок грануляции дает возможность агрегировать данные практически за любые (определенные пользователем) периоды времени, равномерные и неравномерные.
- Движок грануляции никогда "не закрывает" периоды агрегации, давая возможность добавлять информацию к любому периоду в любой момент. В частности, это означает, что данные можно обрабатывать в любом порядке: если какие-то данные поступают с задержкой, вы все равно можете их обработать и добавить агрегат данных к временному интервалу в прошлом.

Если вы знакомы со статистикой на базе RRD, вы можете провести аналогию между ней и движком грануляции. Статистические данные на базе RRD представлены именованными статистическими каналами, каждый из которых обрабатывает значения переменной контекста. В движке грануляции есть очень похожий аналог статистического канала, который называется гранулятором.

Признаки гранулятора

Гранулятор определяется в контексте следующими признаками:

Имя поля	Тип	Описание
Имя	String	Строка, которая идентифицирует гранулятор в контексте, в котором он определен
Переменная	String	Имя переменной в контексте, которая запускает гранулятор
Активный	Boolean	Флажок, показывающий, активен ли гранулятор и обрабатываются ли изменения переменной
Формат значения	DataTable	Формат хранения агрегированных значений в таблице
Выражение значения	String	Выражение для вычисления нового агрегированного значения на базе предыдущего значения и значения новой входящей переменной
Выражение временной метки	String	Выражения, определяющее и идентифицирующее временные периоды
Время хранения	Long	Период времени, определяющий срок хранения гранулы
Точность временной метки	Long	Параметр точности определяет временной интервал, используемый временной меткой для поиска гранул. Из-за ошибок округления временная метка гранулы может отличаться от рассчитанного значения. Поэтому движок поиска гранул находит гранулы, которые лежат с интервалом, близким к рассчитанной временной метке. Интервал определяется как [временная метка - точность, временная метка + точность].

Гранулы

Данные собираются в "Гранулы". Каждая гранула представляет собой определенный временной интервал агрегации данных и содержит:

- Временную метку, которая представляет интервал времени, в течение которого гранула хранит информацию.
- Накопленное значение для временного интервала. Накопленное значение - это таблица данных с форматом, определенным в поле Формат значения.

Предполагается, что временная метка гранулы находится в отношении один к одному с временным интервалом, который она представляет; в то же время только временная метка определяет данную гранулу внутри ее гранулятора.

Например, гранулятор, который каждый час агрегирует средние значения, должен выдавать гранулу для каждого часа; эти гранулы могут быть определены, например, временными метками с нулевыми минутами, секундами и миллисекундами. Существуют, конечно, другие способы воплощения на усмотрение разработчика.

Алгоритм грануляции

Включенный гранулятор автоматически активируется при изменении соответствующей переменной. Другой способ инициировать грануляцию - вызвать функцию Гранулировать 733 контекста переменной.

Гранулятор берет временную метку события изменения и вновь поступившее значение переменной, находит или создает гранулу, которая представляет временной интервал, под который подпадает событие, и обновляет значение гранулы.

Таким образом, алгоритм грануляции включает следующие стадии:

- Вычисление временной метки гранулы, которая определяет требуемый временной интервал и гранулу
- Получение гранулы по рассчитанной временной метке
- Обновление данных гранулы в соответствии с вновь поступившим значением и их хранение

Ниже алгоритм описан более подробно.

ВЫЧИСЛЕНИЕ ВРЕМЕННОЙ МЕТКИ ГРАНУЛЫ

Прежде всего, гранулятор должен определять временной интервал, под который подпадают входящие данные. Интервалы определяются временными метками.

Выражение временной метки используется для преобразования времени события во временную метку гранулы. Разработчик должен убедиться, что преобразование осуществлено 1:1.

Во время вычисления выражения временной метки, время обновления переменной передается среде вычисления 114 как таблица данных по умолчанию с единственным значением. Так, чтобы получить время обновления, можно использовать следующее выражение:

```
cell(dt())
```

Значение может быть преобразовано во временную метку гранулы. Например, для сбора поминутных данных, можно использовать следующее выражение для временных меток гранул:

```
date(year(cell(dt())), month(cell(dt())), day(cell(dt())), hour(cell(dt())),
minute(cell(dt())), 0)
```

Обратите внимание, что вы можете легко предоставить неравные календарные периоды, например, месяцы:

```
date(year(cell(dt())), month(cell(dt())), 0, 0, 0, 0)
```

ПОЛУЧЕНИЕ ГРАНУЛЫ

Гранулы хранятся как события контекста гранулятора по следующим правилам:

- Имя события совпадает с именем гранулятора
- Временная метка сохраняется как время создания события
- Данные гранулы хранятся в таблице данных события.

Имея временную метку, движок грануляции находит гранулу, которая представляет данные для соответствующего временного интервала.

Если гранула еще не существует, новое событие создается с временной меткой и таблицей данных формата, указанного в поле Формат значения.

ОБНОВЛЕНИЕ ГРАНУЛЫ

Выражение значения используется для вычисления значения новой гранулы на основе входящей переменной и предыдущих значений гранулы.

Среда вычисления выражения включает:

- новое значение переменной как таблицу данных по умолчанию; поэтому невозможно получить к ней доступ с использованием функции `dt()`;
- текущее значение гранулы "предыдущая" переменная среды; используйте `{env/previous}`, чтобы получить ее значение.

Результат вычисления выражения значения сохраняется обратно в том же событии, что и новое значение для гранулы.

Помните, что новое значение разворачивается из таблицы до сохранения значения в грануле. Это может вызвать неудобство при попытке сохранить таблицу данных с большим количеством записей как значение гранулы. В этом случае не забывайте упакеовывать получившееся значение во внешнюю таблицу данных:

```
table("<<data><T>>", dt())
```

После обновления гранулы запускается соответствующее событие в контексте владельца (см. ниже 734).

Функция ГРАНУЛИРОВАТЬ

Функция позволяет инициировать грануляцию переменной. Это может быть полезно, например, при обновлении переменной задним числом, когда данные поступают с задержкой.

Функция берет имя переменной, новое значение и временную метку (как показано в таблице ниже) и начинает выполнение алгоритма грануляции, описанного выше 732.

Имя поля	Тип	Описание
Имя переменной	String	Имя переменной для грануляции
Значение	DataTable	Входящее значение

Временная метка	Date	Момент изменения значения

Доступ к гранулам

Гранулы можно читать как обычные события в контексте гранулятора, например, с использованием функции Получить 1518 из контекста события.

Например, чтобы получить все гранулы с именем 'testGranulator' в контексте 'users.admin.devices.virtual', можно использовать следующее выражение:

```
{events:get("users.admin.devices.virtual", {form/cbGranule:selectedItem}, NULL, NULL, NULL)}
```

Обновленное событие гранулы

Когда для контекста активируется грануляция, добавляется событие Обновление гранулы со следующим форматом:

Имя поля	**Тип**	**Описание**
Имя	String	Имя гранулятора
Временная метка	Date	Момент обновления значения гранулы
Значение	DataTable	Новое значение гранулы

Событие запускается, когда гранула обновлена.

12 Обработка данных и аналитика

Этот раздел описывает все модули и функции, относящиеся к обработке и анализу данных, полученных от устройств и сгенерированных самой системой.

12.1 Привязки

Почему именно такое название? *Привязки* называются именно так, потому что они привязывают различные данные друг к другу. Подобно тому, когда вы *привязываете* что-нибудь в реальном мире. В этом и заключается весь смысл - крепко прикрепить что-либо к чему-либо. В случае данных, вы связываете одни данные с другими.

Например, у вас есть недоступное для изменения текстовое поле (выделенное серым цветом). Данное текстовое поле может иметь рядом контрольную кнопку, "Включить данную настройку?". Как только контрольная кнопка нажата, текстовое поле становится белым (включенным), и теперь вы можете в него что-нибудь вписать. Все это происходит в результате привязки данных - *значение* контрольной кнопки (Включено или Отключено) *привязано* к состоянию текстового поля (Включено или Отключено). Привязки данных AtomMind позволяют вам проделывать такие приемы в интерфейсе при помощи виджетов 943 - это всего лишь один пример из всех возможностей..

Каждая привязка состоит из двух частей:

```
reference = expression
```

Если вы еще не знакомы с понятиями ссылки 117 или выражения 112, обратитесь к двум предыдущим главам. Без полного понятия ссылок и выражений, нельзя понять как работают привязки.

`reference` определяет цель *привязки*. Место, куда будет записан результат выражения при обработке привязки. В зависимости от среды обработки, ссылка может указывать на ячейку Таблицы данных 49, свойство компонента Пользовательского интерфейса и т.д. Более подробную информацию о ссылках см. в главе Ссылки 117.

`expression` - выражение 112 AtomMind, определяющее значение, которое должно быть записано в цель привязки при ее обработке.

Для более подробной информации о применении привязок данных обратитесь к следующим главам:

- Привязки в Таблицах данных 741
- Привязки в Виджетах 1295
- Привязки в Моделях 821

12.1.1 Свойства привязок

Каждая привязка имеет следующие параметры:

Цель	Цель привязки представляет собой особый тип ссылки 117, которая указывает, куда будет записываться результат оценки выражения привязки во время её выполнения. Более подробно об этом см. в разделе Цель привязки 1295.
Выражение	Выражение AtomMind 112, вычисляемое каждый раз при выполнении привязки. Результат вычисления сохраняется как **цель привязки**. Более подробно об этом см. в разделе Выражение привязки 1296.
Активатор	Ссылка 118, указывающая на событие или свойство контекста, которое запускает выполнение привязки. Параметр **Активатор** доступен только при включенном параметре **При событии**. Более подробно об этом см. в разделе Активатор привязки 1296. Если **Активатор** не указан, а у привязки все еще есть параметр для вычисления, более подробно об этом см. **При событии**.
Условие	Условие - это Выражение AtomMind 112, которое вычисляется первым после активации привязки. Если это выражение имеет результатом false, выполнение привязки пропускается.
При запуске	Когда этот параметр включен, привязка обрабатывается каждый раз при запуске набора привязок.
При событии	Когда данный параметр включен и задан **Активатор**, привязка обрабатывается каждый раз во время внесения изменений в свойство, на которое ссылается активатор. Если

	Активатор ссылается на событие, выполнение привязки происходит во время запуска события. Если параметр **При событии** включен, а **Активатор** не задан, привязка выполняется автоматически: **Выражение** привязки включает в себя ссылки, указывающие на одну и более переменных. Изменения в любой их этих переменных приведет к запуску привязки. Например, предположим, что мы имеем следующую привязку, однако, **Активатор** для неё не задан: **Цель:** `users.admin.deviceservers.ds1.devices.thermostat:temperature$temperature` **Выражение:** `{user.admin.devices.sensor:temperatureField:value} + {user.admin.devices.sensor:temperatureAdjustmentField:value}` Выражение данной привязки берет значение из поля формы с именем `temperatureField` и добавляет его к значению поля `temperatueAdjustmentField`. Итоговый результат записывается в **Цель**. Теперь, т.к. привязка не имеет **Активатора**, но параметр **При событии** включен, выполнение привязки будет происходить при изменении пользователем одного из полей формы, определенных в Выражении.
Периодически	При включенном данном параметре выполнение привязки осуществляется периодически.
Период	Интервал между сеансами выполнения привязки. Изменение данного параметра возможно только при выключенном параметре **Периодически**.
Очередь	Имя очереди обработки целей привязки. Система привязок гарантирует, что доступ к целям привязок, принадлежащим одной и той же очереди, будет осуществляться последовательно, в том же порядке, в котором выполнялись привязки. Параметр Очередь не поддерживается в некоторых средах вычисления привязок, например, в привязках модели 814.

12.1.2 Привязки сервера

Движок виджетов 943 и моделей 810 AtomMind строится из *привязок контекстов сервера*. Эти привязки определяют отношения между единицами данных (такими как переменные 61, функции 70 и события 73) различных контекстов 41 сервера. Каждая привязка *рассчитывается* на различных этапах выполнения виджета или модели. Результаты расчета используются для изменения данных контекста.

Привязки описаны в разделе Привязки 735. Пожалуйста, прочитайте этот раздел и темы, с которыми он соотносится. Только после того, как вы поймете привязки, ссылки и выражения, вы сможете с легкостью понять эту тему.

У каждой привязки есть три части:

- Активатор 739 и Условие 740 привязки, определяющие, ***КОГДА*** обрабатывается привязка. Обработка может происходить при запуске виджета, нажатии кнопки, изменениях в модели данных или свойствах компонентов виджета и т.д.
- Выражение 738 привязки, определяющее, ***КАКОВ*** результат обработки привязок. Выражение привязки может ссылаться на различные свойства компонента "виджет" или различные данные из контекстов сервера. Это записывается в Языке выражений 112 AtomMind.
- Цель 737 привязки, определяющая, ***ГДЕ*** будет записываться результат расчета выражения привязки, т.е., какое свойство компонента графического интерфейса пользователя или какие данные контекста будут меняться. Это Ссылка 117.

ПОРЯДОК АКТИВАЦИИ ПРИВЯЗОК

Поскольку привязки выполняются параллельно 742, порядок их выполнения нефиксированнный. Если две привязки активируются одновременно, нельзя определить, какая из них будет выполнена первой. Это важно учитывать при создании привязок, зависящих от результатов других привязок.

Например, одна привязка обновляет свойство контекста, а другая привязка активируется этим событием. Она запускает функцию, если указанное свойство контекста было обновлено. Обе привязки активируются одновременно. Если привязка, обновляющая свойство, выполнится первой, вторая привязка вызовет функцию. Если же первой будет выполнена привязка, вызывающая функцию, функция не будет вызвана. В результате, событие, которое должно было запустить выполнение этой привязки, будет проигнорировано.

Чтобы избежать непредсказуемости в поведении привязок, зависящих от других привязок, обязательно указывайте правильную цепочку активаторов привязок. Если привязка A зависит от активации привязки B, то активатор привязки A должен быть результатом активации привязки B. Например, если привязка A меняет свойство контекста, то привязка B должна активироваться этим свойством, а не отдельным событием. Это обеспечит выполнение привязок одну за другой, в правильном порядке.

Единственный способ получить предсказуемый порядок выполнения привязок - это отключить параллелизм их выполнения, например, установить размер пулов потоков привязок на 1. Подробнее об использовании пулов потоков см. в разделе Выполнение привязок 742.

12.1.2.1 Цель привязки

Цель привязки - это объект, находящийся под влиянием привязки. Цель привязки указывает на переменную контекста, поле переменной, функцию контекста или событие контекста.

Фактически цель привязки - это особый тип ссылки 117.

Существует несколько поддерживаемых типов целей привязки:

1. Переменная контекста сервера

`context:variable`

Цель привязки указывает на значение **переменной** контекста. Поскольку переменная всегда является таблицей данных 49, выражение привязки должно разрешаться в таблицу данных, чтобы позволить процессору привязок использовать эту таблицу как новое значение переменной.

2. Поле переменной контекста сервера

`context:variable$field`

Цель привязки указывает на особое **поле** в пределах **переменной контекста** сервера. Только значение этого поля в первой записи таблицы данных переменной будет модифицироваться привязкой.

Пример: `users.admin:childInfo$firstname`

Эта цель привязки изменит поле `firstname` переменной `childInfo` для `users.admin`.

3. Функция контекста

`context:function()`

Этот тип цели указывает на **функцию контекста**. Когда обрабатывается привязка с такой целью, вызывается эта функция. Выражение привязки должно вернуть таблицу данных 49, которая будет использоваться как ввод функции.

4. Событие контекста

`context:event@`

Этот тип цели заставляет привязку запустить событие типа **событие** в **контексте**. Выражение привязки должно вернуть таблицу данных, чье значение будет обрабатываться как данные события 74.

5. Действие контекста

`context:action!`

Эта цель начинает интерактивное выполнение действия **действие** из контекста **контекст**. Результат расчета привязки передается действию как значение ввода. Он должен иметь тип таблица данных 49.

Выполнение действия недоступно в пределах привязок виджета 1295, если виджет запущен как автономное приложение (вне AtomMind Client).

Пример: `cardholders:import!`

Запускает действие импорта держателей карт (модуль Держатели карт - это часть решений Учет рабочего времени и Контроль доступа), которое является действием под названием **импорт** в контексте с путем **держатели карт**.

ЗАПУСК ОТНОСИТЕЛЬНЫХ ВИДЖЕТОВ, ОТЧЕТОВ И ИНСТРУМЕНТАЛЬНЫХ ПАНЕЛЕЙ

Виджеты 943, отчеты 928, инструментальные панели 912 и некоторые другие системные объекты могут быть *абсолютными* и *относительными*. Абсолютный объект запускается сам по себе, относительный объект запускается **для определенного контекста** 41, который служит в качестве источника данных.

Таким образом, запуск относительного виджета, отчета или инструментальной панели через цель привязки напрямую не позволит системе знать, *для какого контекста он должен запускаться*, когда выполняется привязка. Вместо этого действие запуска объекта, **расположенного в целевом контексте**, должно быть определено в цели привязки.

Пример: Предположим, у нас есть относительный виджет **График трафика**, который действителен для всех сетевых устройств 497. Вы можете открыть график трафика **определенного устройства**, используя активатор привязки. Чтобы это осуществить:

- Найдите исходное устройство в редакторе цели привязки
- Выберите среди действий **График трафика**

6. Пустая цель

Если цель привязки является пустой строкой, привязка просто сбросит значение, возвращенное **Выражением**. Однако такая привязка все равно может использоваться, потому что выражение все же будет рассчитываться и может выполнять полезные дополнительные действия (т.е. может выполнять вызовы функции, которые что-либо выполняют).

12.1.2.2 Выражение привязки

Выражение привязки записывается в языке выражений AtomMind 112. Это значит, оно может включать ссылки 117, которые будут разрешаться, когда рассчитывается выражение.

Среда вычисления 114 выражения привязки:

Контекст по умолчанию 119	Контекст по умолчанию 946 для виджета.
Таблица данных по умолчанию 120	Таблица параметров ввода 949 виджета для выражения привязки виджета. Отсутствует для выражения привязки модели.
Строка по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные. Если привязка активируется событием 73, через среду можно также выйти на следующие свойства события:

Имя переменной	Тип значения	Описание
context	Строка	Полный путь к контексту события.

event	Строка	Имя события.
level	Целое	Уровень 75 события.
time	Дата	Временная метка события.
acknowledgements	Таблица данных	Таблица подтверждений 76 события.
enrichments	Таблица данных	Таблица обогащений 76 события.
value	Таблица данных	Таблица данных 49, содержащая специфичные для события данные.

Ссылки

Выражения привязки в привязках контекста могут включать ссылки 117, указывающие на переменные 61 контекста 955, функции 70 и их поля или свойства.

Дополнительную информацию о формате стандартных ссылок и алгоритме разрешения можно найти в разделе стандартные ссылки 118.

Пример выражения привязки сервера

```
"User: " + {users.admin:childInfo$firstname}
```

Это выражение включает ссылку (включенную в фигурные скобки) и рассчитывается в строку, получающую результат в виде цепочки из строкового литерала `"User: "` и строки, содержащейся в поле `firstname` переменной `childInfo` контекста пользователя `admin` (путь этого контекста - `users.admin`). Если имя Администратора Чарли, это выражение будет рассчитываться как `User: Charlie`.

12.1.2.3 Активатор привязки

Активатор привязки - это специальная ссылка 117, которая указывает на переменную контекста сервера, событие контекста сервера, поле переменной контекста сервера. Изменение данной переменной или возникновение данного события вызывает обработку этой привязки.

Когда активатор привязки не указан, ссылки из выражения привязки 738 выступают в роли активатора. В этом случае, не используйте цели привязки 737 в выражениях привязки. Так как результат выражения привязки записан в цель привязки, это может привести к бесконечному циклу обновлений.

Например, возьмем следующую привязку и предположим, что Активатор для нее не задан:

Цель: `users.admin.deviceservers.ds1.devices.thermostat:temperature$temperature`

Выражение:
```
{user.admin.devices.sensor:temperatureField:value} +
{user.admin.devices.sensor:temperatureAdjustmentField:value}
```

Вот список поддерживаемых вариаций синтаксиса активатора:

1. Переменная контекста сервера

```
context:variable
```

Данный активатор запускает привязку, как только меняется переменная под названием `variable` контекста сервера с именем `context`.

Пример: `users.admin.devices.dev1:voltage`

Данный активатор запускается при изменении свойства `voltage` в контексте сервера `users.admin.devices.dev1`.

Только переменные, которые приводят к обновленным 84 событиям, могут использоваться в качестве активаторов.

2. Событие контекста сервера

`context:event@`

Данный активатор запускает привязку, как только событие под названием `event` возникает в контексте сервера с именем `context`.

Пример: `users.admin.devices.access_control_terminal:cardRead`

Данный активатор запускается при возникновении события `cardRead` в контексте `users.admin.devices.access_control_terminal`.

12.1.2.4 Условие привязки

Условие привязки - это выражение, написанное на языке выражений AtomMind 112. Оно рассчитывается первым, когда активируется привязка. Если это выражение результатом имеет false, дальнейшая обработка привязки опускается.

Если условие привязки не заполнено или состоит из пробелов и/или только неэкранированных управляющих символов (например, символ новой строки), условие рассматривается как true.

Среда вычисления 114 выражения привязки:

Контекст по умолчанию 119	Контекст по умолчанию 946 для виджета.
Таблица данных по умолчанию 120	Таблица параметров ввода 949 виджета для выражения привязки виджета. Отсутствует для выражения привязки модели.
Строка по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные. Если привязка активируется событием 73, можно получить доступ к следующим свойствам события через среду:

Имя переменной	Тип значения	Описание
context	Строка	Полный путь к контексту события.
event	Строка	Имя события.
level	Целое	Уровень 75 события.
time	Дата	Временная метка события.
acknowledgements	Таблица данных	Таблица подтверждений 76 события.
enrichments	Таблица данных	Таблица обогащений 76 события.
value	Таблица данных	Таблица данных 49, содержащая специфичные для события данные.

Ссылки

Условия привязки могут включать те же типы ссылок, что и выражения привязки 739.

12.1.3 Привязки таблиц

Формат таблицы данных может также определять одну и более привязки 735. Привязки Таблицы данных похожи на формулы в электронных таблицах (таких как Microsoft Excel). Они определяют отношения между ячейками в таблице. Если изменяется значение в первой ячейке, то значение привязанной ячейки перерассчитывается.

Привязка данных указывает на ячейку в таблице данных т.е. цель привязки, и записывает в нее значение из выражения 112, которое может содержать ссылки 117 на различные источники данных или ячейки таблицы. При изменении такого источника данных, происходит автоматическое вычисление привязки, и новое значение записывается в эту ячейку.

Создание привязок

В большинстве случаев привязки добавляются к формату таблицы путем программирования, т.е. при создании драйвера устройства 518 или написании скрипта сервера 879. См. работа с таблицами данных 1372 для получения более подробной информации.

Однако в некоторых случаях привязки создаются системными операторами с использованием редактора таблицы данных 397. Эти привязки затем используются для изменения таблиц "на лету", без вмешательства оператора, т.е. во время выполнения автоматического действия 87.

Цель привязки

```
variable$field[row]#property
```

Все элементы цели привязки являются дополнительными, кроме **field.** Он указывает на поле в таблице, к которому следует применить изменения.

Если переменная задана, мы только что создали привязку перекрестной таблицы. В данной привязке значения, хранимые в одной Таблице данных, копируются в другую Таблицу данных. Такая привязка будет работать, только если значения обеих переменных были изменены во время выполнения одной GUI процедуры Редактировать свойства 91.

Если `row` не задан, привязка будет применяться к каждому ряду в таблице.

Если `property` не определено, результат оценки записывается в `field`. `Properties` используются для изменения поведения определенного поля во время его редактирования. Вы можете использовать их для условного включения/отключения поля, или для превращения поля в раскрывающийся список. Поддерживаемые свойства:

Свойство	Объяснение
enabled	В данном случае выражение привязки должно иметь результат типа Boolean. Если данное значение является TRUE, будет включено редактирование ячейки, определяемой целью привязки, в противном случае, данная ячейка прейдет в режим "только чтение".
hidden	Выражение привязки также должно иметь результат типа Boolean. Если значение является TRUE, поле таблицы, определенное целью привязки, будет скрыто во всех рядах, и его изменение будет невозможно. Имейте в виду, что невозможно скрыть одну ячейку (т.е. поле в определенном ряду).
choices	Выражение привязки должно иметь результат типа Таблица данных, содержащий список Значений выборки 49, которые будут доступны для ячейки, на которую указывает цель привязки.
options	Выражение привязки должно разрешаться в Строку, которая представляет новые опции редактирования 51 для поля.

Выражение привязки

Выражение привязки может содержать только стандартные ссылки 118. Если ссылка указывает на ячейку в редактируемой таблице, она разрешиться в значение, содержащееся в ячейке на данный момент (которая может быть изменена пользователем, или в неё может быть записано другое значение другой привязкой). Во всех других случаях значение будет взято из контекста сервера, как указано в разделе стандартные ссылки 118.

Среда вычисления 114 привязок таблицы данных:

Контекст по умолчанию 119	Может разниться, см. отдельные случаи обработки привязок Таблицы Данных для получения более подробной информации.
Таблица данных по умолчанию 120	Текущая таблица, т.е. таблица, по которой оцениваются привязки.
Строка по умолчанию 119	Текущая строка, если привязка не относится к отдельной строке и оценивается для каждой строки таблицы отдельно. В другом случае приравнивается к нулю.
Переменные среды 123	Могут разниться, см. отдельные случаи обработки привязок Таблицы Данных для получения более подробной информации.

Пример

```
recipient#enabled = {mailSendingEnabled}
```

Во время обработки привязки, изменение поля адресата (возможно, e-mail адресата) возможно пользователем, только если поле "mailSendingEnabled" типа Boolean содержит значение TRUE (т.е. отправка e-mail сообщений включена).

Если Таблица данных содержит несколько записей, данная привязка будет применяться к каждой записи отдельно.

12.1.4 Производительность привязок

Привязки 814 моделей и виджетов обрабатываются пулами потоков, работающих параллельно. Иногда пул (имеющий определенные пользователем настройки 821) недостаточно большой, и скорость получения обрабатываемых запросов новой управляемой событиями привязки или периодической привязки выше чем скорость обработки.

В этом случае выполнение новой привязки *отклоняется*. Когда это происходит, формируется событие предупреждения Информация 77 в контексте модели 1539. Эти события затем генерируются при других отказах, однако, их скорость ограничена, чтобы предотвратить дальнейшее снижение производительности.

Поэтому, если кажется, что модель ведет себя некорректно и не выполняет привязки, рекомендуется запустить действие Мониторинг относительных событий 109 из данного контекста и проверить его журнал событий для предупреждений об отклонении привязок.

Выполнение периодических привязок

В большинстве случаев, периодические привязки не используются, вместо них должны использоваться привязки при событии. Если выражение привязки при событии посылает некоторые переменные, система достаточно умна для того, чтобы определить, что одно из этих значений было изменено и требуется пересассчет. Периодические привязки должны быть использованы, только если привязка при событии не реагирует на переданные изменения значений.

Преимущества привязок при событии над периодическими привязками:

- Меньшая загрузка ЦП, меньшее использование ввода/вывода диска и сети, так как вычисления выполняются при фактических изменениях значения.
- Незамедлительная реакция на изменения значений. Периодические привязки отражают лишь изменения в конце периода. Уменьшение периода оценки ведет к более высокому потреблению ресурсов.

Распараллеливание привязок

Обычно привязки выполняются в *пуле потоков*, который определяет, сколько параллельных заданий могут быть запущено одновременно.

Пул конфигурируется тремя настройками:

- **Нормальный параллельные привязки**. Эта настойка определяет базовый размер пула, то есть сколько параллельных заданий может быть запущено при нормальных обстоятельствах. Если необходимо начать выполнять больше заданий, чем это число, последующие задачи попадут в очередь для дальнейшего выполнения, когда освободятся потоки выполнения.
- **Максимальная длина очереди необработанных привязок**. Это максимальное количество обрабатываемых заданий привязки, которые могут становиться в очередь. Если очередь заполнена, добавочные потоки обрабатываемых привязок будут добавляться в пул до тех пор, когда не будет достигнут лимит максимальных параллельных привязок.

- **Максимум параллельных привязок**. Это абсолютное максимальное число потоков в пуле обрабатываемых привязок. Если достигнуто максимальное значение и в это время есть больше задач, готовых к выполнению, то их выполнение не состоится или же будет заблокировано до тех пор, когда пулу не будут доступны свободные ресурсы.

Другими словами, пул работает следующим образом:

- Первые обрабатываемые задания привязки обрабатываются потоками базового пула, число их ограничено **параллельно работающими привязками**
- Если все потоки заняты, новые задания обрабатываемых привязок образуют очередь до тех пор, пока длина очереди не достигнет м**аксимальной длины очереди необработанных привязок**
- Если очередь заполнена, новые потоки создаются в пуле до тех пор, пока полное количество потоков не достигнет **максимума параллельных привязок**
- Если все потоки в полностью расширенном пуле заняты, задачи либо отклоняются, либо откладываются, это зависит от контекста

УЧЕТ КОЛИЧЕСТВА ПОТОКОВ

Потоки, которые обрабатывают привязки, обычно достаточно энергозатратные, чтобы создавать и поддерживать их относительно ресурсов системы. Каждый поток получает от RAM от 128 до 512 килобайт для внутреннего *стека* и потребляет определенное количество ресурсов CPU для *переключения контекста*, то есть ядра CPU переключения между разными потоками.

Поэтому обычно количество потоков необходмо удерживать на низком уровне. Практический максимум потоков для современных приборов на серверном уровне около 10-20 тысяч, для автоматизированных приборов это 5-10 тысяч. Однако полное количество потоков в высоко загруженном AtomMind Server должно быть ниже 1000-1500, количество потоков в AtomMind Client, который запускает множество виджетов, должно быть ниже 300-500.

12.2 Правила

Бизнес-правила включают *читаемые машинами базы данных*, помогающие системе принимать решения и производить сложные расчеты во время автономного функционирования.

Правила группируются в наборы правил. Все наборы правил полностью независимы друг от друга. Каждый набор включает одно и более правил, работающих в определенном порядке (см. Типы наборов правил 745). Каждый набор правил может принимать определенные параметры, и его обработка всегда имеет результатом один объект (любого типа, т.е. число, строка или дата).

Компоненты правил

Каждое правило в наборе состоит из цели, выражения и условия. Кажде правило либо возвращает результат всего набора правил, либо (заново) определяет единую переменную среды, действительную до конца обработки набора правил.

Также возможно добавлять комментарии к отдельным правилам.

ЦЕЛЬ ПРАВИЛ

Цель определяет, как обрабатывается результат набора правил. Если цель правила - это **Финальный результат набора правил**, это правило прекращает обработку набора правил и возвращает результат всего набора правил.

В других случаях цель определяет название переменной среды, которое будет определяться, когда закончится обработка правил. На эту переменную можно сослаться из выражений и условий других правил.

ВЫРАЖЕНИЕ ПРАВИЛ

Выражение правил - это выражение AtomMind 112, которое возвращает результат любого типа. Этот результат:

- Возвращается как результат всего набора правил, если Цель правила - это **Финальный результат набора правил**.
- Хранится как переменная внутренней среды набора правил, переписывая значение этой переменной, если оно уже было определено другими правилами. Эта переменная среды будет действительна во время цикла обработки текущего набора правил.

Выражения правил могут ссылаться на результаты других правил из этого набора через ссылки `{env/rule_target_variable_name}`.

УСЛОВИЕ ПРАВИЛ

Условие Правил - это выражение AtomMind 112, которое имеет результатом булевое значение. Если это значение false, обработка правила будет пропущена, и будут обрабатываться другие правила в соответствии с типом набора правил 745.

Обратите внимание, что незаполненные условия или условия, возвращающие NULL, будут иметь значение `true`.

Условия правил могут ссылаться на результаты других правил из этого набора через ссылки `{env/rule_target_variable_name }`.

Условия правил опциональны.

Пример набора правил

Этот пример описывает простой последовательный 745 набор правил, рассчитывающий нагрузку процессора хоста сети независимо от типа хоста и операционной системы.

Этот набор правил используется относительной 811 моделью 810, прикрепленной к каждому хосту сети, и, таким образом, контекстом по умолчанию 119 выражения и условия правил является контекст сетевого устройства, к которому он прикреплен.

Цель	Выражение	Условие	Комментарий
Результат финального набора правил	`aggregate({.:hrProcessorTable}, '{env/previous} + {hrProcessorLoad}', 0) / records({.:hrProcessorTable})`	`hasVariable({.:}, 'hrProcessorTable')`	Standard
Результат финального набора правил	`aggregate({.:cpmCPUTotalTable}, '{env/previous} + {cpmCPUTotal5min}', 0) / records({.:cpmCPUTotalTable})`	`hasVariable({.:}, 'cpmCPUTotalTable')`	Cisco
Результат финального набора правил	`{utilities:statistics({.:}, "hpuxCpuLoad", null, "minute", "false")$average}`	`hasVariable({.:}, 'computerSystemIdleCPU') && hasVariable({.:}, 'computerSystemSysCPU') &&hasVariable({.:}, 'computerSystemUserCPU') && hasVariable({.:}, 'computerSystemNiceCPU')`	HP/UX
Результат финального набора правил	`{.:ssCpuSystem$ssCpuSystem} + {.:ssCpuUser$ssCpuUser}`	`hasVariable({.:}, 'ssCpuSystem') && hasVariable({.:}, 'ssCpuUser')`	Solaris
Результат финального набора правил	`{.:agentCPUutilizationIn1min$agentCPUutilizationIn1min}`	`hasVariable({.:}, 'agentCPUutilizationIn1min') && {.:agentCPUutilizationIn1min$agentCPUutilizationIn1min} != null`	D-Link
Результат финального набора правил	`null`		

Первое правило используется, когда контекст устройства имеет переменную `hrProcessorTable`. Оно проходит через список ядер процессора и рассчитыает среднее.

Второе правило подобным же образом используется, когда у контекста устройства есть переменная `cpmCPUTotalTable`.

Другие правила подобны предыдущим и подходят для устройств HP/UX, Solaris и D-Link.

И, наконец, если не находится подходящих правил, нагрузка процессора считается неопределенной в соответствии с последним безусловным правилом.

12.2.1 Типы наборов правил

Модели поддерживают два типа наборов бизнес-правил:

- Последовательные наборы правил 745
- Зависимые наборы правил 745

12.2.1.1 Последовательные наборы правил

Правила в последовательном наборе правил обрабатываются одно за другим сверху вниз по списку правил.

Правила, чьи условия определяются и возвращают false, пропускаются.

Первое обрабатываемое правило, нацеленное на **Финальный результат набора правил**, остановит обработку набора правил и вернет результат его Выражения как результат всего набора правил.

Если не найдено ни одного правила, нацеленного на **Финальный результат набора правил**, или все такие правила пропускаются из-за Условия false, обработка набора правил прекратится и выдаст ошибку **Результат не определен**.

12.2.1.2 Зависимые наборы правил

Правила в зависимом наборе правил обрабатываются не в том порядке, в каком они появляются в списке правил.

Обработка начинается с выбора всех правил, указывающих на **Финальный результат набора правил**. Сначала рассчитываются условия таких(ого) правил(а), и формируется список активных правил финального результата (т.е., правила, у которых нет условий или условия true):

- Если полученный список правил финального результата имеет больше одного правила, обработка набора правил заканчивается с ошибкой **Найдено несколько активных правил для расчета финального результа**.
- Если полученный список правил финального результата не имеет правил, обработка набора правил заканчивается с ошибкой **Не найдено активных правил для расчета финального результата**.

Если найдено одно активное правило финального результата, рассчитывается его Выражение.

Обработка зависимого правила

Если выражение любого правила ссылается на одну или более переменных среды, оно считается *зависимым* от других правил. Для получения результата таких(ого) правил(а), система находит все активные правила (т.е. с условием no/true), чья цель указывает на эту переменную. Обработка набора правил останавливается, если не найдено ни одного правила или найдено более одного (см. выше). Если найдено одно правило, рассчитывается его Выражение. Это выражение может рекурсивно ссылаться на результат других правил.

12.3 Работа с историческими данными/значениями

AtomMind Server получает значения и события от различных видов устройств и источников данных. Некоторые дополнительные события и обновления значений предоставляются внутренними источниками, такими как модели 810.

Выбранные события и обновления значений постоянно хранятся в БД сервера 692. Заметьте, что в целях унификации AtomMind Server хранит обновления исторических значений в форме событий. Им присвоено имя change 84.

Такие постоянно хранящиеся события и обновления значений называются *историческими данными*.

Работа с историческими данными является важной частью большинства проектов <%AG%. Этот раздел подробно описывает все инструменты и методы хранения и обработки исторических данных.

Обработка исторических данных включает в себя несколько важных аспектов:

- Конфигурация хранилища 746 исторических событий и значений переменных;
- Настройка агрегирования 746 исходных событий/значений и хранение агрегированных данных;
- Получение сырых 747 исторических значений и событий;

- Получение агрегированных 748 исторических данных.

12.3.1 Конфигурация хранилища

По умолчанию, AtomMind Server не хранит события 73 или обновления значений переменных 61 постоянно. Однако, в различных продуктах на базе AtomMind можно активировать хранилище по умолчанию.

Системные администраторы и авторизованные пользователи могут активировать хранилище исторических данных и настраивать периоды хранения для большинства переменных и событий. Однако методы конфикурации различны для различных системных объектов.

Системные настройки хранилища обновлений и событий

Перечисленые ниже методы могут использоваться для настройки хранилища исторических данных для различных контекстов 41:

- Период хранения любого события по умолчанию может быть настроен в таблице правила обработки событий 196. Каждое правило из этой таблицы позволяет определить **истечение срока хранения** события в одном или нескольких исходных контекстах 41.
- Хранилище исторических событий и обновлений может быть дополнительно настроено с помощью таблицы Хранение событий 200. Эта таблица позволяет отправлять исторические события в многочисленные контейнеры для хранения, такие как таблицы БД, а также определить, какие свойства событий должны храниться постоянно.
- Таблица настройки синхронизации по умолчанию 205 позволяет настраивать период хранения по умолчанию для переменных устройства (определяются по своему имени).

Настройки хранения отдельных событий и обновлений

Перечисленые ниже методы могут использоваться для настройки хранилища исторических данных для особых контекстов 41:

- Период хранения событий, полученных от определенного устройства 497, может быть настроен с помощью настройки период хранения событий устройства 511
- Период хранения отдельных переменных устройства может быть настроен при помощи параметров синхронизации настроек устройства 502
- Хранение переменных модели 817 настраивается с помощью настроек **время хранения истории** и **режим записи истории**
- Хранение событий модели 819 настраивается с помощью свойства **время хранения истории**
- Хранение истории датчика 2181 определяется свойством время хранения истории 2183

12.3.2 Настройка агрегации исторических данных

Иногда хранение всего объема сырых событий и обновлений значений провоцирует возникновение слишком большого объема данных для постоянного хранения. В этом случае, можно использовать предварительную агрегацию исторических данных.

В AtomMind Server доступно два метода предварительной агрегации:

- Кольцевая БД
- Гранулы

Агрегация в кольцевой БД

В AtomMind Server встроена кольцевая БД, которая помогает хранить агрегированную статистику 718 временных серий.

Агрегация в гранулах

Гранулы - это особые события, которые содержат различные агрегированные значения для различных периодов во временных сериях. Каждый раз когда сервер получает новое событие/значение, он находит подходящую гранулу и объединяет только что полученное значение с агрегированным значением, вычисляя и непрерывно храня новое агрегированное значение.

Сравнение методов агрегации исторических данных

Метрика	Статистика в кольцевой БД	Агрегированные значения в гранулах
Размер агрегированных данных	Постоянный, ниже	Изменяющийся, выше
Скорость обновления агрегированных данных	Выше	Ниже
Скорость извлечения агрегированных данных	Выше	Ниже
Тип и формат агрегированных значений	Только числовая	Любые
Функции агрегации	Пользовательская	Фиксированные (средние, минимальные, максимальные, итоговое, первое, последнее)
Периоды времени	Периоды фиксированной длительности	Периоды, соответствующие реальным календарным периодам (такие как дни и месяцы), time zones respected
Поддержка агрегации в оперативной памяти	Да	Нет
Встроенная поддержка исходных значений с динамической компенсацией	Да	Нет

12.3.3 Доступ к сырым историческим данным

Независимо от типа базы данных 692, используемой для хранения исторических данных, ядро AtomMind Server предоставляет единый метод для загрузки определенного количества исторических данных.

Однако, существуют разные способы извлечения исторических данных из инструментов обработки данных AtomMind Server. Эти способы описаны ниже.

ПОЛУЧИТЬ ИСТОРИЮ СОБЫТИЯ

Стандартный инструмент для загрузки сырых исторических данных из БД - это функция Получить историю событий 1518 контекста События 1513.

Параметры этой функции позволяют:

- Выбрать маску исходного контекста и имя события для загрузки
- Применить фильтр для загружаемых событий
- Выбрать интервал времени/дат
- Сортировать конечные события
- Задавать максимальное количество событий для загрузки

Все события возвращаются в большой Таблице данных 49, которую можно сразу же использовать в качестве источника данных отчета, алгоритма машинного обучения и т.д..

ПОЛУЧИТЬ ИСТОРИЮ ПЕРЕМЕННОЙ

Функция Получить историю переменной 1613 контекста Утилиты 1613 загружает исторические данные переменной 61 определенного контекста, т.е. она изменяет 84 события, соответствующие истории обновлений этой переменной.

Параметры функции:

- Путь контекста и имя переменной, чья история должна быть загружена
- Интервал времени/дат
- Направление сортировки
- Максимальное количество результатов

Все исторические значения возвращаются в большой Таблице данных 49:

Временная метка	Значение

Таблица может быть использована для:

- Построение запроса 829 историческим значениям
- Построения отчета 928 истории переменной
- Экспорта данных в сторонние системы

Пример выражения 112, которое возвращает историю переменной `temperature`, определенной в устройстве `users.admin.devices.meter`:

```
{utilities:variableHistory("users.admin.devices.meter", "temperature", "2012-05-03
12:00:00.000", "2012-05-04 12:00:00.000")}
```

Агрегация исторических данных при извлечении

Несмотря на то, что сырые исторические данные могут храниться в БД сервера, часто необходимо группировать загруженные значения по временному периоду, вычисляя средние, минимальные или максимальные значения и другие агрегированные показатели для каждого периода времени.

Функция Обобщение 1613 контекста Утилиты 1613 является очень удобным способом агрегации данных при их извлечении. Параметры ввода данной функции позволяют:

- Определить множество серий данных для моментальной загрузки и обработки
- Определить маски исходного контекста и имена событий/переменных, чья история должна быть обработана
- Определить интервал времени/дат
- Определить пользовательское выражение, используемое для извлечения значений для дальнейшей агрегации
- Определить пользовательское выражение, используемое для извлечения временных меток из выборки данных в режиме реального времени
- Определить периоды группировки (напр. часы или месяцы) и временную зону, используемую для квантования времени
- Вычислить различные сводные показатели, такие как средние значения, число выборок за определенный период и т.д.
- Интеллектуально обработать различные значения типа "счетчик"
- Обработать значения вне порядка

Функция Обобщение также возвращает все итоговые данные в большой Таблице данных 49, которая может быть сразу же использована любым другим инструментом обработки данных, таким как наборы правил 813 модели.

12.3.4 Доступ к агрегированным историческим данным

Пути доступа к агрегированным историческим данным различны в зависимости от методов агрегации.

Доступ к статистике в кольцевой БД

Функция Статистика 1616 контекста Утилиты 1613 возвращает агрегированные данные, находящиеся в кольцевой БД.

Функция принимает множество параметров:

- Маска исходных контекстов и имя статистического канала
- Время агрегирования
- Настройки, определяющие, какие агрегированные показатели должны быть возвращены (например, минимальные или суммарные величины)

Данная функция возвращает таблицу данных 49, содержащую агрегированные данные, сгруппированные по временному периоду. Таблица может напрямую использоваться любыми инструментами обработки и визуализации данных.

Похожая функция Сырые статистические данные[1613] контекста Утилиты[1613] возвращает все данные, собранные в специальный канал статистики, указанный путем контекста и именем канала. Возвращенная таблица содержит вложенные таблицы с данными, сгруппированными по временному периоду агрегации.

Вот как выглядит выход функции Статистика[1618]:

Начало периода	Конец периода	Агрегированное числовое значение (Среднее, Минимум или Максимум)

Пример выражения[112], которое возвращает средние, минимальные и максимальные часовые значения для статистического канала `temperature`, определенного для устройства `users.admin.devices.meter`:

```
{utilities:statistics("users.admin.devices.meter", "temperature", null, "hour", true, true, true, true)}
```

12.4 Пригодность ресурсов

Пригодность ресурса - это технология, позволяющая AtomMind Server определить, что некий ресурс "совместим" с какими-то другими ресурсами. Пригодность относится к следующим типам ресурсов:

- Виджеты[943]. Если относительный[946] виджет пригоден для определенного контекста, он может использовать этот контекст как первоначальный источник данных.
- Инструментальные панели[912]. Если относительная[913] инструментальная панель пригодна для определенного контекста, ее можно открыть так, чтобы ее элементы забирали данные из этого контекста.
- Отчеты[928]. Если относительный[932] отчет пригоден для определенного контекста, он может забирать данные из этого контекста для заполнения шаблона во время создания отчета.
- Группы[751]. Если определенный контекст пригоден для динамической группы[753], он будет автоматически добавлен к этой группе.
- Общие таблицы[2176]. Если определенный контекст пригоден для общей таблицы, которая используется, как пользовательское свойство[2177], это пользовательское свойство добавляется к этому контексту.
- Модели[810]. Если относительная модель пригодна для контекста, создается выделенная копия этой модели и прикрепляется к этому контексту, чтобы взаимодействовать с ним.

Настройка пригодности

Пригодность настраивается двумя способами:

- Выражением пригодности[749]
- Правилами обновления пригодности[750]

12.4.1 Выражение пригодности

Выражение пригодности помогает AtomMind Server определить, с какими контекстами[41] может работать ресурс. Оно пишется с использованием встроенного языка выражений[112]. Когда создается новый ресурс или обновляется само выражение пригодности, AtomMind Server вычисляет это выражение для каждого контекста в системе. Если выражение вычисляется как **true**, сервер считает, что ресурс "понимает" данные этого контекста и:

- Для виджетов[943] устанавливает в этом контексте действие запустить виджет[948]
- Для инструментальных панелей[912] устанавливает в этом контексте действие открыть инструментальную панель[913]
- Для отчетов[928] устанавливает в этом контексте действие запустить отчет[933]
- Для групп[751] добавляет этот контекст в группу
- Для общих таблиц[2176] регистрирует эту общую таблицу как пользовательское свойство этого контекста

- Для моделей 810 прикрепляет отдельную копию модели к этому контексту

Пример 1: Выражение пригодности часто используется для проверки типа 43 контекста. Это полезно для создания виджетов, работающих с определенными типами Device или системных ресурсов. Например, чтобы виджет 943 мог работать с каждым устройством SNMP в системе, задайте Выражение пригодности как `{.#type} == 'device.snmp'`. Это выражение разрешается в TRUE, когда для контекста проверяется свойство `type` (обозначенное путем относительного контекста ".") равное `device.snmp`.

Пример 2: Динамические группы 753 используют Выражение пригодности для консолидации устройств определенного типа. Например, чтобы группа могла содержать все устройства типа принтер в системе, задайте Выражение пригодности как `startsWith({.:#type}, 'device') && {.:genericProperties$type} == 'printer'`.. Это выражение разрешается в TRUE, когда проверяемое для контекста свойство `type` (обозначенное путем относительного контекста ".") начинается со слова `device` и поля `type` переменной `genericProperties` (которая содержит выбранный пользователем тип устройства) равного `printer`.

Среда вычисления 114 выражения пригодности:	
Контекст по умолчанию 119	Контекст, для которого проверяется пригодность.
Таблица данных по умолчанию 120	Отсутствует.
Строка по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

ПЕРЕРАСЧЕТ ВЫРАЖЕНИЯ ПРИГОДНОСТИ

Выражение пригодности рассчитывается в следующих случаях:

- При запуске сервера для связывания ресурса со всеми существующими пригодными контекстами
- Когда создается новый контекст для связывания с ним ресурса, если он пригоден
- Когда модифицируется само Выражение пригодности, оно перерассчитывается для всех контекстов в системе, потому что новый набор контекстов станет пригоден в то время, когда другие уже не будут пригодны
- Когда случается событие, указанное правилом обновления пригодности 749

12.4.2 Правила обновления пригодности

Правила обновления пригодности указывают AtomMind Server, когда нужно перерассчитывать выражение пригодности для определенного контекста, чтобы проверить, становится ли он пригоден для ресурса. Каждое правило обновления пригодности состоит из:

- **Маски** контекстов, в которой нужно мониторить события
- Имени **События**, которое проверяется
- **Целевого выражения**, которое, если оно обозначено, указывает на контекст, чью пригодность нужно проверить. Если Целевое выражение не определено (что бывает в большинстве случаев), система проверяет пригодность контекста, в котором случилось **Событие**.

Среда вычисления 114 Целевого выражения:	
Контекст по умолчанию 119	Контекст события, которое запустило правило обновления пригодности.
Таблица данных по умолчанию 120	Данные события, которое запустило правило обновления пригодности.
Строка по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

Пример 1: Предположим, что мы хотим создать отчет 928, который пригоден для всех контекстов устройств 1494, в описании которых есть префикс `Collector_`. Мы можем использовать следующее **Выражение пригодности**:

```
startsWith({info$type}, "device.") && startsWith({info$description},
"Collector_")
```

Выражение пригодности проверяет поля `type` и `description` переменной инфо 63, которая доступна во всех контекстах. Тип контекста никогда не меняется, но его описание может редактироваться в любое время. Если мы хотим добавить отчет к любому контексту сразу после того, как его описание поменялось

на описание, начинающееся с Collector_, нужно добавить следующее **Правило обновления пригодности**:

- Маска: users.*.devices.*
- Событие: infoChanged
- Целевое выражение: <Not Set>

Это правило активируется каждый раз, когда случается событие infoChanged в любом контексте устройства. Это происходит, когда меняется переменная info, т.е. когда происходит изменение описания контекста. Правило вызовет перерасчет Validity Expression для контекста устройства, чье описание поменялось, и если новое описание начинается с Collector_, действие Запустить отчет 933 добавляется к устройству.

Пример 2: Нужно создать инструментальную панель 912, которая может открываться только для устройств, принадлежащих к определенной группе 751. Необходимо использовать следующее **Выражение пригодности**:

```
aggregate({users.admin.devgroups.test:visibleChildren}, "{env/previous} +
({path} == '" + dc() + "' ? 1 : 0)", 0) > 0
```

Это выражение пригодности использует функцию aggregate() для представления всех строк в таблице visibleChildren каждого контекста. Если поле path в текущей строке равно пути контекста, чья пригодность проверяется (возвращеается функцией dc()), этот контекст считается найденным среди членов группы и, таким образом, пригодным. В этом случае функция aggregate() возвращает 1 (в ином случае 0).

Если нужно зарегистрировать инструментальную панель в контекстном меню устройств сразу после их добавления в группу test, нужно добавить следующее **Правило обновления пригодности**:

- Маска: users.admin.devgroups.test
- Событие: visibleChildAdded
- Целевое выражение: {path}

Когда к группе добавляется новый член, в групповом контексте 1529 случается событие visibleChildAdded. **Целевое выражение** ссылается на поле path этого события и, таким образом, **Выражение пригодности** перерассчитывается для новых добавленных членов (указанных полем path этого события). Поскольку оно всегда возвращает true, действие Открыть инструментальную панель 913 инструментальной панели будет добавляться к устройству.

Чтобы убедиться, что инструментальная панель больше не пригодна, когда устройства удаляются из группы, нужно добавить еще одно **Правило обновления пригодности**:

- Маска: users.admin.devgroups.test
- Событие: visibleChildRemoved
- Целевое выражение: {path}

Единственная разница в том, что время события visibleChildRemoved Выражение пригодности вернет как false для контекста, указанного полем path данных события. Поэтому инструментальная панель снимет свою регистрацию с любого устройства, исключенного из группы.

12.5 Группы

Группы используются для сочетания нескольких устройств 497 или других системных ресурсов (фильтров событий 762, тревог 779 и т.д.), чтобы упростить управление большими наборами устройств/ресурсов и сделать выполнение групповых операций более удобным. Гораздо легче выполнить операцию для всех устройств в группе, например, перезапустить все устройства сразу. Группы также обеспечивают простой способ задать похожие настройки для всех составляющих группы и автоматически дублировать между ними изменения в конфигурации.

Документация по теме: Использование шаблонов ресурсов и устройств 1691

Динамические группы

Хотя добавить элементы в любую группу можно путем простого перетаскивания, некоторые группы могут также применять автозаполнение. У динамических групп есть так называемое Выражение пригодности, которое проверяет каждый системный ресурс на пригодность к данной группе. Если ресурс подходит, он автоматически добавляется в группу, и так же автоматически будет удален, как только перестанет соответствовать Выражению пригодности.

Пример: у каждого устройства есть свойство Тип, а некоторые дистрибутивы AtomMind включают динамические группы, содержащие устройства определенных типов, например, принтеры или роутеры.

Вложенные группы

Группы можно поместить внутрь других групп. Уровень вложенности группы не ограничен.

Статус группы

Для наборов устройств и других "динамических" ресурсов группы предоставляют метод расчета агрегированного статуса. Например, группа устройство может получить красный ("Проблема") статус, если хотя бы у одного устройства из группы есть ошибки синхронизации. Если ни у одного устройства нет ошибок, статус группы будет желтым ("Офлайн"), если хотя бы одно устройство из группы отключено, и зеленым ("Нормальным") в противном случае. Это очень гибкий подход.

Мастер значения

Еще одна особенность групп - интеграция с Общими данными 2176, что позволяет иметь "мастер" значения свойств для членов группы. Изменения мастер значения автоматически применяются ко всем участникам группы.

Администрирование группы

Для администрирования группы используются два контекста: общий контекст Группы 1528, который служит контейнером групп определенного типа, и контекст Группа 1529, который представляет одну группу.

12.5.1 Управление членами группы

Новые члены группы могут добавляться в любое время. Максимальное количество составляющих группы не ограничено. При удалении группы ее члены сохраняются.

Объект может принадлежать к нескольким группам.

Примечание для опытных администраторов AtomMind: доступ к членам группы не может быть осуществлен через пути контекста 41, такие как **group_context_path.member_name**. Вместо этого для доступа к ним следует использовать обычные пути членов группы. Например, даже если мы включим пользователя **admin** в группу **usergroup1**, мы не сможем получить доступ, используя путь **usergroups.usergroup1.admin**. Доступ осуществляется через путь **users.admin**.

Добавление членов в группу

Чтобы добавить новых членов группы в пользовательском интерфейсе AtomMind Server, перетащите при помощи мыши контекст 41 в группу. Если контекст подходит для группы, он будет добавлен. Более подробное описание логики данной операции см. здесь 1530.

Удаление членов из группы

В контексте каждого члена группы существует действие 87 **Удалить из группы** (), доступ к которому осуществляется через контекстное меню в системном дереве 370 AtomMind Client или похожие средства других клиентов.

Вложенные группы

Группы поддерживают неограниченное количество вложений, т.е. одна группа может быть членом другой группы, которая, в свою очередь, является членом третьей и т.д. Список членов группы верхнего уровня включает "прямых" членов вместе со всеми членами их подгрупп (вложенных групп) на всех уровнях.

12.5.2 Типы групп

AtomMind Server позволяет объединять все типы устройств и ресурсов: группы пользователей 478, группы Device, группы отчетов 928 и т.д.

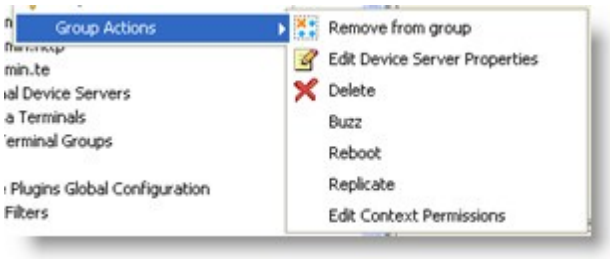
Некоторые типы групп, например, **группа пользователей**, являются глобальными. Их создание, доступ и управление осуществляется только пользователями с правами доступа 477 администратора.

Другие типы групп, такие как группы Device или тревог, принадлежат определенной учетной записи и, таким образом, могут контролироваться как администраторами, так и обыкновенными пользователями (владельцами учетных записей).

12.5.3 Групповые операции

Группы позволяют с легкостью выполнять одну и ту же операцию на нескольких устройствах или ресурсах. Например, меню **Действия группы** для группы Device может иметь действие "Отправить сообщение", используемое для отправки текстового сообщения на ЖКИ (LCD) устройства. Выполняя данное действие, пользователь вводит сообщение только один раз, и затем оно отправляется всем Device в группе.

Меню "Действия группы" внутри группы Device Server выглядит следующим образом (скриншот из AtomMind Client) 359:



12.5.4 Динамические группы

Динамическая группа автоматически поддерживает список ее членов, добавляя контексты, соответствующие выражению пригодности 757, если оно задано. Контексты, которые не соответствуют Выражению Пригодности, убираются автоматически.

См. Пригодность ресурсов 749 для получения более подробной информации.

Возможно также и добавление членов в динамическую группу вручную. Однако нельзя удалить из группы динамически добавленные члены.

Динамические группы включают только членов, доступных для владельца группы в соответствии с его правами доступа 477.

Пример: Выражение пригодности: `startsWith({.:#type}, 'device.') && {.:genericProperties$type} == 'printer'`

Правила обновления:

1. **Маска** = `users.*.devices.*`, **Событие** = `infoChanged`
2. **Маска** = `users.*.devices.*`, **Событие** = `updated`

Динамическая группа включает все сетевые принтеры, т.е. устройства 497, чья строка типа контекста (определенная ссылкой `.:#type`) начинается с `device.` и тип устройства (определенный полем `type` переменной `genericProperties`) обозначен как `printer`. Каждый раз, когда тип контекста или устойства меняется (это будет обозначено событиями `infoChanged` или `updated`), устройство для группы пересматривается.

12.5.5 Статус группы

Любая группа, в качестве дополнения, может иметь статус, который динамически рассчитывается как функция состояния члена группы. Статус отображается цветом узла группы в системном дереве 370.

Например, группа Device выделяется красным цветом, если хотя бы одно устройство сообщает об ошибке, желтым, если какое-либо из устройств находится офлайн, зеленым - в остальных случаях.

Расчет статуса группы

Расчет статуса группы происходит по следующим правилам:

1. Каждый раз, когда добавляется новый член группы или меняется **переменная статуса** уже существующего члена, для него рассчитывается **выражение статуса**, которое превращается в *строку статуса*.
2. Все строки статусов членов группы просматриваются в **таблице статусов** сверху вниз.
3. Статус группы устанавливается согласно *самой верхней* записи таблицы статусов, когда **результат выражения статуса члена группы** совпадает хотя бы с одной строкой статуса.

Предположим, что у нас есть группа из пяти членов. Выражение каждого члена оценивается в строковое выражение, которое обычно равно 100, 200, 300, 400 или 500 (это строковые константы, а не цифры). Таблица статусов будет выглядеть следующим образом:

Результат выражения статуса члена группы	**Статус группы (Цвет)**
100	Зеленый
200	Желтый
300	Красный
400	Малиновый
500	Синий

Приведенная ниже таблица показывает, в какой статус перейдет группа согласно различным комбинациям статусных строк членов группы и приведенной выше таблицы:

Строки статуса члена группы	**Статус группы (Цвет)**
100, 200, 300, 400, 500	Зеленый
500, 400, 300, 200, 100	Зеленый
300, 300, 300, 400, 500	Красный
500, 500, 500, 500, 500	Синий
500, 500, 200, 500, 500	Желтый
200, 200, 500, 200, 200	Желтый
100, 200, 300, 400, 123	**Не определен**

Этот пример иллюстрирует, как рассчитывается статус по умолчанию группы устройств. Настройки статусов групп по умолчанию следующие:

Переменная статуса члена: `contextStatus`

Выражение статуса члена: `({status} % 10 != 2 && ({status} - {status} % 10) == 40 ? 1 : ({status} % 10 == 1 || {status} % 10 == 2 ? 3 : 2))`

Таблица статусов:

Результат выражения статуса члена	Цвет	Описание
1	Темно-красный	Ошибка
2	Темно-желтый	Офлайн
3	Темно-зеленый	Онлайн

Конфигурация статуса вышеобозначенной группы использует информацию о статусе контекста[1495] устройств, чтобы рассчитать статус группы устройств. Существует три определенных статуса группы устройств: `Error` (означает, что как минимум одно устройство сообщает об ошибке), `Offline` (означает, что ошибок нет, но как минимум одно устройство находится в офлайн или состояние его подключения неизвестно) и `Online` (означает, что все устройства в группе подключены и не сообщают об ошибках).

Выражение статуса члена анализирует поле **статуса** с целым числом переменной **contextStatus**. Этот статус можно интерпретировать как:

- Количество десяток, представляющих статус синхронизации[516] устройства (20 = синхронизировано, 40 = ошибка синхронизации и т.п.)
- Количество единиц, представляющих статус соединения[516] устройства (0 = офлайн, 1 = онлайн, 2 = в режиме ожидания, 3 = статус неизвестен)

Выражение статуса отдельного члена группы устройств имеет следующий результат:

- 1 (Ошибка), если устройство не находится в состоянии ожидания (`{status} % 10 != 2`) и выдает ошибку синхронизации (`({status} - {status} % 10) == 40`)
- 3 (Нормальный), если устройство находится в состоянии ожидания или онлайн (`{status} % 10 == 1 || {status} % 10 == 2`)
- 2 (Офлайн) в ином случае

См. статью Конфигурция статуса группы[757] для получения более подробной информации.

12.5.6 Репликация свойств члена группы

Группы также позволяют вам копировать настройки между членами группы вручную или в автоматическом режиме. В ручном режиме некоторые или все настройки могут быть реплицированы от одного члена группы ко всем другим. В автоматическом режиме репликации группа контролирует изменения конфигурации и, при обнаружении таковых, копирует их для всех членов группы. Данное свойство поддерживает внутреннюю синхронизацию всех членов группы.

Репликация в ручном режиме

Каждый контекст группы[1529] имеет действие Копировать в дочерние контексты[111], используемое для копирования настроек от одного члена группы ко всем другим. Данное действие помогает осуществлять одинаковую конфигурацию всех членов группы.

Репликация в автоматическом режиме

Группы также поддерживают *автоматическую репликацию*. Включить автоматическую репликацию можно, установив флажок "Включить автокопирование"[756], контроль над ней осуществляет свойство "Опции копирования[758]". Данное свойство представляет собой таблицу, содержащую список свойств (переменных[61]), появляющихся в контекстах членов группы. Можно включить или отключить авто-репликацию для каждого свойства в отдельности.

Если для определенного свойства включена автоматическая репликация, изменения его выражения в одном из членов группы автоматически применяются ко всем другим членам. Например, если в состав группы **G1** входят три

Device (**A**, **B** и **C**), а устройство **B** отключено (например, в результате изменения свойств в AtomMind Client), Device **A** и **C** будут также автоматически отключены. Автоматическая репликация применима только внутри одиночных групп: если устройство **C** является членом сразу двух групп (**G1** и **G2**) и было отключено во время автоматической репликации в группе G1, другие члены группы G2 не будут отключены. Это предотвращает эффект "домино", когда одно изменение в реплицированной группе приводит к неожиданному хаосу.

Когда внутри группы с автоматической репликацией обнаруживается изменение свойства какого-либо из ее членов, значение данного свойства (полученное из Таблицы данных 49, содержащей значение переменной 61 контекста 41) устанавливается для такого же свойства других членов группы при выполнении операции Интеллектуальное копирование таблицы данных.

12.5.7 Маски контекста членов группы

Часто бывает необходимо выбрать все члены группы для какой-либо обработки с использованием маски контекста 44. Далее приведено несколько примеров:

- Создание тревоги 779, применимой к членам группы
- Выполнение запроса 829 на выборку некоторых данных из контекстов членов группы
- Конфигурация фильтра событий 762 для отображения только тех событий, которые были сформированы устройствами в составе определенной группы

Для создания маски, которая соответствовала 45 бы всем членам группы, в конце контекстного пути группы добавьте групповой символ (".*"), например: `users.admin.alerts_groups.snmpAlerts.*`

Пример: Если администратор по умолчанию 479 владеет группой устройств под названием `environment`, включающую все устройства мониторинга среды ЦОД (Центра Обработки Данных), используйте следующую маску для соответствия всем Device в группе:

```
users.admin.devgroups.environment.*
```

Данная маска будет разрешаться в список устройств внутри группы, например:

```
users.admin.devices.temperature_sensor
users.admin.devices.humidity_sensor
users.ny_datacenter_operator.devices.smoke_sensor
users.ny_datacenter_operator.devices.spill_detector
```

12.5.8 Конфигурация группы

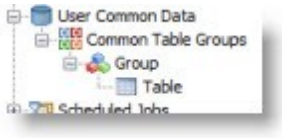
Данный раздел посвящен свойствам конфигурации группы.

Все описанные далее свойства доступны при выполнении действия Настроить 105 в контексте группы 1529.

12.5.8.1 Свойства группы

Данное свойство определяет основные опции группы.

Описание поля	Имя поля
Имя группы. Имя контекста группы, требуемое для ссылки на данную группу из различных частей системы. Должно соответствовать соглашениям о наименованиях 42.	name
Описание группы. Текстовое описание группы. Также является описанием 43 контекста группы.	description
Включить автоматическое копирование. Включает автоматическое копирование 755 для данной группы.	autoReplication
Скрывать членов группы в их основном местоположении. Если установлен данный флажок, контексты членов группы показываются внутри группы, а не под контекстом-родителем. Это способ схематического представления влияет только на видимое представление контекстного дерева сервера (например в Системном дереве 370).	hideMembers

Компоненты, показывающие настоящее (несхематичное) контекстное дерево (например, Селектор объектов 402) в данном случае будут отображать контексты членов группы под контекстом самой группы. Члены группы не скрыты из основного местоположения:  Члены группы скрыты из основного местоположения:  Это свойство можно использовать, чтобы убрать деактивированные контексты из системного дерева, не удаляя 106 их окончательно. Создайте группу, активируйте для нее данное свойство и переместите деактивированные контексты в эту группу. Контексты будут удалены из их изначального местоположения в системном дереве, но будут доступны, если в будущем вы захотите их восстановить. Если контекст имеет более 1000 дочерних контекстов, работа с ним в различных UI компонентах AtomMind может привести к спаду производительности. В этом случае очень рекомендуется распределить дочерние контексты по нескольким группам и включить для них опцию Скрывать членов группы в их основном местоположении.	
Показать количество членов. Если этот флажок активирован, количество членов группы будет показано рядом с описанием группы в круглых скобках.	showMemberCount
Выражение пригодности. Определяет, какой контекст (контексты) может (могут) быть автоматически добавлен (добавлены) в группу. Для более подробной информации см. раздел Динамические группы 753 и Пригодность ресурсов 749.	validityExpression
Правила обновления пригодности. Список масок контекста и имен событий. Если событие, определяемое полем **Событие** данной таблицы, возникает в любом контексте, совпадающем с маской, заданной в поле **Маска** той же записи, **выражение пригодности** будет пересчитано для данного контекста. Данная функция используется для добавления/удаления определенного контекста из группы при возникновении изменений внутри нее.	validityListeners

Данное свойство доступно через переменную childInfo 1532.

12.5.8.2 Статус группы

Данное свойство определяет правила оценки статуса группы 754.

Описание поля	Имя поля
Разрешить пользовательские статусы. Флажок, контролирующий, включен ли пользовательский статус для группы.	enabled
Переменная статуса члена группы. Свойство (переменная) члена группы, которое будет использоваться для расчета статуса каждого члена группы.	variable
Выражение статуса члена группы. Данное выражение будет оцениваться для каждого члена группы с целью обнаружения значения его статуса (строки). Поиск данного выражения будет осуществляться в **таблице статусов.**	expression

Среда вычисления 112 выражения статуса члена группы:	
Контекст по умолчанию 119 — Контекст данного члена группы.	
Таблица данных по умолчанию 120 — Текущее значение **переменной статуса члена группы**.	
Ряд по умолчанию 119 — 0	
Переменные среды 123 — Только стандартные 123 переменные.	
Таблица статусов. Таблица поиска, используемая для расчета общего статуса группы, как было описано в главе Статус группы 754. Данная таблица состоит из трех полей: • **Результат выражения статуса члена группы.** Результат оценки **выражения статуса члена группы.** • **Цвет**. Цвет, используемый для выделения группы с определенным статусом (двенадцать заданных заранее цветов отображения статусов группы). • **Описание**. Описание статуса.	statuses

Данное свойство доступно через переменную groupStatus 1532.

12.5.8.3 Опции репликации

Данное свойство определяет параметры репликации 755, выполняемой группой для каждой настройки ее составляющих.

Описание поля	Имя поля
Свойство. Имя переменной 61 (свойства), к которой применяется правило репликации.	variable
Описание. Описание вышеуказанного свойства.	description
Автоматическое копирование. Использование автоматического копирования для данной переменной.	replicate
Использовать общую таблицу. Для данного свойства используйте заданное значение 759. Включить данную настройку можно только при отключенном **автоматическом копировании** для данного свойства и если в **общей таблице** имеется **заданное значение** (т.е. при наличии **общей таблицы с заданным значением**).	useMaster
Текущая общая таблица с заданным значением. Описание и местоположение 2177 общей таблицы, которая может быть использована в качестве заданного значения свойства. Данное поле доступно только для чтения.	master

Опции репликации доступны через переменную replication 1532.

12.5.8.4 Местоположение

Это свойство определяет местонахождение (географическую точку) этой группы устройств. Оно указывает на переменную местоположения 1529 в контексте этой группы устройств и доступно только для групп устройств.

12.5.8.5 Топология

Это свойство 1826 описывает логические или физические связи с устройствами или группами устройств и доступно только для групп устройств.

12.5.9 Безопасность групп

Только те члены группы, которые доступны для владельца группы, могут добавляться к группе.

Пример: Если устройство **Dev1** недоступно для группы **Grp1** пользователя **Joe** в соответствии с его таблицей прав доступа 477, даже администратор по умолчанию 479 не сможет поместить **Dev1** в **Grp1**.

Группы могут извлекать различные данные из контекстов их членов (например, во время оценки выражений статуса группы 754). Таким образом, у владельца группы должен быть необходимый уровень прав доступа 486 во всех контекстах членов группы.

Права доступа динамической группы

Динамические 753 группы включают только членов, которые доступны владельцу группы в соответствии с его правами доступа.

Заданные значения свойств членов группы

Любая общая таблица 2176, используемая как заданное значение 759 свойств членов группы, должна быть доступна владельцу группы.

12.5.10 Производительность групп

Статические группы - это пассивные компоненты сервера, которые не вызывают значительной нагрузки на сервер или использования памяти.

Динамические 753 группы редко вызывают значительную нагрузку процессора, если переоценка Выражения Пригодности группы часто запускается Правилами Обновления Пригодности группы. См. производительность выражения 141 для оценки влияния на производительность.

Пример: Если динамическая группа включает 100 устройств и ее Правила Обновления Пригодности ссылаются на событие устройства, запускаемое каждым устройством на частоте 10 Гц, Выражение Пригодности оценивается как 1000 раз в секунду.

12.5.11 Заданные значения свойств членов группы

Взаимодействие групп с общими данными 2178 позволяет получать заданные значения свойств членов группы. При изменении заданного свойства происходят автоматические изменения данного свойства всех членов группы.

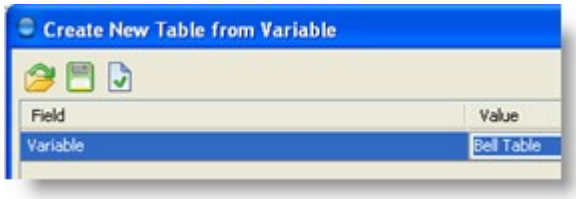
Чтобы создать заданное значение для определенного свойства, нужно:

- Сначала создать общую таблицу, в которой будет храниться заданное значение, используя действие создать новую таблицу из переменной 1483 любого контекста общих данных. Переменная источника для создания общей таблицы должна быть переменной одного из членов группы. Ее значение будет скопировано для всех составляющих группы, когда она включит свойство "заданное значение". Общая таблица должна располагаться в контейнере 2177 пользовательских общих данных владельца группы или в контейнере глобальных общих данных.
- Затем включить настройку **Использовать общую таблицу** в опциях репликации 758 для записи, которую вы хотите реплицировать. Если общая таблица была правильно создана при выполнении предыдущего шага, настройка **Текущая общая таблица с заданным значением** покажет описание и местоположение заданного значения:

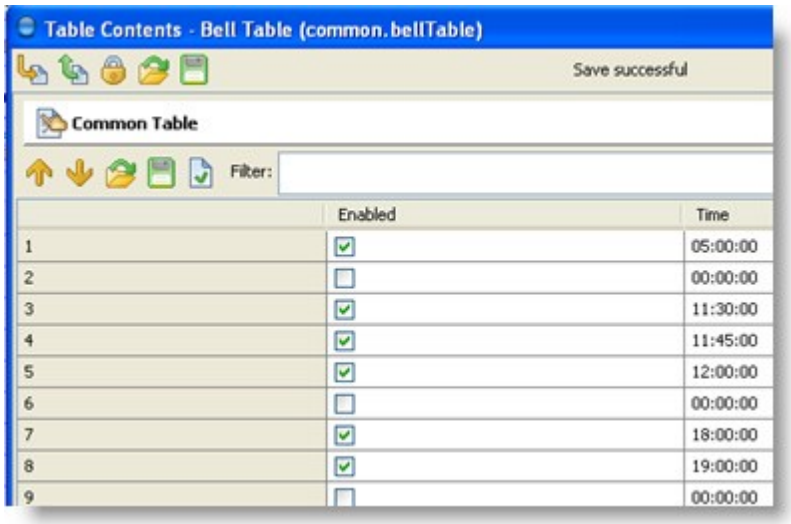
Если в контейнерах глобальных и пользовательских общих данных были созданы общие таблицы, которые могут быть использованы в качестве заданного значения для свойства, будет использоваться пользовательский контейнер.

Пример:

Предположим, что ваше Device - регистратор времени с настройкой "Таблица звонка", определяющей несколько ежедневных тревог. Вы можете перетащить с помощью мыши регистратор времени в глобальные общие данные[1483] в AtomMind Client, запустив действие **Создать новую таблицу из переменной**. Выберите "Таблица звонка":

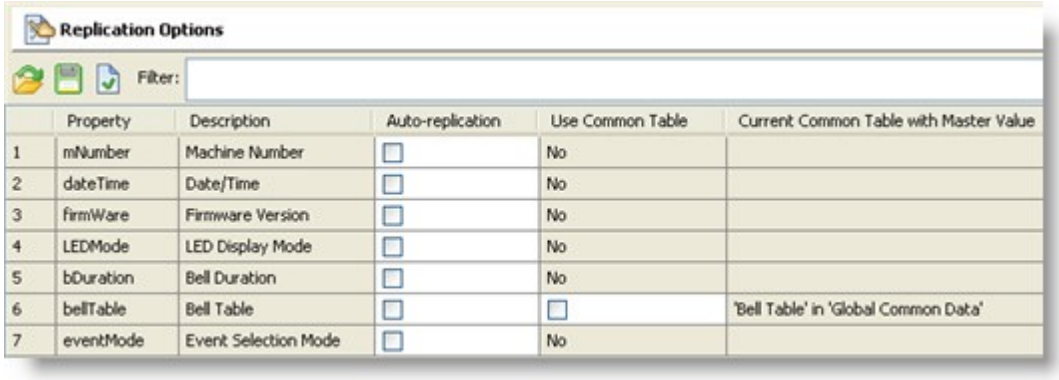


Как только переменная выбрана, в глобальном контейнере создается новая общая таблица. Она содержит список тревог, определенных в перемещенном Device:



Теперь вы можете создать новую группу Device и включить в нее несколько регистраторов времени:

Затем запустите действие свойства группы[1529] для настройки группы. Откроется диалоговое окно настроек группы. Далее приведен скриншот опций репликации:



Настройка **Использовать общую таблицу** может быть включена для **Таблицы звонка**, потому что соответствующее заданное значение находится в контейнере глобальных общих данных. Включите данную настройку и сохраните изменения.

Теперь группа контролирует изменения Таблицы звонка, расположенной в общих данных. Если кто-либо вносит изменения в таблицу, обновленное время звонков сохранится для всех регистраторов времени в группе.

12.5.12 Группы в распределенной архитектуре

Этот раздел описывает особенности поведения групп в распределенной инсталляции[1332] AtomMind.

Если группа из сервера-поставщика присоединяется к дереву контекстов на сервере-потребителе, пользователи сервера-потребителя могут добавить эту группу как подгруппу к другим группам сервера-потребителя. Если группа локального сервера-потребителя имеет пользовательский статус[754], рассчитанный согласно статусам ее членов, включая членов вложенных подгрупп, логика расчета статуса будет отличаться для локальных и удаленных подгрупп:

- В локальной подгруппе каждый член подгруппы обрабатывается напрямую группой верхнего уровня. Группа верхнего уровня рассчитывает **Выражение статуса члена группы** для членов локальных подгрупп и выставляет их индивидуальные статусы в таблице статусы членов группы[1533].
- В удаленной подгруппе обрабатывается только объединенный статус подгруппы, полученный от сервера-поставщика. Члены удаленной подгруппы не обрабатываются отдельно. Объединенный статус подгруппы добавляется к таблице статусов членов группы верхнего уровня и рассматривается во время оценки ее статуса.

Данные правила по сути означают, что если удаленные группы помещаются в локальные группы с динамическим статусом, правила оценки статуса локальных и удаленных групп должны совпадать с целью избежания ошибок на сервере-потребителе.

12.6 Управление событиями

Управление событиями - это технология управления большим количеством событий и определение тех, которые действительно важны. Этот раздел описывает, как события, полученные из различных источников и сгенерированные в системе, могут управляться операторами и администраторами при помощи AtomMind.

Управление событиями - это сложная задача, вовлекающая множество этапов:

- Фильтрация событий[761]. На этом этапе события, которые не удовлетворяют определенному критерию (источник, критичность, специфичные для домена правила), отфильтровываются перед обработкой и бизнес-представлением.
- Агрегирование событий[762]. Агрегирование, также называемое **дедупликацией** или **уменьшением количества** событий, позволяет системе минимизировать общее количество обработанных событий путем объединения экземпляров, у которых есть соответствующие *идентификаторы дедупликации*.
- Маскировка событий[762]. Маскировка означает игнорирование событий, которые исходят из источников, зависящих от системных элементов, давших сбой.
- Корреляция событий[762]. Движок корреляции находит простые отношения между подобными событиями, обычно одно из них отмечает начало определенного процесса или состояния, а второе отмечает его прекращение.
- Анализ первопричин[762]. Это самый сложный этап процесса управления событиями. Он вовлекает анализ отношений между событиями и их средой, а также поиск причин каждого события.

12.6.1 Фильтрация событий

Фильтрация событий представляет собой отказ от событий, которые являются неподходящими. Например, некоторые неприоритетные устройства бывает сложно настроить и периодически отправлять события в AtomMind (такие как "принтеру P нужна бумага A4 в лоток 1").

Существует несколько способов фильтрации ненужных событий в AtomMind Server:

1. Многие драйвера устройств[518] и плагины[207], способные получать события от различных устройств, предлагают различные механизмы пропуска определенных событий или снижения уровня их важности. Например, плагин **Syslog** имеет специальную **Таблицу перевода уровня важности**, позволяющую тонко настроить события AtomMind, генерируемые при получении Syslog-сообщений.
2. Определенные события могут блокироваться для обработки, хранения и маршрутизации цепочек путем настройки **префильтра** в правилах обработки событий[196].
3. Также возможно отфильтровать определенные события на уровне представления. Во-первых, фильтры событий[762] позволяют увидеть только подходящие события во время мониторинга событий и просмотра их истории в реальном времени. Во-вторых, большая часть системных средств, вовлеченных в обработку событий, позволяет определить другой критерий фильтрации, от простого (такого как временные метки начала/конца и минимальные уровни) до гибкого (на основе выражений[112]).

12.6.2 Агрегирование событий

Агрегирование событий (также известное как дедупликация или уменьшение количества событий) представляет собой слияние дупликатов одного и того же события. Такие дупликаты могут появляться в связи с нестабильностью сети передачи (например, одно и то же событие отправляется дважды источником события, потому что первый экземпляр не был быстро подтвержден, но оба экземпляра в конечном итоге достигают места назначения события). Другой пример - это временная агрегация, когда одно и то же событие отправляется снова и снова источником события до тех пор, пока проблема не решится.

Правила обработки событий 196 AtomMind Server позволяют определить **выражение идентификатора дедупликации**, которое будет использоваться для поиска дупликатов события путем сопоставления *идентификаторов дедупликации событий*. Такие события дедупликации сольются, и агрегированное событие сохранит общее количество дупликатов вместе с временными метками самых последних дупликатов.

Другое средство агрегирования событий обеспечивается триггерами события 787 тревоги. Эти триггеры позволяют создавать агрегированное событие (событие тревоги 790), только если исходное событие произошло более N раз в пределах определенной временной рамки.

12.6.3 Маскировка событий

Маскировка событий (также известная как топологическая маскировка) представляет собой игнорирование событий, относящихся к системам ниже неисправной системы. Например, сетевые сервера, находящиеся ниже неисправного маршрутизатора, откажут в опросе.

В AtomMind каждое устройство 497 имеет выражение зависимости 512, предотвращающее его синхронизацию и опрос путем перевода в состояние ожидания, когда это выражение разрешается в FALSE. Таким образом, вы можете сделать ваши устройства зависимыми от других устройств или ресурсов, предотвращая создание событий для любого устройства, автоматически отключенного в соответствии с его зависимостями.

12.6.4 Корреляция событий

Корреляция событий позволяет AtomMind Server находить простые отношения между двумя подобными событиями, одно из которых является инициатором определенного процесса (или состояния), а второе предотвращает этот процесс.

Корреляция событий производится **коррелеторами** или триггерами событий 787 тревоги. Каждый триггер активирует тревогу при получении первого события (названного **начальное событие**) и деактивирует его при получении другого события (**коррелированного события**). Оба события заранее настраиваются в соответствии с выражением 112 триггера, позволяющего активировать и деактивировать тревогу при получении событий одного и того же типа с разными параметрами.

12.6.5 Анализ первопричин

Анализ первопричин - последний и самый сложный этап управления событиями. Он представляет собой анализ зависимостей между событиями, основанными на экземпляре модели среды и графиках зависимости, для определения того, могут ли некоторые события быть объяснены другими. Например, если база данных D запущена на сервере S и этот сервер долгое время является перегруженным (загрузка процессора длительное время равна 100%), событие “договор об уровне обслуживания для базы данных D более не выполняется” может быть объяснено событием “сервер S долгое время перегружен”.

Анализ первопричин при помощи AtomMind осуществляется путем настройки продвинутых тревог 779. Тревога, являющаяся сама по себе событием 790, всегда вызывается определенным событием (если использовать триггеры события 787) или состоянием (если использовать триггеры переменной 782). Эта причина ассоциируется с событием тревоги и постоянно хранится, позволяя находить первопричину любой проблемы во время мониторинга событий в реальном времени или анализа истории событий.

12.7 Фильтры событий

Мониторинг событий в реальном времени является очень важной частью большинства приложений управления удаленными устройствами. Это может применяться в системах защиты, медицинских приложениях, при исследовании ошибок IT оборудования и т.д. Device обычно формируют много различных событий 73, и во время запуска системы, в состав которой входят много Device, ваш AtomMind Server получает события, которые могут быть не очень важными, но с другой стороны, некоторые события могут быть крайне важны. Для операторов системы необязательно контролировать все события, но некоторые из них требуют внимания. Чтобы отделить менее важные события и сконцентрироваться на необходимых, используйте Фильтр событий. Фильтры помогают контролировать различные события системы, такие как выполнение работ по графику 823, эскалация тревоги 779 и т.д.

Фильтр событий используется для скрытия незначительных событий и выделения наиболее важные. Существует набор правил для компонента Журнал событий в Пользовательском интерфейсе AtomMind Server (например, журнал событий 398 в AtomMind Client), позволяющих настраивать визуализацию входящих событий, что позволяет облегчить работу пользователю.

События могут быть отфильтрованы по:

- Ресурсу (устройство, группа устройств, ресурс системы или ресурсы группы)
- Типу события
- Уровню события (severity)
- Параметрам, подтверждениям или обогащениям
- Любым пользовательским критериям, определенным выражением

Фильтрование по выражению делает фильтры событий достаточно гибкими. Далее следуют примеры, как это может помочь:

- Нахождение выполняемых событий в особом диапазоне даты и времени
- Нахождение событий входа особого пользователя (то есть фильтрация по имени пользователя)
- Нахождение всех событий, которые содержат определенную подстроку в любом поле данных
- Нахождение чтения всей температуры, собранной в момент, когда она была выше 120 градусов
- Нахождение событий, совпадающих с условием X и/или условием Y, а также с более сложными комбинациями

В добавление к выбранным критериям фильтры включают в себя правила визуализации списка событий:

- Видимость базовых параметров события, то есть источник, тип, уровень и подтверждение
- Видимость индвивидуальных полей, указывающих на события
- Правила выделения, основанные на выражении и уровне

У каждого сервера, помимо фильтров на уровне браузера, есть дофильтровые правила, позволяющие удалить определенные события до того, как они будут сохранены в базе данных или отправлены в любое направление.

На фильтр может быть наложен параметр, чтобы запрашивать у оператора определенные параметрны каждый раз, когда он активируется.

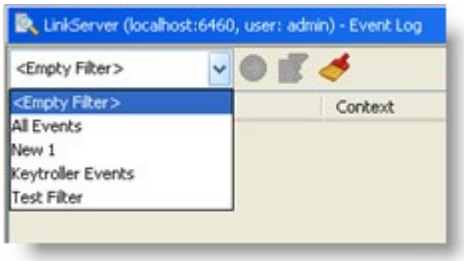
Вертикальные рыночные продукты, основанные на AtomMind, включают в себя наборы встроенных фильтров для просмотра отраслевых событий системы, событий устройства, тревоги и т.д.

У каждого пользователя 478 свой набор правил для фильтра событий.

Создание фильтров журнала событий, работающих с большим количеством данных, может стат причиной высокого расходования памяти AtomMind Server.

Работа с фильтрами событий

Для того, чтобы начать пользоваться фильтром событий, пользователю необходимо *активировать* его, выбрав из контекстного списка в компоненте Журнал событий. Выбор фильтра в Журнале событий AtomMind Client выглядит таким образом:



Компонент журнала событий состоит из двух разделов:

- События реального времени. Показывает только что выполненные события.
- История событий 73, показывает прошлые события.

Можно выбрать отдельные фильтры для каждого раздела. Если выбран фильтр в разделе "События реального времени", Журнал событий начинает прослушивать все категории событий, указанные в таблице правил фильтра 769. Если выбран фильтр в разделе "История событий", AtomMind Server загружает все события, удовлетворяющие списку правил фильтра 769 (см. ниже) в базе данных 692, и показывает первые несколько строк, позволяя пользователю прокрутить весь список и выбрать необходимые события.

Наиболее важным свойством фильтра событий являются правила фильтра 769, определяющие, какие именно события показывать. Во время активации фильтра, обрабатывается каждая **включенная** запись в таблице Правил фильтра, и в журнале отображаются события, соответствующие следующим условиям:

- Контекст 41, в котором было выполнено событие, должен соответствовать 45 **Маске контекста**.
- Если **Имя события** не равно "*****" (Все события), то оно должно совпадать с полем **Имя события**.
- Уровень 73 события должен быть больше или равен уровню серьезности ошибки, обозначенному в поле **Уровень.**
- Событие должно удовлетворять 765 **Выражению фильтра**.

Все поля, имена которых выделены выше **жирным** шрифтом, описываются далее. Каждая запись фильтра может иметь цвет выделения, который будет в дальнейшем использоваться для выделения событий данной категории в Журнале событий.

Цвет выделения, заданный в записи Правила фильтра, может быть изменен при помощи правил пользовательского цветового выделения 768.

В журнале событий представлены по умолчанию следующие столбцы:

- **Временная метка сервера.** Дата и время, когда AtomMind Server зарегистрировал данное событие. Этот столбец нельзя скрыть.
- **Контекст**. Описание и/или путь к контексту, где произошло событие. Данный столбец показывает описания контекста по умолчанию. Пути контекста будут показываться, только если включена настройка **Показывать путь контекста вместе с описаниями** в информации о фильтре 769. Данный столбец может быть скрыт, для этого нужно отключить настройку **Имя контекста** в основных видимых полях 770.
- **Событие**. Имя события и/или описание. Данный столбец показывает описания событий по умолчанию. Имена событий показаны, только если включена настройка **Показывать имена событий вместе с описаниями** в информации о фильтре 769. Данный столбец может быть скрыт, для этого нужно отключить настройку **Имя события** в основных видимых полях 770.
- **Уровень**. Уровень события. Данный столбец может быть скрыт, для этого нужно отключить настройку **Уровень события** в основных видимых полях 770.
- **Данные**. Представление в виде строки в таблице данных, связанной с событием. Этот столбец может показывать имена полей и значения или только значения, в зависимости от настройки **Показывать имена полей** в основных видимых полях 770.
- **Подтверждение**. Подтверждение 73 события. Данный столбец может быть скрыт, для этого нужно отключить настройку **Подтверждение** в основных видимых полях 770.

Для более подробной информации о мониторинге событий смотрите статью Журнал событий 398 в руководстве по эксплуатации AtomMind Client.

Администрирование фильтров событий

Для администрирования фильтров событий используются два контекста: общий контекст фильтров событий 1507, который служит в роли контейнера, и контекст фильтров событий 1509, который содержит информацию о конкретном фильтре.

Структура фильтра событий

Каждый фильтр событий обладает несколькими свойствами:

- **Информация о фильтре.** Имя и описание фильтра, а также другие базовые настройки.
- **Правила фильтра**. Список событий (точнее, типов событий, т.к. мы не обсуждаем здесь особые свойства событий), которые отображаются в Журнале событий после активации фильтра.
- **Основные видимые поля**. Видимые свойства событий, показываемые, если фильтр активирован.

- **Дополнительные видимые поля.** Дополнительные поля, содержащие данные, относящиеся к событию. Вы можете использовать данное свойство для выбора, какое из этих "дополнительных" полей будет отображаться в Истории события для данного события. Например, событие **Вход в систему (login)** содержит поле **Имя пользователя (username)**, указывающее на пользователя, который вошел в систему. Данное поле используется только для события **Вход в систему --** нет смысла использовать его, например, в событии **Тревога**. Поэтому оно содержится в Таблице данных для события **Вход в систему** (такое поле не предусмотрено для Таблицы данных события **Тревога**). Содержание данного поля зависит от типов событий, выбранных в правилах фильтра 769.
- **Пользовательское цветовое выделение**. Правила для назначения различных цветов для событий 768, когда они отображаются в Журнале событий.

Свойство **Правила фильтра** определяет, какие события будут отображаться, когда фильтр событий активирован, в то время как свойства **Информация о фильтре**, **Основные видимые поля** и **Дополнительные видимые поля** описывают, какие параметры и поля будут отображаться для каждого события. Более подробную информацию о свойствах фильтра смотрите здесь 768.

Выражения фильтра

Выражения фильтра применяются для более точной настройки фильтрации и правил цветового выделения. Для более подробной информации смотрите Выражения фильтра 765.

Выделение событий

События, отображаемы в Журнале событий, могут быть маркированы цветом согласно различным правилам. Для более подробной информации смотрите Цветовое выделение событий 768.

Параметризованные фильтры

Некоторые фильтры могут потребовать от системных операторов дополнительной настройки параметров фильтрации при запуске и перезапуске фильтра. Такие фильтры называются *параметризованными фильтрами*. Для более подробной информации смотрите Параметризованные фильтры 772.

12.7.1 Выражения фильтра

Выражения фильтра помогают точно настроить определенный фильтр, показывая только те события, которые соответствуют определенной формуле, заданной выражением AtomMind 112, или выделить цветом события внутри более широкого списка событий. Данное выражение может включать в себя ссылки 117 на ячейки Таблицы данных 49 события, которые содержат относящиеся к событию данные. Если вы еще не разобрались, что представляет собой Таблица данных события, посмотрите Таблица данных события 73 в разделе События 73.

Например, Таблица данных для события **Вход в систему** содержит поле **Имя пользователя**. Таким образом, при фильтрации для отбора событий **Входа в систему** вы можете использовать выражение фильтра, содержащее ссылку на данное поле, чтобы в дальнейшем настроить фильтр и обрабатывать только события входа в систему пользователя Joe.

Также возможно ссылаться на среду фильтрации 768 и данные из различных контекстов AtomMind Server.

Сейчас вы переходите в "расширенный" раздел данной статьи. Для того, чтобы полностью понять выражения фильтра, вам необходимо подробно изучить ссылки 117 и выражения 112. Данные главы расположены в разделе "Внутреннее устройство системы". Мы рекомендуем пропустить изучение данного раздела, прочитать выше указанные главы до полного понимания ссылок и выражений, а затем вернуться сюда для более глубинного разбора выражений фильтра.

ИСПОЛЬЗОВАНИЕ ССЫЛОК ВНУТРИ ВЫРАЖЕНИЙ ФИЛЬТРА

1. Вы можете включить ссылки в состав "таблицы данных по умолчанию" внутри выражения фильтра. Например, для таких ссылок как `firstname`, нет необходимости задавать определенный контекстный путь и имя переменной (типа `users.admin:childInfo$firstname`). AtomMind Server рассматривает Таблицу данных события в качестве Таблицы данных по умолчанию, и данные ссылки уже указаны в ней.
2. Вы также можете указать ссылки на переменные среды, например, `env/level` (уровень серьезности ошибки события, который не входит в Таблицу данных события, а является свойством события. Более подробную информацию о свойствах событий можно найти в статье События 73 раздела "Внутренне устройство системы"). Полный список переменных среды, определяемых при фильтрации события, можно увидеть здесь 768.

ВЫРАЖЕНИЯ ФИЛЬТРА В ПРАВИЛАХ ФИЛЬТРА

Каждая запись в таблице Правил фильтра 769 может содержать собственное Выражение Фильтра. Это может показаться сложным, потому что таблица правил фильтра *уже является* фильтром сама по себе - она определяет маску, имя и уровень события, и отображаются только те события, которые подходят под данные категории. Но

дополнительное выражение фильтра позволяет вам сделать фильтр *более* подробным и точным. Что, если у вас есть события, которые совпадают с маской, именем и уровнем события, но вам все-таки нужно выбрать *некоторые* из них? В таком случае выражение фильтра будет крайне полезным. Все же пока это может казаться не совсем ясным, но вы можете спутиться ниже и найти пример, чтобы стало более понятно.

Выражение фильтра должно выдавать результат в виде логического значения: TRUE или FALSE. Если значение FALSE, событие пропускается фильтром и не отображается в журнале событий. Если значение TRUE, событие фильтруется и показывается в журнале.

Если в правилах фильтра установлен флажок **Параметризован**, поле Параметризованные данные источника используется для построения выражения фильтра в реальном времени при запуске фильтра. Более подробную информацию можно найти в разделе Параметризованные фильтры 772.

Если выражение фильтра не задано в правилах фильтра, будут отображаться все события, удовлетворяющие опциям **Маска контекста, Имя события** и **Уровень** данной записи.

Пример:

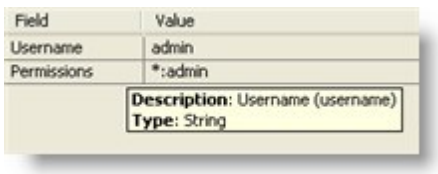
Предположим, что мы управляем входом пользователей в систему. В правилах фильтра мы имеем категорию со следующими настройками:

Поле	Значение
Описание	события входа в систему
Маска контекста	пользователи
Имя события	вход в систему
Включен	TRUE
Уровень	информационный
Выражения фильтра	
Параметризован	FALSE
Цвет выделения	не задан

При двойном нажатии на любое событие входа в систему в журнале мы увидим его данные в Редакторе таблицы данных. Далее приведен скриншот из Редактора таблицы данных 382 в AtomMind Client:

Мы видим, что данные для события **вход в систему** имеют два поля: **Имя пользователя** (аутентифицированный пользователь) 478 и **Права доступа** (текущее содержание его таблицы прав доступа) 483.

Чтобы выражения фильтра могли ссылаться на данные поля, нам необходимо использовать имена полей вместо их описаний. Имена полей показываются во всплывающем окне при наведении мышки на поле описания:



В данном всплывающем окне мы видим, что имя первого поля - `username`. Это имя можно использовать для создания выражения фильтра, которое позволит нам выбирать только события входа в систему определенного пользователя.

К примеру, мы хотим фильтровать события входа в систему пользователя Чарли. Мы будем использовать следующие выражения фильтра:

```
{username} == 'charlie'
```

Когда данное выражение установлено, ссылка `{username}` относится к полю `username` в Таблице данных события. Это строковое поле, поэтому разрешение приводится в виде строкового значения. Данное значение сравнивается со строковым текстом `charlie`. Если они совпадают, событие фильтруется и отображается в журнале событий, в противном случае, оно пропускается. Если мы вводим данное значение в поле "Выражение фильтра" таблицы "Правила фильтра" и запускаем фильтрацию, в журнале событий будут отображаться входы в систему только пользователя Чарли.

Эффекта, показанного в этом примере, можно было бы также добиться, ограничив контекст фильтра до `users.charlie`, но этот пример служит просто для иллюстрации изучаемого вопроса. Более полезным, например, было бы отображение входов в систему нескольких пользователей, а не отдельно взятых.

ВЫРАЖЕНИЯ ФИЛЬТРА В ПОЛЬЗОВАТЕЛЬСКОМ ЦВЕТОВОМ ВЫДЕЛЕНИИ

Каждая запись в таблице пользовательского цветового выделения 768 также может содержать Выражение Фильтра. Если такое выражение задано и оценивается как TRUE, к данному событию применяются правила цветового выделения.

Если выражение фильтра не задано в записи пользовательского цветового выделения, будут выделяться все события, удовлетворяющие опциям **Маска контекста**, **Имя события** и **Уровень**, определяемым данной записью.

Пример:

Некоторые события являются критическими для функционирования системы. Такие события имеют высокий уровень 73, такой как **Ошибка** или **Критическая ошибка**. Мы можем выделить оранжевым цветом события уровня "Ошибка". Уровень события указывается специальной иконкой в журнале событий, но правила цветового выделения могут помочь системным операторам легче заметить данные события.

Для выделения событий Device, нам нужно добавить новое правило в таблицу "Пользовательское цветовое выделение":

Поле	Значение
Маска контекста	users.*.deviceservers.*.devices.*
Имя события	событие
Уровень	не определен
Выражение фильтра	
Цвет выделения	оранжевый

Чтобы выбрать события уровня Ошибка, мы можем использовать следующее выражение фильтра:

```
{env/level} == 4
```

В качестве варианта мы можем добавить правило для выделения событий уровня "Критическая ошибка" красным цветом. Оно будет содержать следующее Выражение Фильтра:

```
{env/level} == 5
```

Если мы просто хотим выделить все события, уровень которых выше, чем Ошибка (т.е. выше уровня 4), мы можем просто ввести "4" в поле **Уровень** пользовательской таблицы выделения события. То же самое можно проделать при помощи выражения `{env/level} > 4`.

СРЕДА ВЫЧИСЛЕНИЯ

В качестве исходного материала вы можете найти ниже сформулированную среду вычисления выражения фильтра.

Среда вычисления 112 Выражения фильтра и Выделения выражения:	
Контекст по умолчанию 119	Контекст события.
Таблица данных по умолчанию 120	Таблица данных, содержащая данные, относящиеся к событию 73.

Строка по умолчанию 119	0
Переменные среды 123	

Имя переменной	Тип значения	Описание
id	длинное	уникальный идентификатор события
context	строка	Полный путь к контексту события.
event	строка	Имя события.
level	целое	Уровень 73 события.
time	дата	Временная метка события.
acknowledgements	таблица данных	Таблица подтверждений 76 события.
enrichments	таблица данных	Таблица обогащений 76 события.

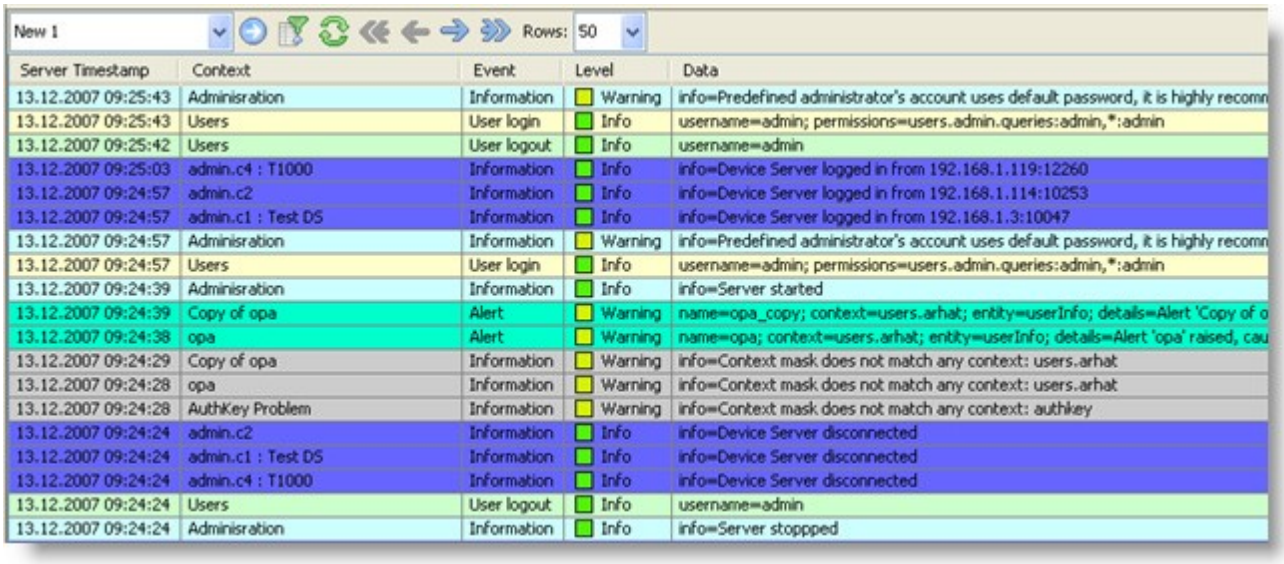
12.7.2 Цветовое выделение событий

События в Журнале событий могут быть маркированы цветом. Выбор цветов определяется в правилах фильтра 769 или правилами пользовательского цветового выделения 768. Правила цветового выделения имеют больший приоритет, поэтому если событие подходит под правило выделения цветом, цвет, заданный в правилах фильтра, будет игнорироваться.

Цвет, определяемый правилами цветового выделения, используется для выделения события, если:

- Контекст, в котором произошло событие, совпадает с **маской контекста**, определяемой в правиле.
- Если **Имя события** не равно "*****" (Все события) или имя события совпадает с параметром **Имя события**.
- Уровень данного события должен быть больше или равен **Уровню** серьезности, обозначенному в правиле.
- Событие совпадает с **Выражением Фильтра**, если таковое назначено правилом выделения.

Список событий в Журнале событий AtomMind Client выглядит таким образом:



12.7.3 Конфигурация фильтра событий

Данный раздел посвящен свойствам конфигурации фильтра событий.

Все свойства, описанные далее, доступны через действие Конфигурировать 105 в контексте фильтра событий 1509.

12.7.3.1 Свойства фильтра

Далее приведены основные опции фильтра событий.

Описание поля	Имя поля
Имя поля. Имя фильтра события[1509]. Должно удовлетворять соглашению об именах[42]. Используется для ссылки на данный фильтр из других частей системы.	name
Описание фильтра. Текстовое описание тревоги. Также является описанием[43] контекста тревоги.	description
Фильтр по умолчанию. Определяет, является ли текуший фильтр фильтром пользователя[478] по умолчанию. Фильтр по умолчанию автоматически активируется в Журнале Событий[398].	defaultFilter
Показывать имена полей в колонке "Данные". Показывает имя каждого поля в таблице данных, определяемое форматом событий[73] в столбце "Данные" журнала событий. Если данная настройка отключена, отображается только значение поля.	showDataFieldNames
Показывать пути контекста вместе с их описаниями. Показывает пути и описания событий в столбце "Контекст" журнала событий.	showServerContextNames
Показывать имена событий вместе с их описаниями. Показывает имена и описания событий в столбце "Контекст" журнала событий.	showServerEventNames

Данная информация доступна через переменную childInfo[1510].

12.7.3.2 Правила фильтра

Данное свойство определяет категории списка событий, которые будут отображаться, когда фильтр активирован.

Описание поля	Имя поля
Описание. Текстовое описание категории события.	описание
Маска контекста. Контролируемые контексты данного события. Подробное описание см. в разделе Маски контекста[44].	маска
Имя события. Имя контролируемого события. Данное поле может быть установлено на "*" (Все события), чтобы выполнять мониторинг всех событий вне системы.	событие
Включено. Когда данный флажок выключен, категория неактивна, события, определяемые ею, не отображаются. Это то же самое, что и "удалить" категорию, но на самом деле вы ее не удаляете (поэтому вы сможете ее в дальнейшем использовать или быстро включить снова).	активен
Уровень. События уровня[73] ниже заданного в данном поле уровня показываться не будут.	уровень
Параметризован. Указывает, что правило фильтра было параметризовано, т.е. поле "Данные параметризованного источника" должно быть использовано для построения выражения фильтра во время его активации с параметрами, заданными оператором. Для более подробной информации см. раздел Параметризованные фильтры[765].	параметризован
Выражение фильтра. Выражение AtomMind[112], используемое для более мелкой фильтрации событий. Используется, если флажок **"Параметризован"** отключен для данного правила.	выражение
Данные параметризованного источника. Используется, если флажок **Параметризован** включен для данного правила. Для более подробной информации см. раздел Параметризованные фильтры[765].	параметризован

Цвет выделения. Цвет, используемый для выделения событий данной категории в журнале событий. Цвет для каждого определенного события может быть изменен в настройках пользовательского цветового выделения 768.	цвет

Важно: Как только вы внесли изменения в данное свойство, **сохраните и перезапустите** фильтр. Это приведет к обновлению свойства Дополнительные видимые поля 770 согласно выбранным вами типам событий. Это важный момент.

Данная информация доступна через переменную rules 1510.

12.7.3.3 Основные видимые поля

Данное свойство определяет, какие столбцы будут отображаться в журнале событий при показе определенного события.

Описание поля	Имя поля
Имя контекста. Показывает контекст 41, где было инициировано событие.	context
Имя события. Показывает имя события.	event
Уровень события. Показывает уровень 73 события.	level
Данные события. Показывает содержание таблицы данных 49, относящейся к событию.	data
Подтверждения. Показывает подтверждение 73 события.	ack

Данная информация доступна через переменную shownFields 1511.

12.7.3.4 Дополнительные видимые поля

Как было указано ранее, события имеют *дополнительные* поля. Это данные, относящиеся к событию, такому как **имя пользователя** (в случае рассмотрения события **вход в систему**). Данное свойство позволяет вам настроить, какие именно из этих полей будут отображаться в журнале событий.

Как только вы выбрали, какие типы событий будут фильтроваться (т.е. изменили таблицу правил фильтра 769 согласно вашим требованиям), вам необходимо сохранить таблицу. После сохранения и перезагрузки конфигурации фильтра **дополнительные видимые поля** будут обновлены.

Данный список включает теперь *все* дополнительные поля, но только для тех типов фильтруемых событий, для которых вы настроили фильтр (и даже для *отключенных* категорий событий). Итак, перед вами появится масса дополнительных полей, из которых вам нужно выбрать.

Предположим, вы осуществляете фильтрацию как событий **вход в систему**, так и событий **тревоги**. Список дополнительных видимых полей для данного фильтра позволит вам включить (сделать видимыми) все поля для обоих типов событий. Таким образом, у вас будет **имя** (для событий Тревоги, чтобы указать имя Тревоги в списке), а также **имя пользователя** (для событий Вход в систему, чтобы показать, кто выполнил вход). Если вы включите **имя пользователя**, дополнительный столбец **имя пользователя** появится в таблице фильтра, когда вы будете просматривать его в журнале событий. Для строк, показывающих события **вход в систему**, данный столбец будет содержать имя пользователя. Для строк, показывающих события **тревоги**, данный столбец будет пуст (не будет содержать никакого значения, потому что это неприменимо).

Мы рекомендуем вам тщательно выбирать на данном этапе, иначе в результате вы получите огромный список для фильтра в журнале событий, и его прокрутка может быть неудобна.

Описание поля	Имя поля
Имя поля. Имя поля	name
Описание поля. Текстовое описание поля.	description
Описания событий. Показывает, к какому событию относится данное поле.	edescs

Видимый. Указывает, что данное поле отображается в журнале событий.	shown

Вам кажется, что таблица не содержит требуемые вами значения? Попробуйте **сохранить**, а затем **перезагрузить** весь фильтр. Это необходимо делать постоянно, чтобы данная таблица обновлялась динамично и соответствовала событиям, сконфигурированным в Правилах Фильтра.

Данная информация доступна через переменную additionalFields 1511.

12.7.3.5 Пользовательское цветовое выделение

Данная таблица определяет особые правила выделения 768.

Описание поля	Имя поля
Маска контекстов. Маска контекстов для тех событий, к которым будет применяться данное правило цветового выделения.	mask
Имя события. Имя события, которое нужно выделить. Значение данного поля может быть установлено на "*" (Все события), чтобы назначить выделение для всех событий.	event
Уровень. Минимальный уровень события, которое нужно выделить.	level
Выражение для фильтрации. Выделяет события, удовлетворяющие выражению фильтра, если задано.	expression
Цвет подсветки. Цвет выделения.	color

Данная информация доступна через переменную customHighlighting 1512.

12.7.3.6 Настройки просмотра истории

Данная таблица определяет настройки, применимые для просмотра истории событий 73.

Описание поля	Имя поля
Ограниченный диапазон времени. Флажок, указывающий, что будет выбрана только определенная подгруппа событий в истории.	limitTimeFrame
Диапазон времени (в единицах времени). Если включено поле "Ограниченный диапазон времени", данная опция определяет, какой интервал времени должен быть выбран. См. далее пример.	timeFrame
Единица времени. Определяет единицу измерения для диапазона времени (месяц, неделя, день, час, минута и т.д.).	timeUnit
Использовать пользовательское конечное время. Если данный флажок активирован, браузер истории событий будет показывать события, которые произошли до времени окончания.	useCustomEndTimePoint
Конечное время. Если опция "Задать конечное время" включена, в журнале событий будут показываться те, которые произошли до данного времени.	customEndTimePoint

Пример: Если интервал времени равен 10, а Единица Времени выбрана День, просмотр истории событий будет показывать те события, которые произошли в течение последних 10 дней. Если в то же время включена опция "Задать конечное время", просмотр истории событий будет показывать те события, которые произошли за 10 дней до времени окончания.

Данная информация доступна через переменную historySettings 1512.

12.7.4 Параметризованные фильтры

Параметризация фильтров полезна, когда в хотите позволить оператору конфигурировать фильтр во время его работы. Если запись в правилах фильтра 769 отмечена, как "Параметризован", пользователю будет необходимо ввести один или более параметры в процессе активации фильтра (т.е. выбор из контекстного списка журнала событий).

Параметризация является сложной темой, которая выходит далеко за рамки данной статьи. Однако, чтобы понять этот раздел, вам нужно иметь четкое представление о процессе и механизме параметризации. Поэтому, если вы хотите использовать параметризованные фильтры и вы не до конца усвоили данную тему, рекомендуем вам перейти к теме Механизм параметризации 1444, прочитать всю статью до полного ее понимания, а затем вернуться в данный раздел.

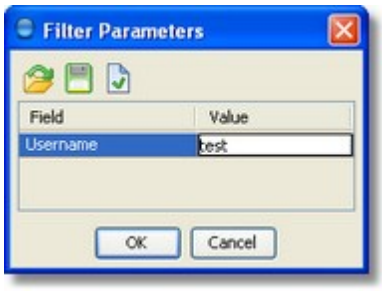
Как было упомянуто выше, свойство "Правила фильтра" содержит много строк (записей), каждая из которых описывает тип фильтруемого события. Каждая запись может включать в себя выражение фильтра (которое может быть параметризовано). Даже если свойство "Правила фильтра" содержит более одного такого параметризованного выражения фильтра, все параметры фильтра вводятся в одно диалоговое окно. Фильтр событий использует UI-алгоритм Редактировать данные 90, чтобы вызвать параметры фильтрации.

Когда опция **Параметризован** включена в записи правил фильтра, метод обработки фильтра использует **параметризованные данные источника** для построения выражения фильтра в реальном времени, основанного на заданных оператором параметрах. Для подробной информации обратитесь к разделу Метод параметризации 1444.

Далее приведен пример Данных источника Параметризатора с одним строковым полем в **Формате** (имя пользователя) и **Параметризованное выражение**:

```
{username} ~= '.*<e>{username}</e>.*'
```

Фильтр, содержащий данное параметризованное выражение, потребует параметр "Имя пользователя" при активации:



Данный фильтр обрабатывает только те события, у которых поле **Имя пользователя** содержит значение, вводимое оператором в качестве параметра `Username`, потому что итоговое выражение фильтра будет таким:

```
{username} ~= '.*<e>VALUE_OF_USERNAME_PARAMETER_ENTERED_BY_USER</e>.*'
```

Опять же не нужно ожидать полного понимания этого раздела на данном этапе -- это всего лишь мимолетный взгляд на возможности параметризованных фильтров. Чтобы до конца понять, как это работает, прочитайте статью Метод параметризации 1444.

Автоматическая параметризация

Данная функция автоматически формирует данные параметризации для категорий события, выбранных из правил фильтра 769. Вы можете автоматически параметризовать запись в правилах фильтра, только если она уже не содержит выражение фильтра. Процесс автоматической параметризации описан здесь 1513.

Данные параметризации, формируемые данной функцией, определяют два параметра для каждого поля в таблице данных события. Т.е., при автоматической параметризации события 73, которое имеет только одно поле в свойстве **Формат события**, пользователю будет предложено ввести два значения при выполнении фильтрации:

- **"Filter by 'description_*of_field (name_of_context_where_event_is_defined)*'":** Это что-то похожее на **Фильтр по 'Username (users)'**. Представляет собой логическое значение (иными словами, неизменяемое), которое включает или отключает фильтр в зависимости от значения данного поля.
- ***"description_of_field (name_of_context_where_event_is_defined)"*:** Это что-то похожее на **Фильтр по 'Username (users)'**. Представляет собой поле ввода (или иной вид ввода изменений, в зависимости от типа поля), куда вы можете ввести значение, по которому необходимо осуществлять фильтрацию.

Например, если вы применяете автоматическую параметризацию к категории "**События администрирования**", данные параматризации будут определять два параметра: Фильтр по **'Info (administration)'** и **Info (administration)**:

Первый параметр позволяет фильтровать по полю **Info (информация)** события уровня "Информация" 77 (в контексте **администрирования**). Второй параметр содержит в себе ожидаемое значение поля. Будут фильтроваться только события с полем **Info (информация)**, равным данному значению.

12.7.5 Встроенные фильтры

Дистрибутив AtomMind Server содержит несколько встроенных Фильтров Событий:

Все события

Фильтр всех событий показывает все важные события системы и Device.

Правила данного фильтра используются большинством системных администраторов и операторов. Далее приведен их краткий обзор:

Правило	Примечания
События администрирования	Типичные события администрирования, такие как невыполненные попытки входа в систему пользователя 478 и регистрация новых учетных записей, отчеты о подключенных серверах внешних устройств 2098, уведомления о запуске и отключении AtomMind Server и т.д.
События пользователей	События, относящиеся ко всем учетным записям пользователей.
События пользователя	События, относящиеся к определенной учетной записи пользователя 478.
События терминала сбора данных	События, относящиеся к определенному терминалу сбора данных 497, такие как проблемы коммуникаций, вопросы синхронизации, события, полученные от аппаратных устройств и т.д.
События тревоги	Уведомления о тревогах 779, которые высвечиваются в интерфейсе или присылаются по e-mail, даже если вы не контролируете фильтр. В основном используется во время проверки истории тревог и просмотра подтверждений. Другая информация о тревогах, таких как эскалации 805 тревог, проблемы доставки уведомлений 791 и т.д., также включены в данную категорию.
События планировщика	Информация о выполнении или невыполнении запланированных задач 823.
События датчика	События датчика 2181, такие как изменения состояния или ошибки в расчете значений.

Журнал сервера

Данный фильтр показывает события журнала 166 сервера, которые обычно записываются в серверную консоль или файл журнала.

Технически фильтр установлен отображать события журнала 1451 из контекста администрирования. Так как события журнала непостоянны, события истории недоступны фильтру журнала сервера.

Большинство сообщений, записываемых в журнал сервера, являются системными сообщениями, которые не предусмотрены для анализа операторами. Таким образом, некоторые сообщения могут быть неинтернационализированы и доступны только на английском языке.

События Device

Данный фильтр показывает только те события, которые относятся к взимодействию с Device, и события, формирующиеся аппаратными устройствами.

Тревоги

Данный фильтр отображает только события тревог 790.

Действия

Фильтр отбирает события действий 1451, т.е. списки действий 87, выполненные системными администраторами и компонентами.

12.7.6 Фильтры событий в распределенной архитектуре

Этот раздел описывает особенности поведения фильтров событий в распределенной инсталляции 1332 AtomMind.

Чтобы фильтр событий, подключенный из сервера-поставщика, появился в Журнале Событий 398 клиентов, подключенных к серверу-потребителю, убедитесь, что путь контекста подключенного контекста фильтров на сервере-потребителе соответствует контекстной маске `users.*.filters.*`, т.е. удаленный фильтр появился среди локальных фильтров.

12.8 Корреляторы событий

Корреляция событий - это механизм, который позволяет обнаружить сложные схемы взаимодействия между событиями и реализовать для них алгоритмы обработки.

Коррелятор событий:

- "Слушает" события из входных потоков. Это события 73, порождаемые контекстами 41, или необработанные события, поступающие от поддерживаемых движком коррелятора протоколов.
- Парсит, коррелирует и обрабатывает эти события согласно набору определенных правил. Эти правила определены в языке скрипта, используемом движком коррелятора.
- Отправляет события в выходные потоки, которые являются результатом корреляции и обработки, осуществляемой коррелятором событий. И это снова события 73, порождаемые контекстами 41, или необработанные события, поступающие от поддерживаемых протоколов.

роль корреляторов событий в AtomMind

Коррелятор событий подписывается на события 73, генерируемые контекстами 41. После обработки данных из этих событий, коррелятор генерирует свои собственные события. Вы можете настроить коррелятор, чтобы публиковать такие событий как поступающие из контекста самого коррелятора, либо из другого контекста.

Контексты Коррелятора событий 774 хранятся к контейнере Коррелятора событий 774 в Системном дереве. Каждый контекст определяет только один коррелятор событий.

Поскольку коррелятор может объединять несколько сходных потоков и обрабатывать каждый из них отдельно от других, необязательно создавать коррелятор для каждого входного потока, который вы хотите обработать.

расширения

Расширения - это модули, установив которые можно расширить базовый функционал движка коррелятора. Например, вы можете установить расширение с функциями, позволяющими работать с регулярными выражениями или с необработанными данными событий из протокола Kafka.

Технически, расширения представляют собой автономные JAR файлы и могут быть установлены отдельно.

Чтобы узнать больше об использовании расширений, см. раздел Плагин корреляторов событий 774. Информацию о доступных расширениях см. в документации по расширениям движка коррелятора.

12.8.1 Плагин коррелятора событий

Функционал корреляторов событий обеспечивается отдельным плагином 207 для AtomMind Server.

установка плагина коррелятора событий

Чтобы установить плагин корреляторов событий:

1. В Системной дереве кликните правой кнопкой мыши на корневой контекст (сервер). Выберите **Установка модулей и решений**.
2. Откроется диалоговое окно **IoT магазин**. Выберите магазин.
3. На экране **Выберите решения** нажмите **OK**. Не выбирайте никакие решения.
4. На экране **Выберите модули**, выберите плагин **context.correlator**. Нажмите **OK**.

5. Перезапустите сервер для завершения установки.

Свойства плагина

Плагин коррелятора событий располагается в контейнере **Драйвера и расширения**. Он имеет следующие свойства.

ДИРЕКТОРИЯ С РАСШИРЕНИЯМИ

Это свойство определяет папку для расширений плагина движка коррелятора. Движок загружает файлы расширений, расположенные в этой папке. Чтобы узнать больше о расширениях, см. документацию по расширениям движка коррелятора.

- **Описание**: Директория с расширениями
- **Поле**: extensionsFolder
- **Тип**: String

Установка расширений

Расширения хранятся в JAR файлах. Чтобы установить расширение, скопируйте его в папку, указанную в параметре **Директория с расширениями**.

Следующие расширения предустанавливаются вместе с плагином. Вы можете пользоваться функционалом этих расширений, не устанавливая их.

- math (execution-math)
- str (execution-string)
- time (execution-time)

12.8.2 Скрипты коррелятора событий

Скрипты коррелятора используют язык *Streaming SQL*.

Так как Streaming SQL - сам по себе обширный язык, в данном разделе объясняются только базовые понятия этого языка, имеющие отношение к AtomMind. Более подробно о синтаксисе, операторах и возможностях Streaming SQL см. Гайд по Streaming SQL.

Основные понятия языка Streaming SQL:

- **Поток -** серия событий, упорядоченных по времени. Потоки делятся на входные и выходные. Коррелятор "слушает" входные потоки и выводит события в выходные потоки.
- **Запрос -** выражение, которое может брать события из одного или более потоков, обрабатывать эти событий в потоковом режиме и выводить в выходной поток.
- **Функция** - упакованная сложная логика исполнения, производящая операции с данными событий и возвращающая полученные данные. Можно вызвать функции для выполнения операций с данными событий.
- **Фильтр** - условное выражение, определенное для потока. В результате, будут обрабатываться только события, соответствующие определенному выражению.
- **Окно** - подмножество данных с определенным критерием, взятых из потока,. Окна динамичны. По мере генерирования новых событий в потоке, данные окно самообновляются.
- **Шаблон** - выражение, которое определяет логику корреляции событий. С помощью шаблонов можно коррелировать между собой события из одного и более входных потоков и генерировать выходные события из данных коррелируемых событий.

Пример скрипта см. ниже. Скрипт берет значения из событий, генерируемых виртуальным устройством, и выводит их в выходной поток вместе с дополнительным сообщением.

```
@Source(type = 'internalEvent', context='users.admin.devices.virtual', event='event1', @map(ty
define stream SourceStream (string string, int int);

@Sink(type = 'internalEvent', event='test', @map(type='internalEvent'))
define stream OutStream (message string, str string, num double);

from SourceStream
select 'Event 1 detected' as message, string as str, cast(int, "double") as num
```

```
    insert into OutStream;
```

В этом примере, `SourceStream` - входной поток, `OutStream` - выходной поток. `@Source` и `@Sink` - аннотации потока (см.ниже). `from ... select ... insert into` - запрос. `cast(int, "float")` - встроенная функция, которая преобразует значения из Integer в Double. В приведенном примере нет фильтров, окон и шаблонов. Они описаны в отдельных разделах.

Определение потоков

Осуществляется при помощи команды `define stream` с дальнейшим указанием определений атрибутов потока:

```
define stream TestStream (message string, num double);
```

Указанный в примере выше поток `SourceStream` имеет два атрибута: *message* типа *string* и *num* типа *double*.

Имена атрибутов должны совпадать с именами полей событий. Например, если событие в AtomMind называется *message*, нельзя использовать для него другое имя поля в скрипте коррелятора. В противном случае, коррелятор событий может проигнорировать некоторые события, что может привести к потере данных.

Привязка потоков к событиямAtomMind

Определение потока требует использования аннотации. Механизм, который привязывает события AtomMind к потокам движка коррелятора, использует аннотации.

Аннотация **@Source** указывает на то, что следующее за ней определение потока - это входной поток.

Пример входного потока с аннотацией:

```
    @Source(type = 'internalEvent', context='users.admin.devices.virtual', event='event1', @map(ty
    define stream SourceStream (string string, int int);
```

Атрибуты аннотации **@Source**:

- **type** определяет тип входного потока. Этот параметр определяет протокол, который будет использоваться для создания данного входного потока. По умолчанию поддерживается только тип `internalEvent`. Этот тип соответствует событиям, генерируемым внутри AtomMind. Вы можете расширить количество поддерживаемых типов потоков, установив расширения 774. Например, вы можете использовать MQTT и RabbitMQ типы потоков, установив соответствующие расширения движка коррелятора.
- **context** - специальный атрибут для типа потока `internalEvent`. Определяет контекст, имеющий событие, которое должен "слушать" коррелятор.
- **event** - специальный атрибут для типа потока `internalEvent`. Определяет имя события в определенном **контексте**. Коррелятор будет "слушать" события с этим именем, генерируемые контекстом.
- **@map()** - преобразователь входного формата для движка коррелятора. Преобразователь по умолчанию `internalEvent` преобразует формат событий, сгенерированных в AtomMind, в формат, который может обработать коррелятор.

Аннотация **@Sink** указывает на то, что следующее за ней определение потока - это выходной поток.

Пример выходного потока с аннотацией:

```
    @Sink(type = 'internalEvent', context='users.admin.devices.virtual' event='event2', @map(type=
    define stream OutStream (string string, int int);
```

Атрибуты аннотации **@Sink**:

- **type** определяет тип выходного потока. Аналогичный атрибут, как и для аннотации **@Source**.
- **context** - специальный атрибут для типа потока `internalEvent`. Определяет контекст, который будет генерировать выходные события. Если данный атрибут не определен, события будут генерироваться контекстом самого коррелятора.
- **event** - специальный атрибут для типа потока `internalEvent`. Определяет имя события, которое будет сгенерировано в определенном **контексте**. Если событие генерируется контекстом коррелятора, его формат определяется свойством Формат выходного события 1522. Если другим контекстом, событие с определенным именем уже должно существовать в определенном контексте. В обоих случаях, имена полей событий должны соответствовать именам атрибутов потока. Если требуется подмножество полей из события, можно указать для них только атрибуты потока.

- **@map()** преобразователь выходного формата для движка коррелятора. Преобразователь по умолчанию `internalEvent` преобразует формат событий, сгенерированных движком коррелятора, в формат, используемый AtomMind.

Обработка данных события

Для обработки данных события используйте запросы. Запросы - это команды, которые собирают данные из одного или более входных потоков и выводят данные в выходной поток.

Простой запрос выглядит следующим образом:

```
from SourceStream
select int as num, string as message
insert into OutStream;
```

Запрос отбирает из входного потока поля с именами *int* и *string.* Значение поля *int* помещается в поле *num*. Значение поля *string* помещается в поле *message*. Получившееся событие имеет два поля: *num* и *message*. Оно отправляется в выходной поток.

Следующий запрос выполняет функции для вычисления значений полей для исходящего события:

```
from SourceStream
select ifThenElse(regex:matches("(.*)bbb(.*)", string), "match", "no match") as message, cast(
insert into OutStream;
```

Этот запрос использует три *функции* для вычисления значений полей. Функция **ifThenElse() -** встроенная функция языка Streaming SQL. Она обеспечивает условную логику. Функция **cast() -** встроенная функция языка Streaming SQL. Она преобразует свой параметр в другой тип. В приведенном примере, значение типа Integer из поля *int* преобразуется в значение типа Double, которое записывается в поле *num* выходящего потока. Функция **regex:matches()** приводит поле *string* входного события в соответствие с регулярным выражением. **regex:matches()** - функция расширения. Чтобы использовать функции расширения, необходимо установить соответствующие расширения 774.

Функции расширений должны вызываться с использованием пространства имен. В приведенном выше примере `regex` - это пространство имен функции `matches`.

Более подробно о встроенных функциях см. Документацию по API streaming SQL. Подробнее о функциях расширений см. документацию по расширениям движка коррелятора.

фильтры

Фильтр - это условие, определенное для потока. Будут обработаны только события, отвечающие условию, а все остальные - проигнорированы.

Следующий фильтр отбирает события конкретного пользователя:

```
@Source(type = 'internalEvent', context='users.admin.models.actionHandler', event='action', @m
define stream UserActions (user string, action string, level int);``

from UserActions[user == 'administrator']
select action as message
insert into OutStream;
```

Можно использовать логические операторы внутри фильтров, чтобы комбинировать условные конструкции. Следующий пример отбирает события конкретного пользователя, превышающие определенный уровень серьезности.

```
from UserActions[user == 'administrator' and level > 1]
select action as message
insert into OutStream;
```

окна

Окно - это подмножество данных с определенным критерием, взятых из потока. По мере того, как все больше данных добавляется через входной поток, обновляются также и данные в окне. Как и в случае с входящим потоком, можно обрабатывать данные внутри окна и отправить их в выходящий поток.

Например, вы можете использовать окно для хранения десяти последних значений температуры с устройства и сверять с ними все остальные входные события. При поступлении новых данных, данные в окне будут обновляться соответственно. Или, например, вы можете использовать окно, чтобы выбрать последние пять минут

активности для определенного устройства. Со временем данные в этом окне будут обновляться, поэтому события, содержащиеся в окне, всегда будут актуальными.

В потоке окна определяются при помощи префикса #window, за которым идет точка и тип окна с параметрами. Следующее окно получает последние 10 событий из потока:

```
from SourceStream#window.length(10)
```

Определения окон могут комбинироваться с фильтрами. Следующее окно получает 10 последних событий из потока, принадлежащего определенному пользователю:

```
from SourceStream#window.length(10)[user == 'administrator']
```

Следующий запрос использует окно и показывает максимальную температуру для последних 10 событий. Каждый раз при получении нового события, генерируется выходное событие:

```
from SourceStream#window.length(10)
select "Maximum temperature" as message, cast(max(int), 'double') as num
insert into OutStream;
```

Окна могут быть *скользящие* и *переворачивающиеся*

Скользящее окно обновляется при каждом новом событии, отвечающем критериям окна. Переворачивающееся окно обновляется, когда вся длина окна полностью заполнена соответствующими событиями. Например, окно вмещает три последних события. Генерируется новое событие. Скользящее окно обновится. В нем будет два старых события и одно новое. Переворачивающееся окно не будет обновляться, пока не поступят еще два новых события, а после обновления в нем окажется три новых события.

Чтобы определить скользящее окно, используйте типы **length** и **time**. Тип **length** будет выбирать количество последних событий. Тип **time** будет отбирать все события за указанные период времени.

```
-- select last 10 events from SourceStream
from SourceStream#window.length(10)

-- select all events that came in the last 10 minutes from SourceStream
from SourceStream#window.time(10 min)
```

Чтобы определить переворачивающееся окно, используйте типы **lengthBatch** и **timeBatch**. Тип **lengthBatch** будет обновлять каждое установленное число событий. Тип **timeBatch** будет обновлять окно через каждый заданный интервал времени.

```
-- select every 10 events from SourceStream
from SourceStream#window.lengthBatch(10)

-- select all events that come every 10 minutes from SourceStream
from SourceStream#window.timeBatch(10 min)
```

Например, если длина окна **lengthBatch** равна 10, оно будет генерировать выходящее событие для каждых 10 входных событий. Окно **timeBatch** с 10-минутным временным периодом будет генерировать выходящее событие каждые 10 минут.

Следующий пример будет генерировать событие каждые 10 минут. Это событие будет содержать максимальное значение поля *int* из всех событий, полученных за последние 10 минут. Если событий не будет получено, событие не будет сгенерировано.

```
from SourceStream#window.timeBatch(10 min)
select 'Maximum temperature' as message, cast(max(int), 'double') as num
insert into OutStream;
```

Шаблоны

С помощью шаблонов вы можете коррелировать события друг с другом и применять сложную логику на основе событий.

Синтаксис шаблонов очень простой и легко настраиваемый. Следующий пример - лишь небольшая часть того, что модно сделать при помощи шаблонов. Для большей информации см.Гайд по Streaming SQL.

Базовый шаблон определяет последовательность событий, которые должны идти друг за другом, отделяемые оператором `->`. Такой шаблон показан в следующем примере. Если срабатывает аварийная сигнализация, и затем в течение 10 минут следует повышение температуры выше 80 градусов, активируются средства пожаротушения.

```
from InStreamAlarms[alarm == 'emergency'] ->
    InStreamTemperature[temp > 80]
    within 10 min
select
    "activateExtinguishers" as action
insert into OutSystemControl;
```

Другой пример. Есть два входных потока. Один из них - от датчик, который посылает информацию об уровне топлива в баке. Другой подает сигналы, когда уходит и возвращается оператор. Выходящий поток контролирует насос.

```
-- 0 to 100
@Source(type = 'internalEvent', context='users.admin.devices.fuelSensor', event='fuelLevel', @
define stream SourceSensor (level int);

-- "isAway", "isPresent"
@Source(type = 'internalEvent', context='users.admin.models.operator', event='action', @map(ty
define stream SourceOperator (status string, name string);


-- 1 - enable, 0 - disable
@Sink(type = 'internalEvent', context='users.admin.models.pumpController', event='control', @m
define stream OutPumpControl (action int, message string, operator string, value int);
```

Если оператор ушел и не вернулся в течение 10 минут, а в это время уровень топлива достиг 5%, включить насос. Этот шаблон использует ссылки на события (`e1` и `e2`), чтобы обращаться к данным из связанных событий.

```
from every( e1=SourceOperator[status == 'isAway'] ) ->
    not SourceOperator[status == 'isPresent' and name == e1.name] for 10 min and e2=SourceSens
select
    1 as action,
    "Automatic activation (operator not present)" as message,
    e1.name as operator,
    e2.level as value
insert into OutPumpControl;
```

Если в какой-то момент уровень топлива поднимается выше 95% и не падает в течение следующих 2 минут, остановить насос. Необязательно искать этот шаблон для каждого события, когда уровень топлива превышает 95% (нет необходимости в ключевом слове `every`). Первое событие с повышением уровня более, чем до 95%, запустит данный шаблон, и все другие подобные события будут обрабатываться как его часть.

```
from e1=SourceSensor[level > 95] ->
    not SourceSensor[level < e1.level] for 2 min
select
    0 as action,
    "Automatic pump shutdown" as message,
    "auto" as operator,
    level as value
insert into OutPumpControl;
```

12.9 Тревоги

Тревога - это ответ системы на событие или условие, заданное пользователем. AtomMind поддерживает широкую систему тревожных оповещений - один из важнейших инструментов в современных системах монторинга. Тревоги оповещают системных операторов, когда что-то идет не так в любой части распределенной системы. Если бы не было тревог, оператору приходилось бы постоянно проходиться по всей системе и нажимать на устройства, чтобы просто убедиться, что все в порядке. Тревоги сообщают операторам, на что они должны обратить внимание.

У каждого пользователя свой набор тревог, но можно также пользоваться тревогами совместно. Тревога включает:

- Триггеры
- Правила уведомлений
- Правила эскалации
- Корректирующие действия
- Триггеры тревоги

Тревоги запускаются триггерами (условиями поднятия тревог). Триггером может быть событие, состояние или изменение состояния компонента/аппаратного устройства системы. Более подробно см. в разделе триггеры 780.

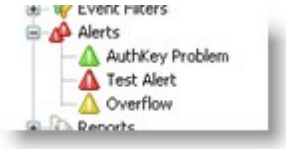
Когда поднимается тревога, система отвечает следующими способами:

- Уведомление о тревоге может быть немедленно отправлено владельцу тревоги. Его также могут попросить подтвердить 806 тревогу.
- Могут быть разосланы E-mail сообщения и другие уведомления 781 владельцу тревоги, пользователю 478(ям) системы или заданным получателям.
- Тревога может храниться в истории событий 75 как обычное событие.
- Могут быть выполнены некоторые корректирующие действия 87 в интерактивном и неинтерактивном режимах 99.
- Тревога может изменить свое состояние 781.

Каждый пользователь 478 имеет свой набор тревог.

Администрирование Тревог

Для администрирования тревог используются два контекста: общий контекст Тревоги 1452, который выступает в роли контейнера, и контекст Тревога 1454, который содержит в себе информацию об одной тревоге.



Триггеры

Тревоги запускаются *триггерами*. Каждая тревога может иметь несколько связанных с ней триггеров. Каждый триггер определяет условие, при котором должна быть активирована тревога. Каждая тревога может проверять одно или более устройств или ресурсов, например, все устройства в группе. Вместе с возможностью задать несколько триггеров для одной тревоги, это делает настройку очень гибкой. Существует два типа триггеров:

- Триггеры события 787
- Триггеры переменной 782

Триггер: действие, которое запускает ход некоторых событий.

Каждая тревога может не определять ни одного или же определять несколько триггеров каждого тип. Если триггеры не определены, активация тревоги никогда не происходит. Если же определено несколько триггеров, они действуют отдельно - каждый триггер активирует тревогу (нет необходимости сразу во всех - одного триггера достаточно, чтобы активировать тревогу).

ТРИГГЕРЫ СОБЫТИЙ

Триггер события срабатывает, когда происходит событие определенного типа, определенного условиями триггера. Это условие гибко настраивается выражением, что делает возможной комплексную проверку. Например, система мониторинга транспортного средства может создавать тревогу, если событие об ударе, полученное от контроллера на транспортном средстве, показывает превышение порогового значения силы удара.

Триггеры событий поддерживают корреляцию событий, что позволяет активировать тревогу событием одного типа, а деактивировать событием другого типа (коррелирующим событием).

Любой триггер события можно настроить так, чтобы он активировался только при возникновении более N соответсвующих событий за определенный период времени.

ТРИГГЕРЫ СОСТОЯНИЯ

Триггер состояния может сработать либо в ответ на определенное состояние, либо на любое изменение состояния объекта мониторинга. Триггер состояния периодически проверяет значение определенной переменной (также указанное пользовательским выражением).

Время гистерезиса (застоя) триггера состояния можно настроить, чтобы активировать тревогу, только если условие длится дольше определенного времени. Например, триггер состояния может поднять тревогу, если подъем температуры выше 120 градусов сохраняется больше 3 минут. Отдельно можно задать гистерезис возврата к исходному состоянию.

Кроме того, триггеры состояния поддерживают обнаружение биения значения (частой смены), о чем сообщается с помощью тревоги особого вида.

Уведомления о тревогах

Когда тревога активирована одним из ее триггеров, AtomMind Server начинает отправку уведомлений 791 и выполнение корректирующих действий 793.

Уведомления о тревогах сообщают операторам об условиях, вызвавших тревогу, и дают другую релевантную информацию. Типы уведомлений включают:

- Всплывающие сообщения оператору (могут также включать просьбу подтверждения)
- Настраиваемые звуки
- E-mail уведомления. Тревоги можно подтверждать отправкой ответа на e-mail сообщение
- SMS уведомления
- Любые другие способы доставки уведомлений, такие как отправка сообщений в Skype через внешнее приложение

Кроме того, корректирующие действия 793 тревог могут реализовывать любые другие способы уведомления.

Состояния тревоги

Состояние тревоги указывает серьезность текущей тревоги. Оно включает в себя несколько факторов, таких как доступность событий тревоги, ожидающих обработки, активность триггера и правила эскалации.

См. Состояния тревоги 802 для более подробной информации.

АКТИВНЫЕ ТРЕВОГИ

Тревога может оставаться активной после возникновения, пока вызвавшее ее условие остается в силе, либо пока продолжается получение события, коррелирующего с активирующим событием. Сервер хранит список глобальный активных тревог и отслеживает активные экземпляры, связанные с каждым ресурсом и устройством. Активные тревоги с высоким приоритетом обычно визуализируются на инструментальных панелях с обзором системы.

Встроенные тревоги

ТРЕВОГА УСТРОЙСТВО ВЫКЛЮЧЕНО

Тревога "Устройство выключено" встроена в базовый пакет дистрибутива AtomMind Server. Она активируется, когда происходит разрыв соединения любого терминала данных 497 с сервером на более, чем на 10 минут.

Согласно триггерам переменной 782, данная тревога активируется только для тех Devices, у которых активна настройка Включить тревогу при отсутствии соединения 514.

Тревога "Устройство выключено" контролируется администратором по умолчанию.

ВОССТАНОВЛЕНИЕ ПОСЛЕ ОТКАЗА

Тревога восстановления после отказа активируется при сбое главного узла отказоустойчивого кластера 1326 AtomMind Server, и когда контроль осуществляется главным узлом сети. E-mail сообщение отправляется администратору по умолчанию 479 при возникновении такой тревоги.

Примеры

Некоторые примеры конфигураций реальных тревог описаны в разделе Примеры тревог 807.

12.9.1 Триггеры переменной

Триггер переменной активирует тревогу в случае:

- Когда значение переменной 61 удовлетворяет некоторому условию или
- Когда изменяется значение переменной (неважно, каким является новое значение) или
- Когда значение переменной выходит за пределы заданных порогов.
- Когда строка с определенным хэш-кодом (ID) впервые появляется в табличном значении переменной

Выбор между данными методами осуществляет свойство триггера **Режим**. Другие свойства триггера переменной описаны здесь 797.

Каждый триггер переменной исследует значение переменной каждого контекста, соответствующего настройке, определяемой маской 44 **контекста.** Параметр **Период проверки** задает период опроса. Таблица данных 49, которая содержит значение переменной, выбирается из контекста, затем, согласно настройке **Режим**, происходит следующее:

- Если в поле **Режим** выбрано **Состояние (ложь/истина)**, сервер вычисляет выражение 112, заданное в настройке **Выражение**. В таком случае данное выражение имеет логическое значение. Оно может содержать ссылки 117 на ячейки таблицы данных, содержащей значение переменной, но также может ссылаться на другие переменные и функции контекстов AtomMind Server. Если значение выражения равно TRUE, а результат предыдущего вычисления был FALSE (или если переменная проверяется впервые после запуска AtomMind Server или после последнего изменения свойств данной тревоги), триггер запускает и активирует тревогу. Если же результат вычисления равен TRUE и был такой же во время последней проверки, ничего не происходит. Если же она изменилась с TRUE на FALSE, действие также не выполняется, но последующее изменение на TRUE активирует тревогу.

Если определено **Record Key Expression** is specified, это **Выражение** is independently evaluated for each line of the Variable value.

- Если в поле **Режим** выбрано **Изменение состояния (любое значение)**, **Выражение** вычисляется с периодичностью заданной в поле **Период проверки** по такому же сценарию, но результат может быть любого типа. Сервер запоминает результат последнего вычисления. Если новый результат вычисления **Выражение** отличается от предыдущего, триггер поднимает тревогу.
- Если в поле **Режим** выбрано **Динамические пороги (число), Выражение** вычисляется с периодичностью заданной в поле **Период проверки** как и в первом сценарии, но результат должен быть только числом. Если это число меньше нижнего порога или больше верхнего порога, определенных в поле **Пороги**, триггер активируется и поднимает тревогу.

Если параметр **Маска контекста** указывает на более, чем один контекст, каждый контекст исследуется отдельно и тревога запускается каждый раз, когда значение переменной соответствует условию (или изменению, или выходу за установленные пороги в зависимости от настройки **Режим**) любого из данных контекстов.

Пример триггера переменной, когда Режим триггера – "Состояние (ложь/истина)":

Предположим, что мы осуществляем мониторинг переменной **Текущая температура**, полученной от датчика температуры. В данном случае **Выражение** может ссылаться на поле **Температура по шкале Цельсия** данной переменной и активировать тревогу, если температура превысила заданный предел.

Параметры триггера могут быть следующими:

Маска контекста	`users.user123.devices.temperature_sensor` (Данная маска соответствует одному контексту терминала данных) 1494.
Переменная	`currentTemperature`
Выражение	`{celsiusTemperature} > 100`
Режим	`State (true/false)`
Период проверки (секунды)	`10`
Задержка (секунды)	`0`
Уровень тревоги	`Warning`

Пример триггера переменной с "Изменение состояния (любое значение)":

Теперь предположим, что мы контролируем переменную **Список предупреждений**, поступающую от медицинского оборудования, контролирующего некоторые важные параметры состояния пациента. Значение данной переменной является таблицей данных из пяти записей, и каждая запись имеет одно строковое поле **Предупреждение**, содержащее текстовое предупреждение о состоянии пациента. По данному сценарию **Выражение** может объединить все сообщения-предупреждения, соединяя их, и активировать тревогу, когда любое из этих предупреждений удаляется, добавляется или изменяется.

Параметры триггера могут быть следующими:

Маска контекста	`users.user123.deviceservers.medical12.devices.monitor1` (Данная маска соответствует одному контексту терминала данных)[1494].
Переменная	`warningList`
Выражение	`{warning[0]} + {warning[1]} + {warning[2]} + {warning[3]} + {warning[4]}`
Режим	`Изменение состояние (любое значение)`
Период проверки (секунды)	`100`
Задержка (секунды)	`0`
Уровень тревоги	`Warning`

Время выдержки и гистерезис (зона нечувствительности)

Триггеры переменной имеют параметр **Задержка**, который помогает задать такие тревоги, как *"Предупреждение: температура ниже 10 градусов более часа"*.

Параметр **Задержка** применяется, когда в поле **Режим** выбрано **Состояние (ложь/истина)**. Он определяет интервал между временем, когда выражение триггера становится TRUE и временем поднятия тревоги. Если **Задержка** не равна нулю, сервер ждет определенное количество секунд до *активации* триггера и поднятия тревоги. Если **Выражение** становится снова FALSE до окончания **Задержки**, тревога не поднимается и счетчик времени сбрасывается.

Независимо от значения **Задержки**, выражение триггера заново вычисляется каждый раз за **Период проверки** (который также является параметром триггера -- см. выше). Например, если **Период проверки** задан на 10 секунд, а время задержки - 15 секунд, тревога активируется спустя 20 секунд после того, как выражение было оценено как TRUE в первый раз.

Триггер переменной активируется, если **Выражение** является TRUE в течение времени большего, чем **Задержка**. Он *деактивируется*, если:

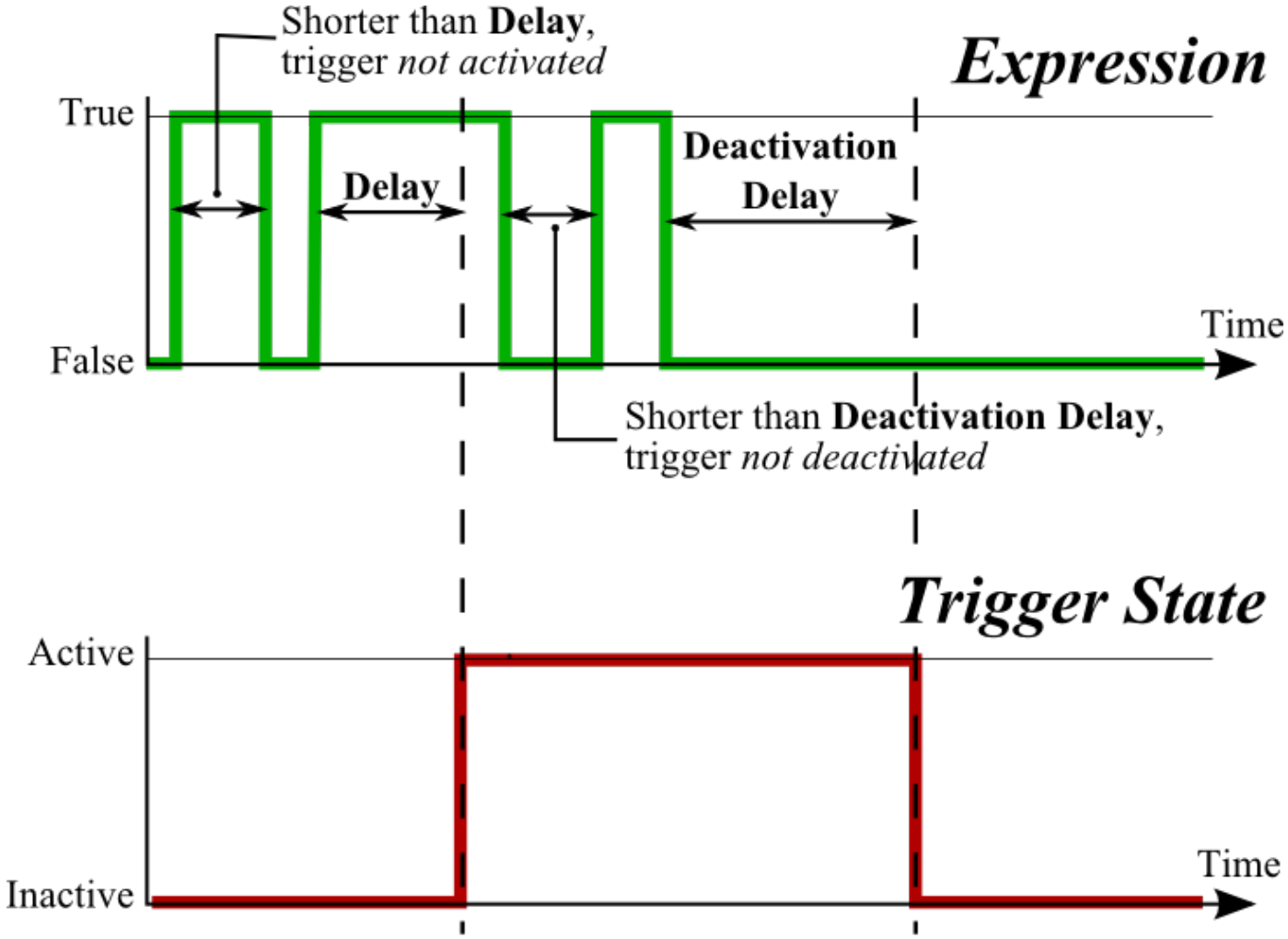
- **Выражение деактивации** не задано, и **Выражение** является FALSE не дольше **Задержки деактивации**.
- **Выражение деактивации** задано и является TRUE не дольше, чем время **Задержки деактивации**.

Активированный триггер приводит тревогу в состояние **Активный** и добавляет Активный экземпляр[804].

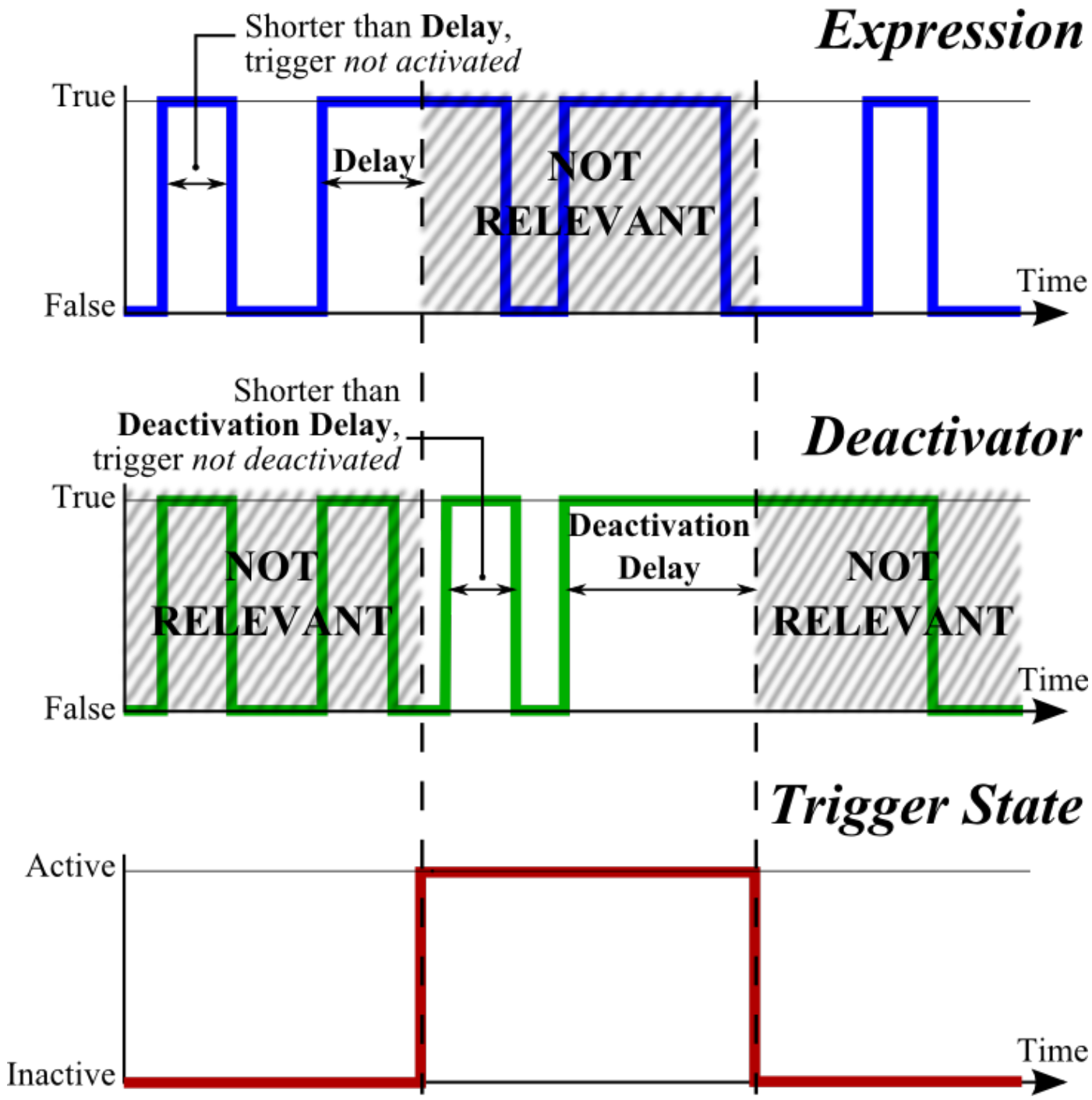
Комбинация **Задержки** и З**адержки деактивации** называется гистерезисом.

Гистерезис - задержка между действием и его причиной

Изображение, приведенное ниже, показывает, как активируется/деактивируется триггер переменной, когда **Выражение деактивации** не задано.



Когда **Выражение деактивации** задано, то активация/деактивация триггера происходит по схеме, изображенной далее:



Пример триггера переменной с гистерезисом:

Вернемся к вышеупомянутому примеру о контроле температуры.

Мы хотим установить триггер, который:

- Поднимает тревогу, когда температура выше 50 градусов более 5 минут.
- Деактивирует тревогу, когда она активна и температура ниже 35 градусов более 20 минут.

Параметры триггера могут быть следующими:

Маска контекста	`users.user123.devices.temperature_sensor` (Данная маска соответствует одному контексту терминала данных)[1494].
Переменная	`currentTemperature`
Выражение	`{celsiusTemperature} > 50`

Контролировать изменения состояния	disabled
Период проверки, секунды	10
Задержка, секунды	300
Уровень тревоги	Warning
Выражение деактивации	{celsiusTemperature} < 35
Задержка деактивации, секунды	1200

Распознавание "биения"

"Колебанием" называется ситуация, когда **Выражение** триггера меняет свой результат слишком быстро - постоянно переключает между TRUE и FALSE. Это может происходить ввиду следующих причин - например, если устройство постоянно перезагружается, его онлайн статус будет постоянно меняться.

AtomMind может обнаружить "колебание" триггера переменной. Это происходит посредством анализа предыдущих результатов оценки **Выражения** относительно количества произошедших изменений состояния. AtomMind хранит историю 101 последней проверки и анализирует изменения внутри истории. Если выражение TRUE не меняется на FALSE или наоборот в течение последней 101 проверки состояния (т.е. 100 изменений состояния), процент "колебания" будет равен 0%. Если все проверки имеют различные состояния, процент "колебания" равен 100%.

Распознавание "биения" контролируется тремя параметрами:

- Включить распознавание "биения"
- Порог активации
- Порог деактивации

Первый параметр должен быть включен, чтобы разрешить тревоге определять наличие "биения". При его включении триггер хранит результат расчета **Выражения**, совершаемый во время каждого **Периода проверки** и заново рассчитывает процент "биения" при последующей проверке.

Тревога "биение" возникает, когда в истории находятся более, чем 11 проверок (10 изменений состояния) и процент "биения" превышает **Порог активации**. Тревога "биение" обрабатывается и доставляется подобно другим тревогам, ее единственными отличиями являются причина и данные тревоги.

Триггер будет оставаться в состоянии **"Обнаружено биение"** до тех пор, пока процент "биение" не упадет ниже **Порога деактивации.**

"БИЕНИЕ" И АКТИВАЦИЯ ТРИГГЕРА

"Биение" и обычная активация триггера совершенно независимы друг от друга. Это означает, что триггер может быть одновременно в состояниях **"Активен"** и **"Обнаружено биение"**. Это также означает, что обычная тревога и тревога "биение" могут подниматься триггером в любом порядке.

Распознавание "биения" не вызовет обычную активацию, которая происходит, когда результат **Выражения** остается TRUE дольше времени **Задержки**.

Анализ внешних таблиц инцидентов

Иногда AtomMind Server извлекает таблицу инцидентов из внешнего источника. Появляется необходимость активизировать отдельную тревогу для каждой строки внешней таблицы. В этом случае, нужно настроить **Выражение ключа записи** триггера переменной. Это выражение будет оцениваться независимо для каждой строки значения **Переменной**, и оно должно вернуть уникальный ID инцидента, представленного строкой. В большинстве случаев, это выражение указывает на колонку "ID" внешней таблицы инцидентов.

В данном случае AtomMind Server активизирует отдельный экземпляр для каждого нового ID (значения), возвращенного **Выражением ключа записи**. Этот экземпляр будет активен пока во внешней таблице существует строка с этим ID. Если определенный ID на некоторое время исчезает из таблицы, а затем появляется снова, создается новый экземпляр тревоги.

В случае использования **Выражения ключа записи**, **Выражение** основного триггера проанализирует отдельную строку значения **Переменной**, а не целое табличное значение. Если оно вернет false, строка не будетактивизировать новую тревогу. Это удобно для отфильтровывания инцидентов с низкой уровнем критичности от инцидентов, которые не должны обрабатываться AtomMind Server.

Сообщение триггера

В дополнение к выражению **Сообщения** тревоги, каждый триггер переменной имеет свое собственное выражение **Сообщения триггера**. Это выражение разрешается в строку, когда активирована тревога. Итоговая строка становится частью События тревоги 790, и содержит специализированную информацию о причине активации тревоги или других обстоятельствах.

Среда вычисления

Среда вычисления 112 выражения триггера переменной и выражения деактивации:	
Контекст по умолчанию 119	Контекст, чья переменная проверяется.
Таблица данных по умолчанию 120	Текущее значение переменной триггера.
Ряд по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

12.9.2 Триггеры события

Триггер события активирует тревогу, когда возникает определенное событие 73 контекста 41 и оно удовлетворяет условиям, обозначенным в настройках триггера. Данное событие может быть сформировано самим AtomMind Server или может исходить из терминала данных 497.

Свойства триггеров события описаны здесь 796. Каждый триггер заставляет AtomMind Server "слушать" событие, определенное настройкой **Событие** в каждом контексте, соответствующем настройке **Маска Контекста**. При обнаружении такого события сервер вычисляет выражение 112, определенное настройкой **Выражение**. Данное выражение обычно ссылается на данные, относящиеся к событию, но оно также может ссылаться на другие данные, такие как значения контекстных переменных 61.

Если параметр **Выражение** не определен, каждое наступление события активирует тревогу.

Пример триггеров события:

Предположим мы запускаем систему мониторинга транспорта, которая имеет Device на различных транспортных средствах предприятия. Данные Device создают события, такие как событие **удар**, которое формируется, если транспортное **средство** ударяется или разбивается о что-либо. Таким образом, **Маска контекста** будет соответствовать Device, контролирующему транспортное средство, появится событие **удар**, и мы также можем получить выражение, относящееся к полю **сила** события **удар**, которое приведет к активации триггера, если сила удара превысит заданное значение.

В данном случае параметры триггера события будут следующими:

Маска контекста	`users.user123.devices.vehicle12_controller` (Данная маска соответствует одному контексту терминала данных) 1494.
Событие	`impact`
Выражение	`{strength} > 5.5`

(Конечно, **удар** и **сила** являются всего лишь теоретическими событиями, отправляемыми некоторыми составными Device - они не являются встроенными системными событиями AtomMind Server).

Пример выражения триггеров события: `contains({message}, "FAILED LOGIN") && {facility} == 4 && {level} == 5`

Это выражение активирует тревогу, если получено сообщение от сервера Syslog:

- Содержит строку `FAILED LOGIN`
- Имеет `5`ый уровень

- Генерируется средством Syslog с ID = 4

Триггер множественных событий

Параметры триггера **Счет** и **Период** работаю в паре, чтобы разрешить активацию триггера, только если событие произошло *X раз в течение последних Y секунд.*

Если Счет установлен на 1 (по умолчанию), любое событие, соответствующее **Выражению** триггера, приведет к активации триггера и тревоги. Если Счет установлен на 3 и Период равен 10 минутам, триггер будет активирован при возникновении события, если два предыдущих его возникновения произошли не более 10 минут назад и все три возникновения соответствуют **Выражению**.

Пример: Представим, что мы мониторим несколько серверов в сети. Одно событие "Аутентификация не состоялась", полученное от сервера, не определяет проблему, поскольку пользователь, возможно, просто опечатался, набирая пароль. Однако множественные события, полученные от одного сервера в течение 10 минут, должны отправить тревогу о нарушениии защиты. Вот необходимая настройка триггера события для этого случая:

Маска контекста	`users.*.devgroups.servers.*`
Событие	`authenticationFailure`
Выражение	
Счет	`20`
Период	`10 Minutes`

Связь событий

Связь событий является техникой для упорядочения огромного количества событий и выявления некоторых из них, действительно важных среди все этой массы информации.

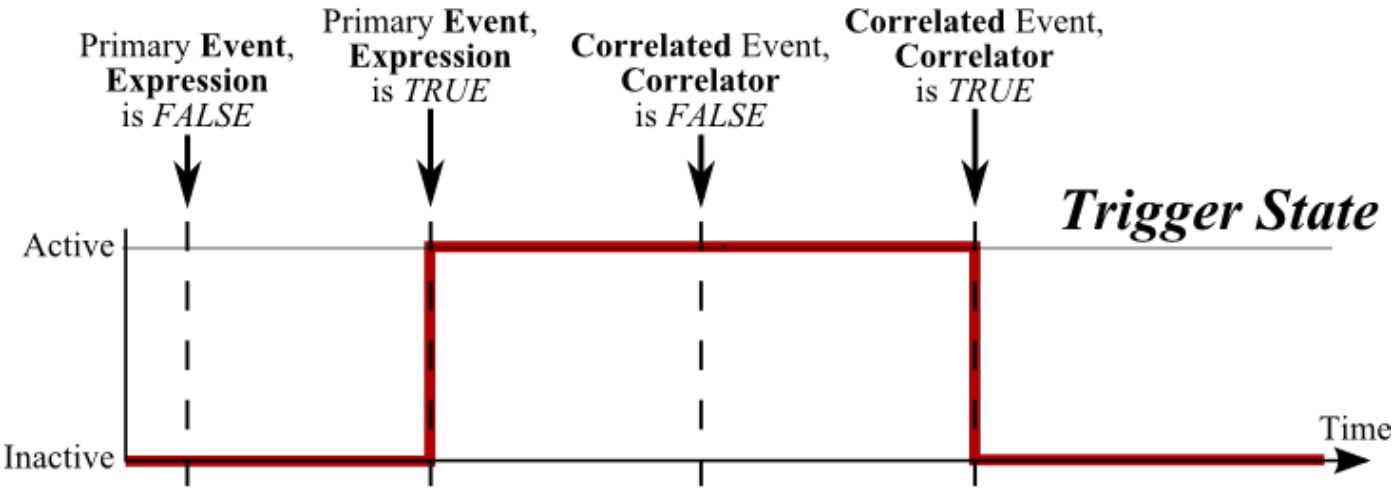
Что касается тревог AtomMind, связь события является способом активировать 804 тревогу триггером события, когда возникает одно ("основное") событие, и деактивировать тревогу при другом ("связанным") событии.

Чтобы включить связь событий для определенного триггера события, задайте имя в поле **Деактивирующее событие** в настройках триггера. В данном случае:

- Триггер события (как и сама тревога) будет *активирован* событием, указанным в настройке **Событие** (которое будет является "основным" в данном случае).
- Он будет деактивирован, если **Деактивирующее событие** происходит в том же контексте, что активирующее **Событие**, и **Выражение деактивации** является TRUE. Может быть так, что возникает **Деактивирующее событие**, но **Выражение деактивации** является FALSE, таким образом, триггер остается активированным.

Как только триггер был активирован, он переводит тревогу в состояние 802 **Активен** и добавляет Активный экземпляр 804.

На приведенной далее схеме изображен процесс активации/деактивации триггера события основными и связанными событиями:



Настройки **Счет** и **Период** также применяются к коррелированным событиям. Например, если **Счет** установлен на 5, а **Период** - на 1 минуту, триггер деактивируется, только если пять событий заданных в поле **Деактивирующее событие** возникают в течение одной минуты и все пять **Выражение деактивации** являются TRUE.

Примеры триггеров событий с использованием корреляции событий:

Иногда нам может потребоваться посмотреть тревогу в списке активных тревог[833], когда определенное устройство находится офлайн (недоступно для AtomMind Server). Предположим, что мы имеем события **Подключение** и **Отключение** в контексте[1494] Device, мы можем использовать следующие настройки триггера, чтобы принудительно оставить тревогу активной, когда устройство отключено:

Маска контекста	`users.user123.devices.critical_device`
Событие	`disconnection`
Деактивирующее событие	`connection`

Данный триггер поднимет тревогу по событию **отключение** и отменит ее по событию **подключение**.

При отсутствии событий подключения/отключения в контексте %dt%>, мы все равно можем использовать событие contextStatusChanged[83] с той же целью. В данном случае установка триггера будет сложнее:

Маска контекста	`users.user123.devices.critical_device`
Событие	`contextStatusChanged`
Выражение	`{status} == 0` (Assuming that status 0 means "device is disconnected")
Деактивирующее событие	`contextStatusChanged`
Выражение деактивации	`{status} == 1` (Assuming that status 1 means "device is connected")

Данный триггер будет активирован, когда возникает событие **contextStatusChanged** и значение поля **Статус** данных события установлено на ноль. То же событие со статусом 1 деактивирует триггер.

Сообщение триггера

Вдобавок к выражению тревоги **Сообщение** каждый триггер события имеет собственное выражение **Сообщение Триггера**. Это выражение преобразуется в строку, когда появляется тревога. Строка-результат становится частью События тревоги[790], сохраняя любую пользовательскую информацию о причине тревоги или других обстоятельствах.

Пример выражения сообщения триггера события: `'Device-provided custom SNMP trap field:' + cell({variableBindings}, "myMIBDataField")`

Это выражение можно использовать для вставки значения определенной пользовательской привязки переменной SNMP-ловушки[638] в тревогу. Это выражение сначала ссылается на вложенную таблицу

SNMP-ловушки таблица данных события 74, используя ссылку `{variableBindings}`, затем извлекает значение поля `myMIBDataField` из первой строки таблицы.

Среда вычисления

Среда вычисления 112 выражения триггера события и выражения деативирующего события:

Контекст по умолчанию 119	Контекст, от которого было получено событие.
Таблица данных по умолчанию 120	Таблица данных, в которой содержатся данные события 73 для триггера события или деативирующего события.
Ряд по умолчанию 119	0
Переменные среды 123	

Имя переменной	Тип переменной	Описание
context	строка	Полный путь контекста события.
event	строка	Имя события.
level	целое	Уровень 75 события.
time	дата	Временная метка события.
acknowledgements	таблица данных	Таблица подтверждений 76 события.
enrichments	таблица данных	Таблица обогащений 76 события.

12.9.3 Событие тревоги

Когда поднимается тревога, в контексте тревог 1454 создается специальное AtomMind Server событие 73. Оно называется "Событие тревоги" или "Экземпляр тревоги" и используется в следующих случаях:

- Когда данное событие не подтверждено 73, оно называется "ожидающий экземпляр тревоги". Когда в тревоге присутствуют экземпляры, ожидающие подтверждения, то она переходит в состояние 802 **Активен**. Ее иконка меняется на  . Это похоже на принцип работы email клиента - ваш почтовый ящик отображается как "Не прочтен", если в нем имеются непрочитанные сообщения.
- История 73 событий тревог помогает проследить, когда и почему была поднята тревога.

Когда вы просматриваете ожидающие тревоги 1454, вы в действительности смотрите на историю всех событий для определенной тревоги. Доступ к данному действию разрешен только при включенной настройке **Разрешить ожидающие тревоги**.

Формат события тревоги:

Поле	Описание
Имя тревоги	Имя контекста тревоги 1454.
Описание тревоги	Описание контекста тревоги 1454.
Контекст	Путь контекста, для которого была поднята тревога, т.е. контекст, где произошло событие или изменение состояния, контролируемое триггером тревоги.
Объект	Событие (если тревога поднята триггером события) или переменная (если триггером переменной), которые вызвало тревогу.
Причина	Текстовое описание причины тревоги.

Сообщение	Сообщение о тревоге. Строка ввода вручную, которая должна представлять собой читаемое описание тревоги (т.е. "Пожар в доме"). См. раздел Конфигурация тревоги 795.
Триггер	Сообщение триггера. Дополнительное текстовое сообщение, передаваемое триггером тревоги.
Данные тревоги	Данные, связанные с событием, поднявшим тревогу. Данное поле имеет значение NULL (не определено), если причинами тревоги были состояние определенной переменной или изменение значения переменной. Для события **Вход в систему**, заданного в контексте **Пользователи**, данная таблица данных будет содержать одну запись с двумя полями: **Имя пользователя** и **Права доступа**.

Пример события тревоги:

Field	Value
Alert Name	deviceOffline
Alert Description	Device Offline
Context	users.admin.devices.192_168_1_108_ip
Entity	status
Cause	State of variable 'Device Status' in context 'WIN-2UR4L8XA18I (IP Network Host)'
Message	WIN-2UR4L8XA18I (IP Network Host) is offline
Trigger	<Not set>
Data	Device Driver=IP Network Host, Last Synchronization Time=Thu Sep 03 02:09:42 MSD 2009, Connection St...

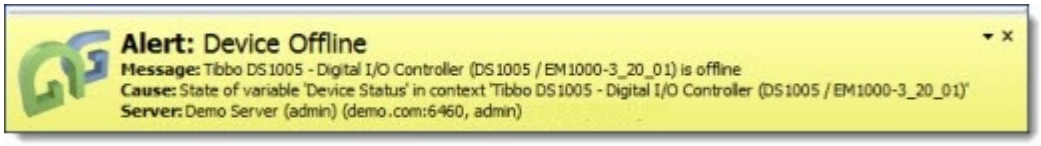
12.9.4 Оповещения

Оповещения используются для информирования пользователя и других заинтересованных сторон о возникновении и эскалации тревоги.

Настройки уведомлений описаны в разделе Настройки уведомлений

Всплывающие оповещения о тревогах и звуковые оповещения

Пользователь 478 AtomMind Server, у которого была активирована тревога, получает уведомление, если включена настройка **Показывать всплывающие окна оповещений пользователю**, и данный пользователь вошел в пользовательский интерфейс AtomMind Server. Например, в AtomMind Client пользователю показывается такое всплывающее окно:



Всплывающее окно тревожного оповещения содержит:

- Описание тревоги
- Причину тревоги
- Сообщение тревоги
- Сообщение триггера
- Сервер, от которого была получена тревога.

Всплывающие окна тревожных уведомлений исчезнут автоматически после определенного времени, которое зависит от степени серьезности ошибки. Тем не менее, тревогу можно закрыть в любой момент нажатием клавиши "Закрыть".

Всплывающее окно тревожного уведомления обеспечивает доступ к следующим опциям:

- **Закрепить на экране**. Отключает автоматическое закрытие всплывающего окна тревоги.
- **Конфигурировать**. Открывает конфигурацию тревоги.
- **Показать данные тревоги**. Показывает детали тревоги в Редакторе таблицы данных 382.
- **Закрыть все**. Закрывает все видимые на данный момент всплывающие окна тревог.

Если уведомление пользователя включено, вы можете задать **звук уведомления** (.WAV или .MP3 file), которые будет проигрываться в пользовательском интерфейсе AtomMind Server до появления всплывающего окна.

ПОДТВЕРЖДЕНИЕ ТРЕВОГ

Если включен параметр **"Требуется подтверждение"**, пользователю предлагается ввести текст подтверждения в это же всплывающее окно тревоги. Вот как это выглядит в AtomMind Client:

ВРЕМЯ СУЩЕСТВОВАНИЯ ВСПЛЫВАЮЩИХ ОПОВЕЩЕНИЙ О ТРЕВОГЕ ПО УМОЛЧАНИЮ

Если определена опция **Время существования пользовательских уведомлений**, длительность видимости окна всплювающего сообщения зависит от уровня критичности тревоги:

- 10 секунд для тревог уровня None
- 20 секунд для уровня Notice
- 300 секунд для тревог уровня Info
- 3600секунд для тревог уровня Warning
- Тревоги уровня Error и Fatal не скрываются автоматически

Отправка E-mail уведомлений

Тревожные e-mail уведомления будут отправлены только тогда, когда настройки почтового сервера правильно сконфигурированы и включена опция отправки сообщений. Для более подробной информации см. Настройки отправки сообщений 1428 в разделе "Общая конфигурация AtomMind Server".

AtomMind Server может подготовить и отправить *e-mail сообщение тревожного уведомления.* Данное сообщение может быть отправлено трем категориям адресатов:

- Владельцу тревоги
- Другому (другим) пользователю (пользователям) 478 AtomMind Server
- На заданный(ые) e-mail адрес(а)

Отправка сообщения владельцу тревоги возможна при включении настройки **"Отправить E-mail сообщение владельцу тревоги"**. Таблица **получателей E-mail сообщений** содержит список пользователей AtomMind Server, которым будет отправлено e-mail уведомление.

В список получателей могут входить только те пользователи, чей e-mail адрес задан в аккаунте пользователя.

Также возможно отправлять e-mail уведомления на заданные e-mail адреса. Данные адреса должны быть включены в список **"Дополнительные адресаты E-mail сообщений"** и разделены запятыми.

Содержание email сообщения:

- Описание тревоги
- Событие, активирующее тревогу
- Контекст, в котором определено событие
- Сообщение тревоги
- Время тревоги
- Данные события, которое активировало тревогу (если она была поднята триггером события 787) или значение переменной (если тревога была поднята триггером переменной 782).

Данное email сообщение выглядит таким образом:

Server Alert

Alert	Test
Cause	State change of variable 'Device Server Information' in context 'admin.c1 : Auto-registered Device Server'
Value of Variable	Owner Name=admin, Device Server Name=c1, Password=, Description=Auto-registered Device Server, Register in DNS=false, Blocked=true, Inband commands allowed for client=false, Time Zone=GMT+03:00, Europe/Moscow, with DST, Data Transfer/Device Plugin=Transparent Data Routing (Link Service)
Expression	{blocked}
Current State	true
Previous State	false
Time	Wed May 21 15:35:42 MSD 2008

Простая текстовая версия email сообщения о тревоге:

Alert 'AuthKey Problem' raised, caused by event 'Status changed' in context 'AuthKey'
Message: AuthKey is not available or returns an error code
Time: Fri Dec 07 18:30:00 MSK 2007
Data: status=1; comment=Not Available

Отправка SMS оповещений

Отправка оповещений через SMS возможна, только если настройки отправки SMS правильно сконфигурированы и включена опция отправки SMS. Для более подробной информации см. Настройки отправки SMS сообщений 1437 в разделе "Общая конфигурация AtomMind Server".

AtomMind Server может отправлять оповещения через SMS на определенное количество мобильных телефонов. Установить адресаты возможно при помощи настройки "**Адресаты SMS**", расположенной в разделе Настройки уведомлений 799.

Номер отправителя определяется опцией общей конфигурации **"Номер отправителя"**.

Пример: Для тревоги "**Предупреждение об ударе**", которая информирует оператора системы мониторинга транспорта о произошедшем ударе транспортного средства путем отправки следующего SMS уведомления:

```
Alert: Impact Warning
```

Также возможно отправлять тревожные SMS уведомления через **GSM/GPRS модем**:

- Подключите AtomMind Server к модему, использующему драйвер устройства типа "модем 587".
  Убедитесь в правильности настроек модема и учетной записи его устройства.
- Добавьте новое автоматическое корректирующее действие 800 к тревоге.
- Укажите **Контекст** корректирующего действия для учетной записи вашего модема.
- Измените **Действие** на `Send SMS`.
- Введите номер адресата и текст SMS в **Параметрах**.

12.9.5 Корректирующие действия

Когда возникает ошибка, для нее часто бывает необходимо одно специфичное средство. Например, когда остается мало доступной памяти она устройстве, необходима загрузка или очистка его внутренней базы данных. Это всегда делается через одно действие, и никак иначе, не через отключение устройства, например, или запуск сервосистемы.

Из-за такой предсказуемости, это действие можно автоматизировать. Любое системное действие 87, доступное в интерфейсе пользователя, можно запускать автоматически в ответ на тревогу. Таким образом это действие становится *корректирующим действием*.

Корректирующие действия можно также использовать для отправки нестандартных уведомлений. Например, корректирующее действие может запустить внешнее приложение для отправки сообщения на пейджер или бипер (подробности здесь 1564). Другими примерами являются отправка Unix syslog сообщения или события trap протокола SNMP, используя специальные опции в составе расширений сетевого управления AtomMind.

Типы корректирующих действий

Действия могут запускать в интерактивном и неинтерактивном режимах 99. В неинтерактивном режиме вход всех действий определен заранее, и выход действия отправляется в журнал 166 сервера или записывается в историю событий 75. Данный режим также используется планировщиком 823. В интерактивном режиме связь между пользователем и действием осуществляется при помощи UI процедур 88. Пользователь должен войти в пользовательский интерфейс AtomMind Server (например, AtomMind Client) для выполнения интерактивных действий. Для более подробной информации см. раздел режимы выполнения действия 99.

Некоторые интерактивные корректирующие действия:

- Запуск очистки базы данных, спросив перед этим оператора: “Вы уверены?”
- Перезапуск критически важного устройства только после получения подтверждения от оператора

Некоторые автоматические корректирующие действия:

- Подготовка отчета о состоянии устройства, которое подняло тревогу, и отправка на e-mail
- Выполнение внешнего приложения для исправления ошибки
- Создание нового тикета в системе службы поддержки

АВТОМАТИЧЕСКИЕ КОРРЕКТИРУЮЩИЕ ДЕЙСТВИЯ

Автоматическое (также называемое автономное или неинтерактивное) выполнение корректирующих действий контролируется свойством Автоматические корректирующие действия 800. Это свойство позволяет конфигурировать множество событий для запуска при тревоге.

Таблица автоматических корректирующих действий имеет несколько полей:

- Каждое действие запускается из каждого контекста, соответствующего параметру **Маска контекстов**.
- Имя действия задается параметром **Действие**.
- **Параметры -** это список параметров, специфичных для действия (чтобы не запутаться, просто помните, что **Параметры** в данном случае - это параметр сам по себе. Понятно?). Эти параметры используются для замены вводимой пользователем информации, когда механизм обработки тревоги выполняет действие в неинтерактивном режиме. Например, если данное действие требует подтверждения (т.е. спрашивает у пользователя что-то типа "Удалить запрос?" и позволяет нажать **OK** или **Отменить**), это поле будет содержать параметр **Удалить запрос?** с двумя возможными опциями: **OK** или **Отменить**.
- **Условие -** это выражение 112, которое, если оно определено и вычислено как FALSE, подавляет выполнение корректирующего действия.

Для просмотра истории выполнения автоматических корректирующих действий и ошибок, обнаруженных ими во время выполнения, используйте действие Отслеживать соответствующие события 109 в Контексте тревог 1454.

ИНТЕРАКТИВНЫЕ КОРРЕКТИРУЮЩИЕ ДЕЙСТВИЯ

Действия, выполняемые в интерактивном режиме, перечислены в таблице Интерактивные корректирующие действия 801. Каждое действие имеет два параметра: **Маска контекстов** и **Действие**, принцип работы которых такой же, как и в неинтерактивном режиме (см. выше). Колонка **Параметры** позволяет определить некоторые интерактивные параметры выполнения, такие как расположения окон.

Интерактивные корректирующие действия выполняются только когда включены всплывающие оповещения 791 о тревогах.

12.9.5.1 Обработка параметров действий

Параметры 827 ввода корректирующих действий заранее определяются на стадии создания тревоги и сохраняются в базе данных 692. Однако часто необходимо изменить/обновить эти параметры, когда появляется тревога.

Например, если нужно отправить пользовательские e-mail или SMS сообщения при появлении тревоги, их текст должен составляться при действительном возникновении тревоги и включать в себя элементы данных, описывающих причину.

Чтобы сконфигурировать параметры динамических корректирующих действий, нужно:

- Открыть таблицу параметров для редактирования. Тогда можно добавлять/модифицировать их привязки 397.
- Добавить в таблицу одну или более привязок.

Эти привязки рассчитываются, когда появляется тревога и выполняются ее корректирующие действия. Они обновляют значение заранее определенных параметров действия.

См. Привязки параметров ввода 1001 для получения более подробной информации.

12.9.6 Конфигурация тревоги

Данный раздел содержит информацию о свойствах конфигурации тревоги.

Все свойства, которые будут описываться далее, доступны для просмотра через действие Конфигурировать 105 в контексте тревог 1454.

12.9.6.1 Свойства тревоги

Далее приведены базовые опции тревоги.

<table>
<tr><th>Описание поля</th><th>Имя поля</th></tr>
<tr><td>Имя тревоги. Имя контекста тревоги 1454, необходимый для ссылки на данную тревогу из других частей системы. Оно должно соответствовать соглашениям по наименованию 42.</td><td>name</td></tr>
<tr><td>Описание тревоги. Текстовое описание тревоги. Также является описанием 43 контекста тревоги.</td><td>description</td></tr>
<tr><td>Активная тревога. Отключение данной функции деактивирует все триггеры тревоги и она никогда не будет поднята.</td><td>enabled</td></tr>
<tr><td>Сообщение. Сообщение тревоги. Будет показано во всех тревожных уведомлениях, включая визуальные оповещения, e-mail уведомления, события тревоги и т.д. Это выражение 112, поэтому статические сообщения должны заключаться в кавычки (например, <code>'Some message'</code>).
<table>
<tr><td colspan="2">Среда вычисления 114 выражения сообщения тревоги:</td></tr>
<tr><td>Контекст по умолчанию 119</td><td>Контекст источника тревоги.</td></tr>
<tr><td>Таблица данных по умолчанию 120</td><td>Таблица данных события тревоги 790.</td></tr>
<tr><td>Строка по умолчанию 119</td><td>0</td></tr>
<tr><td>Переменные среды 123</td><td>Только стандартные 123 переменные.</td></tr>
</table></td><td>message</td></tr>
</table>

Требуемый уровень прав доступа источника. Определяет уровень прав доступа 486, который необходим пользователю в контексте источника тревоги для просмотра тревоги. Значение по умолчанию **Отсутствует** означает, что пользователь может совсем не иметь доступа к контексту источника тревоги, но может видеть и обрабатывать тревогу. Уровень прав доступа, определенный этой опцией, сохраняется в базе данных 692 с **каждым** экземпляром события. Таким образом, смена этой опции может сделать новые экземпляры доступными/недоступными для операторов, но доступность экземпляров событий тревог, сохраненных ранее, не будет подвергаться воздействию.	permissions
Сохранять состояние. Эта настройка определяет, будет ли состояние триггера тревоги сохраняться в базе данных сервера и восстанавливаться при перезапуске сервера. Настройка по умолчанию отключена, т.к. может приводить к излишнему потреблению ресурсов базы данных в высоконагруженных системах.	persistState

Доступ к данной информации осуществляется через переменную childInfo 1455.

12.9.6.2 Триггеры события

Далее приведен список триггеров события 787.

Описание поля	Имя поля
Маска контекстов. Данная маска 44 определяется во время обработки триггера. Если событие, определенное параметром **Событие**, возникает в любом контексте, соответствующем маске, триггер активирует тревогу. В большинстве случаев данная маска указывает на определенный контекст, т.е. является скорее путем контекста, нежели контекстной маской.	mask
Событие. Имя отслеживаемого события.	event
Выражение. Выражение AtomMind 112, которое должно разрешаться в значение TRUE для активации триггера. Может содержать ссылки 117 на ячейки таблицы данных 49, относящиеся к **событию**.	filter
Выражение ключа записи. Выражение AtomMind 112, которое используется для определения ключа (обычно это определенное поле таблицы **События**), когда таблица **События** содержит несколько записей, и их нужно проверять построчно.	recordKeyExpression

Среда вычисления 114 выражения ключа записи:

Контекст по умолчанию 119	Контекст-источник тревоги.
Таблица данных по умолчанию 120	Таблица данных **События**.
Ряд по умолчанию 119	Текущий ряд таблицы данных **События**.
Переменные среды 123	Только стандартные 123 переменные.

Уровень тревоги. Уровень 73 события тревоги 790. В дополнение к заданным значениям может быть установлен на **Как у события** (настройка по умолчанию). В данном случае уровень события тревоги совпадает с уровнем события, которое активировало тревогу.	level
Деактивирующее событие. Имя деактивирующего события. Возникновение этого события деактивирует триггер, если выражение, определенное в поле **Выражение деактивации** является TRUE.	correlated

Выражение деактивации. Выражение AtomMind 112, которое должно разрешаться в значение TRUE для деактивации триггера при возникновении события, заданного в поле **Деактивирующее событие**. Может включать в себя ссылки 117 на ячейки таблицы данных 49, относящейся к событию, заданному в поле **Деактивирующее событие**.	correlator
Количество. Количество событий, которые должны возникнуть в течение времени, определенного в поле **Период** для активации или деактивации триггера.	count
Период. Период времени, в течение которого должны возникнуть события, активирующие или деактивирующие триггер.	period
Сообщение триггера. Сообщение триггера обычно используется для указания корневой причины тревоги. Это выражение 112, поэтому статические сообщения должны заключаться в кавычки (например, `'Текст сообщения'`). Среда вычисления 114 выражения сообщения триггера тревоги: Контекст по умолчанию 119 — Контекст источника тревоги. Таблица данных по умолчанию 120 — Таблица данных событий тревоги 790. Строка по умолчанию 119 — 0 Переменные среды 123 — Только стандартные 123 переменные.	message

Доступ к данной информации осуществляется через переменную eventTriggers 1456.

12.9.6.3 Триггеры переменной

Каждый триггер переменной периодически проверят значение переменной каждого контекста, соответствующего маске 44,заданной настройкой **Маска контекста**. Период опроса определен в секундах параметром **Период проверки**. Из контекста выбирается таблица данных 49, в которой содержится значение переменной, затем, согласно настройке **Режим**, происходит следующее:

Данное свойство определяет список триггеров переменной 782.

Описание поля	Имя поля
Маска контекста. Значение данной маски 44 разрешается во время обработки триггера. Будет контролироваться значение переменной, определяемое параметром **Переменная** каждого контекста, соответствующего маске. В большинстве случаев данная маска указывает на определенный контекст, т.е. является скорее путем контекста, нежели контекстной маской.	mask
Переменная. Имя контролируемой переменной контекста.	variable
Выражение. Выражение AtomMind 112, используемое для проверки значения переменной. Результат данного выражения будет обрабатываться согласно параметру **Режим**.	expression
Режим. Если выбрано **Состояние (ложь/истина)**, триггер срабатывает, когда результат параметра **Выражение** меняется с FALSE на TRUE. Если выбрано **Изменение состояния (любое значение)**, триггер срабатывает во время любого изменения результата поля **Выражение** (с одного на другое, необязательно TRUE/FALSE). Если выбрано **Динамические пороги (число),** триггер срабатывает при выходе результата вычисления выражения, заданного в поле **Выражение** за пределы, определенные в настройке **Пороги.** Все это более подробно рассмотрено в разделе Триггеры переменной 782.	mode

Выражение ключа записи. Выражение AtomMind 112, используемое для определения ключа (как правило это определенное поле **Переменной**), когда **Переменная** содержит несколько записей, и их нужно проверять построчно. Относится только к режиму **Состояния (ложь/истина)**. Среда вычисления 114 выражения ключа записи: Контекст по умолчанию 119 — Контекст-источник тревоги. Таблица данных по умолчанию 120 — Текущее значение **Переменной**. Ряд по умолчанию 119 — Текущий ряд значения **Переменной**. Переменные среды 123 — Только стандартные 123 переменные.	recordKeyExpression
Период проверки. Временной интервал между проверками **Переменной**.	period
Задержка. Временной интервал между моментом, когда **Выражение** стало TRUE и временем активации тревоги.	delay
Уровень тревоги. Уровень 73 данного события тревоги 790.	level
Выражение деактивации. Выражение деактивации тревоги. Вычисляется, когда триггер активен, с периодичностью заданной в поле **Период проверки.**	deactivator
Задержка деактивации. Временной интервал между моментом, когда **Выражение деактивации** стало TRUE и временем деактивации тревоги.	deactivationDelay
Пороги. Выражения для нижнего и верхнего порога. Если результат вычисления выражения, заданного в поле **Выражение** меньше нижнего порога или больше верхнего, тревога будет активирована.	baselining
Биение. Контролирует обнаружение биения 786 для данного триггера.	flapping
Сообщение триггера. Обычно используется для указания корневой причины тревоги. Alert Trigger Message Expression Resolution Environment 114: Контекст по умолчанию 119 — Контекст источника тревоги. Таблица данных по умолчанию 120 — Таблица данных события тревоги 790. Строка по умолчанию 119 — 0 Переменные среды 123 — Только стандартные 123 переменные.	message

Данная информация доступна через переменную variableTriggers 1456.

12.9.6.4 Настройки оповещений

Данное свойство определяет правила тревожных уведомлений.

Описание поля	Имя поля
Показать всплывающее окно владельцу. Всплывающее окно показывается пользователю 478, которому принадлежит тревога, если он осуществил вход в пользовательский интерфейс AtomMind Server (такой как AtomMind Client) при активации тревоги.	notifyOwner
Требуется подтверждение. Пользователь обязан подтвердить 806 тревогу при показе тревожного уведомления.	ackRequired
Пользовательская длительность отображения оповещения. Определяет время, в течение которого всплывающее окно тревоги будет видимым на экране.	lifetime
Звук оповещения. Звук, проигрываемый для владельца тревоги, до появления всплывающего окна.	sound
Отправить E-mail владельцу. Отправить e-mail сообщение владельцу тревоги на адрес, указанный в его аккаунте.	mailToOwner
Получатели E-mail. Список пользователей AtomMind Server, которые получат тревожные e-mail уведомления.	mailRecipients
Дополнительные получатели E-mail. Список e-mail адресов, разделенных запятой. Уведомление о тревоге будет отправлено на каждый из этих адресов.	additionalMailRecipients
Получатели SMS. Список телефонных номеров, на которые будут отправлены тревожные уведомления. Данный список может содержать номера телефонов пользователей 478 AtomMind Server, указанные в настройках их учетных записей 481, а также обычные номера телефонов. Все номера должны записываться в международном формате.	smsRecipients

Данная информация доступна через переменную notifications 1457.

12.9.6.5 Настройки эскалации

Данное свойство определяет опции обработки активных (неподтвержденных) 804 тревог и правила эскалации 805.

Описание поля	Имя поля
Разрешить ожидающие тревоги. Если данный параметр включен, то тревоги могут переходить в состояние "ожидания". Отключение данного параметра также отключает эскалацию.	pending
Эксалация по количеству ожидающих тревог. Включает эскалацию в зависимости от определенного количества активных экземпляров тревоги.	numberEscalation
Количество ожидающих тревог, необходимое для эскалации. Когда количество активных экземпляров тревоги (данного типа) превышает лимит, происходит эскалация тревоги.	numberThreshold
Эскалация по времени. Включает эскалацию, если экземпляры тревоги остаются активными дольше указанного временного периода.	timeEscalation
Время до эскалации (минут). Если какой-либо экземпляр тревоги остается неподтвержденным в течение данного времени, происходит эскалация тревоги.	timeThreshold

Данная информация доступна через переменную escalation 1457.

12.9.6.6 Автоматические корректирующие действия

Данное свойство определяет действия 87, выполняемые в неинтерактивном режиме при активации тревоги. Хорошим примером для иллюстрации применения данного свойства может стать выполнение 1564 внешнего приложения при возникновении тревоги.

Описание поля	Имя поля
Тип выполнения. Определяет, когда действие должно быть выполнено. Доступны следующие опции: • **Возникновение**. Выполняет действие каждый раз при создании экземпляра тревоги. • **Активация**. Выполняет действие при активации триггера для определенного контекста. Схож с типом **Возникновение**, исключая то, что активация не произойдет для триггеров событий, которые не имеют коррелирующих событий. • **Деактивация**. Выполняет действие при деактивации триггера тревоги для определенного контекста. • **Эскалация**. Выполняет действие при эскалации тревоги. • **Деэскалация**. Выполняет действие при деэскалации тревоги. • **Подтверждение**. Выполняет действие при подтверждении экземпляра тревоги (экземпляра события тревоги). Корректирующие действия, которые инициируются на **подтверждении**, будут запущены лишь в случае, если один из ожидающих экземпляров 804 был подтвержден.	executionType
Маска контекста. Будет выполняться действие для всех контекстов, удовлетворяющих маске 44.	mask
Действие. Имя действия, которое должно быть выполнено.	action
Параметры. Данный список состоит из двух типов параметров: параметры выполнения 102, относящиеся к действию, и определенный заранее вход для различных Пользовательских процедур 88, таких как подтверждение 89, изменение данных 90 и т.д. Определенный заранее вход требуется для замены события входа пользователя, когда тревога выполняет действие в неинтерактивном режиме. Например, если автоматическое корректирующее действие требует подтверждения (т.е. выдает вопрос пользователю типа "Удалить запрос?" и предлагает нажать OK или Отменить), данное поле будет содержать параметр "Удалить запрос?" с двумя возможными вариантами на выбор: OK или Отменить.	input
Условие. Выражение 112, которое вычисляется перед выполнением корректирующего действия. Если выражение возвращает false, корректирующее действие пропускается. Среда вычисления 114 выражения условия: Контекст по умолчанию 119 — Контекст тревоги. Таблица данных по умолчанию 120 — Данные события тревоги 790. Строка по умолчанию 119 — 0 Переменные среды 123 — Только стандартные 123 переменные.	condition
Запускать из контекста источника. Если этот флаг активен, корректирующее действие запускается из контекста, откуда пришла тревога. **Пример:** Если тревога контролирует использование памяти приложения с утечками памяти, нужно выбрать корректирующее действие `Restart` и включить **Запустить из контекста источника**, чтобы перезапустить	runFromSource

приложение на сервере, инициирующем тревогу. Те же приложения на другом сервере затрагиваться не будут. Параметр **Маска контекстов** игнорируется, если включен **Запускать из контекста источника**. Однако необходимо выбрать **Маску контекстов**, которая в первую очередь *соответствует одному или всем возможным контекстам источников тревог*, поскольку это позволяет правильно выбрать **Действие** и его **Параметры**.	

Данная информация доступна через переменную alertActions 1458.

12.9.6.7 Интерактивные корректирующие действия

Данное свойство определяет действия 87, выполняемые в интерактивном режиме, когда поднимается тревога и установлено соединение между клиентом пользователя и сервером.

Описание поля	Имя поля
Тип выполнения. Определяет когда действие должно быть выполнено. Доступны следующие опции: • **Возникновение**. Выполняет действие каждый раз при создании экземпляра тревоги. • **Активация**. Выполняет действие при активации триггера для определенного контекста. Схож с **Возникновение**, исключая то, что активация не произойдет для триггеров событий, которые не имеют коррелирующих событий. • **Деактивация**. Выполняет действие при деактивации триггера тревоги для определенного контекста. • **Эскалация**. Выполняет действие при эскалации тревоги. • **Деэскалация**. Выполняет действие при деэскалации тревоги. • **Подтверждение**. Выполняет действие при подтверждении экземпляра тревоги (экземпляра события тревоги). Корректирующие действия, которые инициируются на **подтверждении**, будут запущены лишь в случае, если один из ожидающих экземпляров 804 был подтвержден.	executionType
Маска контекста. Будет выполняться действие из любого контекста, удовлетворяющего маске 44.	mask
Действие. Имя действия, которое должно быть выполнено.	action
Параметры. Параметры выполнения 102, относящиеся к действию.	input
Запускать из контекста источника. Если этот флаг проверяемый, корректирующее действие запускается из контекста, откуда пришла тревога. **Пример:** Если тревога контролирует использование памяти приложения с утечками памяти, нужно выбрать корректирующее действие `Restart` и включить **Запустить из контекста источника**, чтобы перезапустить приложение на сервере, инициирующем тревогу. Те же приложения на другом сервере затрагиваться не будут. Параметр **Маска контекстов** игнорируется, если включен **Запускать из контекста источника**. Однако необходимо выбрать **Маску контекстов**, которая в первую очередь *соответствует одному или всем возможным контекстам источников тревог*, поскольку это позволяет правильно выбрать **Действие** и его **Параметры**.	runFromSource

Данная информация доступна через переменную interactiveActions 1458.

12.9.7 Состояния тревоги

Тревоги могут быть в одном из следующих состояний:

- Включена ()
- Отключена ()
- Активна ()
- Эскалирована ()

Включенные/отключенные тревоги

Изначально только что созданная тревога находится в состоянии **Активна**. Это означает, что все триггеры включены и могут поднять тревогу. Вы можете выключить тревогу (т.е. перевести ее в состояние **Отключена**), отключив базовую опцию 795 **Тревога включена**. Активация тревоги никогда не произойдет, потому что все триггеры неактивны.

Активные тревоги

Тревога *активна*, если::

- Один из ее триггеров события 787 активен, т.е. основное событие произошло, а коррелированное еще нет.
- Один из ее триггеров переменных 787 активен, т.е. были удовлетворены все условия для активации, после которой не была осуществлена деактивация.
- Один из ее триггеров переменных 787 в состоянии "Обнаружено колебание".
- Присутствуют ожидающие 804 (неподтвержденные) тревоги.

Причина активации отображается во всплывающей подсказке при наведении мыши на тревогу в пользовательском интерфейсе AtomMind Server.

Эскалированные тревоги

Описание состояния **Эскалирована** см. в разделе Эскалация тревоги 805.

12.9.8 Жизненный цикл тревог

Тревога является довольно сложным объектом с многоступенчатым жизненным циклом. В этом разделе представлен обзор доступных состояний тревог и их переходов, в то время как последующие статьи описывают подробности.

Экземпляры триггеров тревог

Каждая тревога может иметь несколько триггеров (типа переменная 782 и/или событие 787), и каждый из них имеет настройку **Маска контекстов** , направляющую его в группу контекстов (источники тревог). Экземпляр каждого триггера создается для отслеживания каждого контекста, соответствующего маске. Например, если у тревоги два триггера, первый из которых имееет 10 контекстов, соответствующих его маске, и второй, имеющий 20 контекстов, тревога будет иметь в общей сложности 30 экземпляров триггеров.

Поднятие тревоги

Каждый экземпляр триггера выполняет мониторинг его "равноправных" контекстов по-своему, отдельно от других. Триггеры переменных периодически оценивают **Выражение** триггера относительно значений его равноправных контекстов, в то время как триггеры событий просто ждут появления **События**.

В определенный момент триггер может решить, что условие триггера выполняется. Этот процесс называется "поднятие тревоги". В этот момент происходит следующее:

- Экземпляр события тревоги 790 создается, сохраняется в базе данных и отправляется подписанному слушателю.
- Оповещения 791 о тревоге отправляются как e-mail и/или SMS сообщения.
- Всплывающее окно с тревогой показывается системным операторам.
- Если установлена опция **Требуется подтверждение**, операторам отправляется запрос для подтвеждения тревоги. Подтверждение тревоги 806 - это техническое подтверждение соответствующего экземпляра события тревоги. Это событие тревоги может быть позже подтверждено другим способом, например, из контекстного меню Журнала Событий.

- Выполняются Корректирующие действия (как автоматические 800, так и интерактивные 801).
- Если системные операторы имеют открытый Журнал Событий, чей фильтр 762 настроен на мониторинг событий Тревоги (например, активны тревоги или подобный фильтр), операторы смогут также видеть событие Тревоги в Журнале Событий.
- Если включена настройка тревоги **Активировать тревогу, если появляются ожидающие (неподтвержденные) экземпляры**, тревога изменит свое состояние 802 с **Нормальная** на **Активная**.
- Если включена настройка **Эскалация на численной основе** и число ожидающих (неподтвержденных) событий тревоги превышает порог **Число ожидающих тревог, необходимых для начала эскалации**, тревога переключится из состояния **Активная** на **Эскалированная**. Уведомления об эскалации также высылаются в этом случае (e-mail и SMS).
- Сам триггер может активироваться. Для триггеров переменной активация происходит в любом случае. Для триггеров событий активация происходит только тогда, когда у триггера есть **Коррелированный** набор событий, и, таким образом, система знает, как деактививровать его при "парном" событии.
- Если триггер активируется, тревога переключится из состояния **Нормальная** на **Активная**.
- Активированный триггер также добавляется в список Активные экземпляры 804 тревоги.
- Активированный триггер также добавляется в список Активные тревоги 67 контекста источника тревоги (в котором это событие/состояние подняло тревогу).

Временная эскалация

Если настройка **Временная эскалация** включена, тревога отслеживает истекшее время с тех пор, когда каждый из этих экземпляров был запущен. Если время превышает значение порога **Время до эскалации**, тревога эскалируется и меняет свое состояние с **Активная** на **Эскалированная**. Уведомления об эскалации (e-mail и SMS) также высылаются в этом случае.

Деактивация тревоги

Каждый активный экземпляр триггера продолжает мониторинг по-своему. Триггер переменной деактивируется после задержки, поскольку результат его **Выражения** (или выражения **Деактиватора**, если оно установлено) меняется на FALSE. Триггер события деактивируется, если было(и) получено(ы) **Коррелированное(ые)** событие(я).

Как только триггер деактивируется, происходит следующее:

- Деактивированный триггер удаляется из списка Активные экземпляры 804 тревоги.
- Деактивированный триггер удаляется из списка Активные тревоги 67 контекста источника тревоги (в котором это событие/состояние подняло тревогу).
- Если другие триггеры неактивны и нет ожидающих экземпляров 804, состояние тревоги меняется обратно с **Активная** на **Нормальная**.

РУЧНАЯ ДЕАКТИВАЦИЯ

Тревоги, активированные Триггерами событий 787 остаются активными до момента появления **Коррелированного** события (если это определено в настройках триггера). Однако в некоторых случаях, это событие может не произойти, напр. из-за его потери вследствие ошибки сети, или оно не было сгенерировано из-за возникающей проблемы устройства.

В таких случаях возможна ручная деактивация триггера. Для этого найдите экземпляр тревоги в таблице Активные экземпляры 804, кликнете по нему правой кнопкой мыши и выберите **Деактивировать тревогу**. Триггер будет деактивирован без ожидания коррелированного события.

Подтверждение тревоги

Если какое-либо Событие тревоги 790 подтверждается 806 и включена настройка тревоги **Активировать тревогу, если есть ожидающие (неподтвержденные) экземпляры**, происходит следующее:

- Если тревога была **Эскалирована**, отсутствуют неподтвержденные экземпляры, ожидающие дольше **Времени до эскалации**, и число ожидающих экземпляров ниже **Числа ожидающих тревог, необходимых для начала эскалации**, тревога деэскалируется и меняет свое состояние с **Эскалированная** на **Активная**.
- Если нет других ожидающих (неподтвержденных) экземпляров тревог, тревога меняет свое состояние с **Активная** на **Нормальная**.

12.9.9 Активные экземпляры

Каждый триггер тревоги (как на основе переменной 782, так и на основе события 787) может быть *активен* в данных случаях:

- Триггер события активен, если основное событие произошло, а коррелированное еще нет.
- Триггер переменной активен, если условие активации является true (вышло время запаздывания), но условие деактивации является false (или время запаздывания деактивации еще не прошло).
- Триггер переменной считается активным, если он находится в состоянии "обнаружено колебание".

Т.к. маски контекста триггера могут использоваться для проверки множества контекстов 41, триггер может быть активен для более одного контекста одновременно.

Экземпляр тревоги называется *активным*, если его триггер остается активным для определенного источника тревоги в контексте.

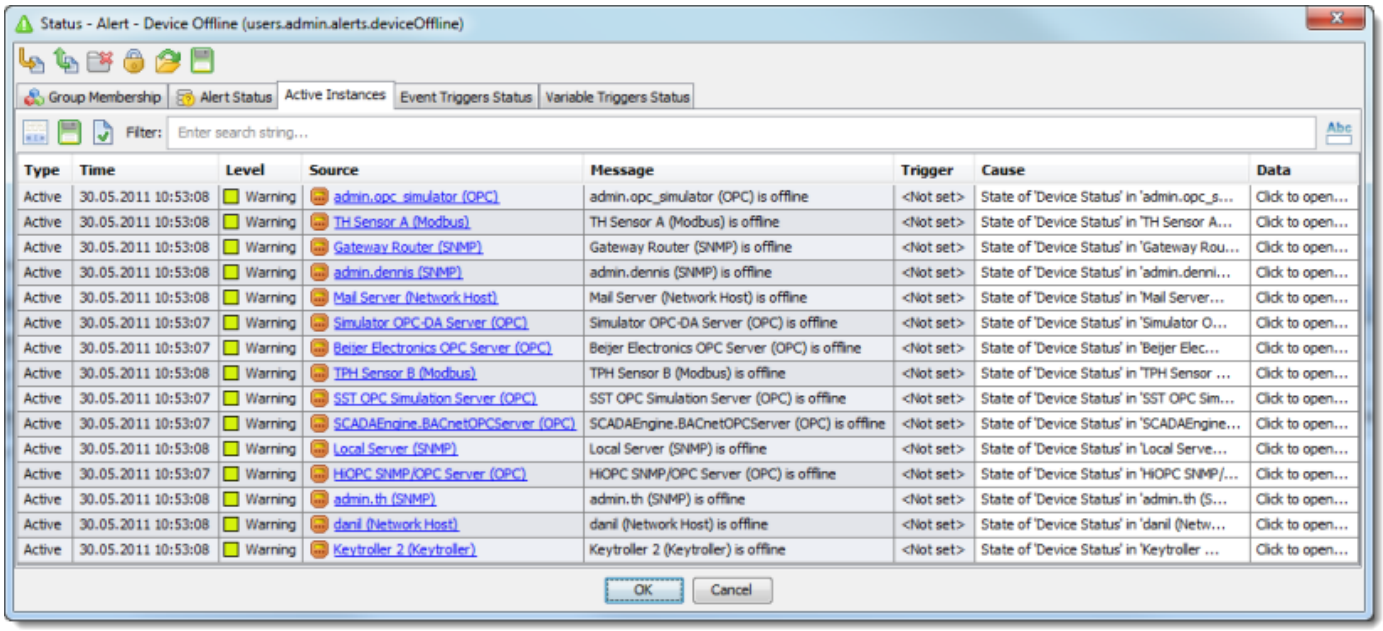
Важно различать **активные** и ожидающие 804 экземпляры тревоги.

Просмотр активных экземпляров

Чтобы увидеть активные экземпляры определенной тревоги, запустите действие Показать статус 111 и проверьте таблицу **Активные экземпляры**.

Чтобы увидеть все активные тревоги в системе, выполните запрос Активные тревоги 833.

Список активных тревог в AtomMind Client выглядит следующим образом:



12.9.10 Ожидающие экземпляры

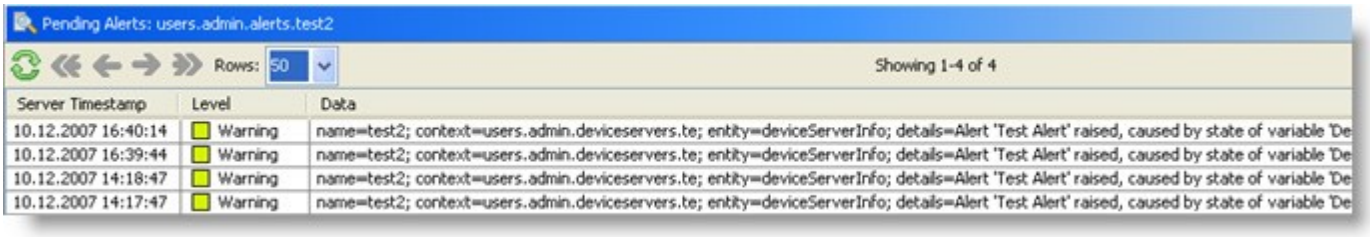
Когда поднимается тревога, создается специальное событие тревоги 790 и сохраняется в базе данных 692 AtomMind Server. События тревоги, которые не были подтверждены 806, называются **Ожидающие экземпляры тревоги**. Для более подробной информации о событиях и подтверждении событий см. тему События 73.

Важно различать активные 804 и **ожидающие** экземпляры тревоги.

Управление ожидающими экземплярами

Контроль ожидающих экземпляров каждой тревоги может осуществляться при помощи действия Контролировать ожидающие экземпляры 1454, заданного в контексте тревоги. Данное действие показывает, когда и почему была поднята тревога, и позволяет удалить или подтвердить ожидающие тревоги. Экземпляры тревоги, которые уже были подтверждены, не отображаются в списке.

Список ожидающих тревог выглядит таким образом (скриншот из AtomMind Client):



Когда появляется одно или более неподтвержденное событие тревоги, тревога остается в состоянии "Активна". Данный статус обозначается специальной иконкой в пользовательском интерфейсе AtomMind Server. Статус "Активна" указывает на то, что необходимо принять меры для решения проблемы, о которой сигнализирует тревога. Когда проблема устранена, системный оператор может запустить действие "Посмотреть ожидающие тревоги" и подтвердить все ожидающие экземпляры тревоги. Тревога в таком случае вновь перейдет в состояние "Включена" (что является нормальным состоянием), обозначаемое иконкой .

Неподтвержденные экземпляры тревоги могут перевести тревогу в состояние "Активна" только в том случае, если включена настройка Разрешить ожидающие тревоги 799.

Просмотр ожидающих экземпляров

Чтобы увидеть ожидающие экземпляры определенной тревоги, запустите действие Посмотреть статус 111 и проверьте таблицу **Активные экземпляры**. Ожидающие экземпляры отмечены специальной меткой в столбце **Тип**.

Чтобы увидеть все ожидающие тревоги в системе, выполните запрос Активные тревоги 833.

Отключение ожидающих тревог

При конфигурации тревоги, одно из свойств 799 называется **Разрешить ожидающие тревоги**. При его отключении (т.е. запрещении ожидающий тревоги для тревоги, которую вы в данный момент настраиваете), тревога не будет отображаться в списке ожидающих тревог, даже если она не была подтверждена. Она также не будет эскалирована 799.

12.9.11 Эскалация тревоги

Ожидающие 804 тревоги могут быть *эскалированы*. Эскалация обычно используется для привлечения внимания к каким-либо критическим условиям. В большинстве случаев эскалирвоанные тревоги, обозначаемый иконкой , требуют от системного оператора принятие быстрых мер.

Для эскалации тревоги необходимо, чтобы был по крайней мере один ожидающий (неподтвержденный) экземпляр, и она должна удовлетворять хотя бы одному из правил эскалации, определяемых свойством Эскалация тревоги 799. Существует два типа эскалации:

- Количественная эскалация
- Временная эскалация

Количественная эскалация означает, что тревога эскалируется, если количество ожидающих экземпляров тревоги 804 превышает лимит, заданный настройкой **Необходимое количество активных тревог для эскалации**.

Временная эскалация происходит при наличии ожидающих экземпляров (событий тревоги), которые не были подтверждены в течение промежутка времени, определяемого настройкой **Время до эскалации.**

Когда тревога эскалируется, e-mail *уведомление об эскалации* отправляется всем адресатам, указанным в настройке **Оповещения о тревоге**. Данное уведомление содержит информацию о времени и причине эскалации:

Причина также показывается во всплывающей подсказке, которая появляется при наведении мыши на тревогу в пользовательском интерфейсе AtomMind Server. В AtomMind Client это выглядит следующим образом:

Чтобы вернуть тревогу в нормальное состояние или состояние ожидания, подтвердите 806 некоторые из ее ожидающих экземпляров, тем самым правила эскалации не будут удовлетворены.

Правила эскалации тревог могут показаться недостаточно гибкими. Тем не менее, существует легкий способ эскалировать тревогу в любом ином случае, отправить уведомления об эскалации и выполнить корректирующие действия при эскалации.

Просто настройте вторую тревогу, чьи триггеры будут:

- Проверять события или переменные того же сервера, как это делает сама тревога, но с использованием других порогов/правил для активирования этой "эскалированной" тревоги
- Либо проверять состояние или статус самой тревоги (например, количество ее активных экземпляров)

Эта вторая тревога-"эскалация" будет иметь собственные правила уведомления и корректирующие действия.

12.9.12 Подтверждение тревоги

Каждая поднятая тревога постоянно хранится в виде события AtomMind Server (см. подробности в Событие тревоги 790). Данные события могут быть подтверждены подобно всем другим событиям AtomMind Server. *Подтверждение тревоги* представляет собой то же самое, что и подтверждение события "Тревога". Подтверждение сохраняется в истории событий, подобно любому другому подтверждению события. Его параметры (автор, время и текст) можно увидеть во время просмотра истории событий (например, используя журнал событий 398 в AtomMind Client).

E-MAIL ПОДТВЕРЖДЕНИЯ

AtomMind Server предоставляет сервис E-mail подтверждения тревоги, позволяя операторам подтверждать события, отвечая на тревожные e-mail уведомления 792. Чтобы начать использовать данный сервис, включите опцию получения e-mail сообщений в настройках e-mail сообщений 1428 общей конфигурации AtomMind Server и сконфигурируйте свойства сервера входящих e-mail сообщений (адрес сервера POP3, логин и пароль). Убедитесь, что включена проверка входящих сообщений 824 по графику (по умолчанию она отключена).

E-mail подтверждения тревоги будут правильно функционировать только в том случае, если ответы на e-mail уведомления будут доставлены в почтовый ящик, контролируемый AtomMind Server. Тревожные e-mail уведомления отправляются с заголовком **Ответить**, включающим адреса, заданные в настройке общей конфигурации **Адрес отправителя e-mail сообщений**. Таким образом, AtomMind Server должен получить все e-mail сообщения, отправленные по данному адресу. Это позволит адресатам тревожных e-mail уведомлений использовать функцию "Ответить" своих почтовых клиентов и ввести текст подтверждения в ответном e-mail сообщении.

Чтобы подтвердить тревогу через e-mail сообщение, ответьте на тревожное e-mail уведомление и введите текст подтверждения **в первой строке** ответа. Подтверждение тревоги будет добавлено, когда ответ будет доставлен и получен AtomMind Server. AtomMind Server периодически проверяет наличие новых сообщений согласно опции общей настройки **Период проверки входящих сообщений**.

Очень важно сохранить тему оригинального тревожного e-mail уведомления при ответе. Почтовый клиент может добавить текст (обычно "Re: "), что является правильным. Однако ID в теме сообщения не должен быть удален или изменен.

12.9.13 Безопасность тревог

Триггеры тревоги производят все проверки, используя таблицу прав доступа 487 владельца тревоги. **Маска контекстов** триггера разрешается только в тех контекстах, которые доступны для владельца тревоги.

Пример: Если у пользователя Джо есть тревога с Маской Контекстов триггера, настроенной на **users.*.devices.***, триггер активируется для всех устройств, доступных с правами доступа 483 Джо. При этом тот же триггер, добавленный к тревоге администратора по умолчанию 479, сработает для всех устройств в системе.

Триггеры тревоги считывают различные данные из контекстов, соответствующих их Маске Контекстов (например, во время оценки Выражения Триггера). Таким образом, владелец тревоги должен иметь достаточный уровень прав доступа 486 во всех этих контекстах.

Другие заметки:

- Триггеры переменной 782 активируются только для переменных, которые читаются при наличии прав доступа владельца тревоги
- Триггеры события 787 активируются только для событий, чьи определения доступны при наличии прав доступа владельца тревоги

Права доступа для корректирующих действий

Все корректирующие действия 793 тревоги также выполняются при наличии прав доступа владельца тревоги.

Пример: Корректирующее действие, принадлежащее тревоге Джо, не сможет остановить или перезапустить AtomMind Server до тех пор, пока у Джо не появится уровень прав доступа 486 **Администратора** в Корневом контексте 1559. Однако тревога администратора по умолчанию 479 всегда может остановить/перезапустить сервер, поскольку у него есть права доступа для осуществления всех возможных действий.

12.9.14 Производительность тревог

Тревоги - это "активные" компоненты сервера, вызывающие постоянную нагрузку процессора. Нагрузка процессора, вызванная тревогой, - это результат сложения нескольких факторов:

- Количество триггеров тревоги.
- Количество контекстов, проверяемых каждым триггером. Например, в большой инсталляции триггер, привязанный к маске 44 контекстов `users.*.devices.*`, приводит к большей нагрузке, нежели триггер, привязанный к `users.john.devices.dev1`.
- Влияние нагрузки процессора 141 выражения триггера тревоги.

Как только один из этих триггеров поднимает тревогу, оказывается дополнительное влияние на нагрузку, вызванное его корректирующими действиями 793.

Тревоги не оказывают постоянной и значительной нагрузки на память. Однако значительное использование памяти происходит при оценке выражения триггера с учетом большого пакета данных или выполнения корректирующего действия.

12.9.15 Примеры тревог

В данном разделе приводятся некоторые существенные конфигурации тревоги, чтобы в дальнейшем вы смогли ссылаться на них при создании своих собственных настроек тревог.

Примеры выражения триггера

Далее приведены несколько реальных примеров выражения триггера тревоги:

Выражение	Примечания
`select({}, 'ifAdminStatus', 'ifIndex', 1) == 2`	Данное выражение будет обращаться к таблице по умолчанию (т.е. к значению переменной 787 в случае триггера переменной), находить запись со значением 1 в поле `ifIndex` и активировать тревогу, если значение поля `ifAdminStatus` равно 2 в данной записи.

`{connectionStatus} == 0 && {genericProperties$offlineAlert}`	Данное выражение используется тревогой "Устройство отключено" 781. Оно активирует тревогу, если поле `connectionStatus` контролируемой переменной (а именно переменной статус 1494 Device) 1494 равно 0 (офлайн), и значение поля `offlineAlert` переменной `genericProperties` (общие свойства) 1497 является TRUE (т.е. активирована тревога статуса Офлайн для данного устройства).
`{temperature} > 27.00 && dayOfWeek(now()) != 0 && dayOfWeek(now()) != 6 && hour(now()) > 9 && hour(now()) < 18`	Данная тревога будет активирована, если температура будет держаться выше 27 градусов с понедельника по пятницу с 9.00 до 18.00.
`contains({message}, "FAILED LOGIN") && {facility} == 4 && {level} == 5`	Данное выражение триггера события активирует тревогу, когда поле `message` данных события содержит часть строки `"Вход в систему не выполнен (FAILED LOGIN)"`, поле `facility` равно 4, а поле `level` равно 5.
`select({.:settingsStatus}, 'duration', 'setting', 'mySqlQuery') > 500`	Это выражение анализирует последнее время выполнения запроса SQL под названием `mySqlQuery`, которое периодически выполняется драйвером устройства базы данных 647. Если время выполнения запроса превышает 500 милисекунд, поднимается тревога. То же выражение может использоваться для анализа продолжительности ввода/вывода (например, времени чтения/записи) **любой переменной устройства**.

Тревога Предупреждение об ударе

Предположим, что наше аппаратное устройство является контроллером погрузчика, который генерирует событие **Удар**, когда погрузчик сталкивается с препятствием. Данное событие содержит поле **Уровень**, которое представляет уровень удара, более высокие уровни сигнализируют о более серьезных ударах.

Тревога **Предупреждение об ударе** имеет только один триггер события 787, который активируется при возникновении события **Удар** в контексте терминала данных 1494, если значение целочисленного поля **Уровень** в данных события выше 50 (т.е. тревога активируется только при серьезных ударах).

ИНФОРМАЦИЯ О ТРЕВОГЕ

Поле	Значение
Имя тревоги	impact_warning
Описание тревоги	Предупреждение об ударе
Тревога включена	TRUE
Сообщение	Сильный удар транспорта

ТРИГГЕРЫ, АКТИВИРУЕМЫЕ СОБЫТИЯМИ

Маска контекста	Событие	Выражение фильтра
users.admin.devices.forklift	impact	{level} > 50

ТРЕВОЖНЫЕ УВЕДОМЛЕНИЯ

Поле	Значение
Уведомлять владельца при наличии подключения к серверу	TRUE
Требуется подтверждение	FALSE

Звук уведомления	<Not set>
Отправит e-mail сообщение владельцу	TRUE
Адресаты e-mail сообщений	NONE
Дополнительные адресаты e-mail сообщений	

Проблема нескольких устройств

Данный комплексный пример показывает, как взаимодействие двух функций системы, тревог и запросов[829], может быть полезным для обнаружения ошибок, касающихся сразу нескольких Device.

Допустим, что десять температурных датчиков подключены к AtomMind Server и принадлежат пользователю **john.** Контекст терминала данных[1494] каждого датчика имеет переменную "Температура" (имя переменной: **temperature**), значение которой обладает целочисленным полем, **celsius**, показывающим температуру в градусах Цельсия. Предположим, что температура отдельно взятого датчика не критична, но если три или более датчика сообщают о температуре выше 100 градусов, что может привести к проблеме в дальнейшем, необходима активация тревоги, чтобы позволить операторам выяснить и устранить проблему.

Мы будем использовать запрос, который поможет найти все термометры, сообщающие о температуре выше 100 градусов:

```
SELECT * FROM users.john.devices.*:temperature WHERE temperature$celsius > 100
```

Данный запрос вернет таблицу (таблицу данных[49]), содержащую одну запись для каждого датчика, сообщающего о высокой температуре.

Выполнить данный запрос можно путем вызова функции[70] `executeQuery` из корневого[1559] контекста и ввода текста запроса в качестве параметра входа. Смысл в том, чтобы установить тревогу с триггером, который будет периодически выполнять данный запрос и проверять количество записей выхода. Если количество больше трех, триггер активируется.

Триггеры переменной могут и не использоваться напрямую для проверки выхода функции, но в данном случае нам необходимо проверить выход функции - `executeQuery` является функцией. Как было сказано ранее, тревоги могут быть активированы двумя способами - когда возникает событие (так называемые "триггеры события") или согласно значению какой-либо переменной ("триггер переменной"). В нашем примере мы будем использовать триггер переменной, но немного его доработаем. Триггер переменной всегда имеет **контекст**, **переменную** и **выражение**. В данном примере выражения *могут* вызывать функции.

Итак, мы собираемся создать тревогу с не имеющими значения контекстом, переменной (которая все равно соответствует контексту) и очень важным значением выражения. Данное выражение не будет даже ссылаться на контекст и переменную тревоги. В данном случае для нас важно само выражение, потому что именно оно приведет к выполнению запроса. Все остальные части (контекст и переменная) являются просто шаблонами. В качестве их значений мы возьмем "" (корневой) контекст и доступную только для чтения переменную **version**, содержащую версию сервера.

Тревога активируется, когда значение оценивается как **true**.

Чтобы получить доступ к записям, содержащимся в выходе функции **executeQuery**, мы будем использовать свойство Записи (более подробное описание о процессе разрешения ссылки см. в разделе Ссылки)[117]. Итак, итоговая ссылка, которая вернет количество датчиков, зарегестрировавших высокую температуру, выглядит таким образом:

```
{:executeQuery("SELECT * FROM users.john.devices.*:temperature WHERE
temperature$celsius > 100")#records}
```

Принцип действия:

1. Триггер периодически опрашивает корневой контекст на значение переменной **version.**
2. Данное значение отвергается, т.к. оно не используется в выражении триггера.
3. Происходит оценка выражения. В процессе оценки преобразователь ссылок по умолчанию разрешает вышеуказанную ссылку - она вызывает функцию "Выполнить запрос" и возвращает количество рядов в результате. Данное количество сравнивается с численной константой "3". Итоговое выражение возвращает TRUE, если оно более или равно 3, в противном случае, значение будет FALSE.
4. Если выражение возвращает TRUE, триггер активирует тревогу и отправляются все уведомления[791].

ИНФОРМАЦИЯ О ТРЕВОГЕ

Поле	Значение
Имя тревоги	temperature_warning
Описание тревоги	Предупреждение о температуре
Тревога включена	TRUE
Сообщение	Три и более датчиков показывают высокую температуру

ТРИГГЕРЫ ПЕРЕМЕННОЙ

Маска контекста	Переменная	Выражение	Monitor State Changes	Check Period (seconds)	Level
"" *(empty string - Root context)*	version	{:executeQuery("SELECT * FROM users.john.deviceservers.sensors.devices.*:temperature WHERE temperature$celsius > 100")#records} >= 3	FALSE	10	Error

ТРЕВОЖНЫЕ УВЕДОМЛЕНИЯ

Поле	Значение
Уведомлять владельца при наличии подключения к серверу	TRUE
Требуется подтверждение	FALSE
Звук уведомления	<Not set>
Отправить e-mail сообщение владельцу	TRUE
Адресаты e-mail сообщений	NONE
Дополнительные адресаты e-mail сообщений	

Дополнительные примеры можно найти в разделе Триггеры 780.

12.9.16 Тревоги в распределенной архитектуре

Этот раздел описывает особенности поведения тревог в распределенной инсталляции 1332 AtomMind.

Чтобы всплывающее уведомление о тревоге, пришедшее с сервера-поставщика, появилось у клиентов, подключенных к серверу-потребителю, убедитесь, что путь контекста подключенного контекста тревоги на сервере-потребителе соответствует маске контекстов `users.*.alerts.*`, т.е. удаленная тревога появляется среди локальных тревог.

12.10 Модели

Модели используются для имитирования различных бизнес-объектов и процессов. Модели могут быть как простыми, определяющими несколько свойств и бизнес-правил, так и сложными, т.е. представляющими целый промышленный объект.

Модели могут функционировать самостоятельно, но также могут прикрепляться к объектам низшего уровня, таким как устройства. Во втором случае создается несколько экземпляров модели, каждая из которых использует объект, к которому она прикреплена в качестве первичного источника данных.

Каждая модель включает в себя:

- Определения переменных (свойства), представляющие значения модели
- Определения функций (операции), указывающие модели выполнить некую обработку или расчет
- Определения событий, которые может воспроизводить модель
- Определения привязок, связывающих вместе свойства, операции и события модели, позволяющие ей реагировать на события и состояния других объектов
- Наборы бизнес-правил для принятия решения в соответствии с машиночитаемой базой знаний

Пример простой реальной модели

Простые системы мониторинга и контроля допускают прямую визуализацию значений, собранных с устройств. Однако устройства различных производителей используют разные способы предоставления значений с одинаковым физическим смыслом.

Например, продукция по менеджменту сети используется для отслеживания нагрузки процессора узлов сети. Эта простая метрика представлена в виде множества разных форм:

- Компьютеры на основе Windows выражают "точечные" значения нагрузки процессора через SNMP 637.
- Те же Windows-машины обеспечивают чтение нагрузки процессора на основе WMI 661, если SNMP недоступен.
- Маршрутизаторы Cisco предоставляют предварительно рассчитанные данные по использованию процессора в виде среднего значения за 5 минут или 1 час.
- Серверы HP-UX имеют значения по типу счетчика, показывающие, сколько секунд был занят процессор с момента запуска. Эти счетчики требуют сложных вычислений для подсчета текущей нагрузки.

И все же панель инструментов 912 Устройства Сети должна иметь диаграмму "Нагрузка процессора", имеющую одинаковый вид для всех типов устройств. Тревога "Высокая нагруза процессора" должна вести себя подобным же образом. Помимо этого, мы бы хотели достроить отчёт "Обзор нагрузки процессора", показывающий текущее использование всех устройств нашей сети.

Лучший способ соблюсти обозначенные требования - иметь числовую метрику "Нагрузка процессора" в каждом устройстве, содержащем данные по процессору. У этой метрики должен быть общий формат, но ее расчет и процедура обновления будет отличаться в зависимости от типа устройств.

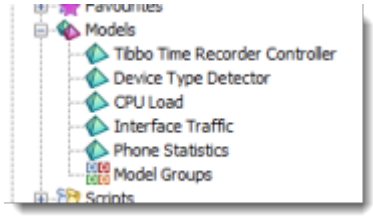
В данном случае поможет модель. Модель "Нагрузка процесора" прикрепляется к каждому устройству сети и включает набор бизнес-правил для расчета нагрузки процессора. Расчет правил вызывается периодической привязкой. Результат расчета набора бизнес-правил записывается в переменную нагрузки процессора, которая декларируется той же моделью.

Модели немного похожи на виджеты 943. Модель можно интерпретировать как виджет, существующий внутри сервера и не имещий пользовательского интерфейса. Однако у моделей и виджетов есть механизмы привязки 735, которые приводят их в действие, активируя определенное поведение при появлении событий системы и изменении состояния.

Каждый пользователь 478 имеет собственный набор моделей.

Модели администрирования

Два контекста используются для администрирования моделей: один -это общий контекст моделей 1537, выступающий в качестве контейнера. Другой - это контекст модели 1539, содержащий информацию по одной модели.



12.10.1 Типы моделей

Существует три типа моделей:

- **Абсолютные** Модели
- **Относительные** Модели

- **Экземплярные** Модели

Абсолютные модели

Абсолютная модель не имеет изменяемого источника данных. Она использует абсолютные ссылки, чтобы получить данные из разных частей системы. Все переменные, функции и события, описанные абсолютной моделью, добавляются к собственному контексту 1539.

Например, модель, имитирующая технологический процесс, должна извлекать данные из множества специфичных контроллеров и производить над ними сложные математические вычисления. Результат вычислений должен отображаться в виде свойств модели. Такая модель должна быть абсолютной.

Привязки 821 и выражения правил 813 абсолютной модели не могут включать ссылки 117 относительной.

Относительные модели

Относительные модели обрабатывают данные из определенного контекста 41 источника и его дочерних контекстов. Отдельный экземпляр относительной модели создается для каждого контекста, к которому он прикреплен.

Например, модель, рассчитывающая нагрузку процессора устройства сети относительно его типа и типа связи, должна прикрепляться к каждому устройству сети, применять набор правил к его свойствам и добавлять "общее" свойство нагрузки процессора к каждому устройству. Такая модель должна быть относительной.

Привязки 821 и выражения правил 813 относительной модели могут включать ссылки 117 относительной.

Экземплярные модели

Экземплярные модели имеют *экземпляры,* создающиеся по запросу. Как только Вы определяете экземплярную модель, AtomMind Server создает несколько контейнеров для хранения экземпляров модели и прикрепляет эти контейнеры к различным частям контекстуального дерева 41 AtomMind Servera.

Таким образом, экземпляры экземплярных моделей очень похожи на другие объекты AtomMind Servera, такие как Тревоги 779 или Отчеты 928. Однако эти экземпляры полностью пользовательские. Их свойства и поведение описывается моделью.

Например, Вы управляете нефтяными вышками. Каждая вышка - это сложный объект, включающий в себя множество устойств, обеспечивающих различную телеметрию и серверные правила для хранения и обработки статистики, подготовки пригодных для печати отчетов и т.д.

В этом случае вам нужно создать экземплярную модель Нефтяная Вышка и прикрепить контейнеры экземпляров этой модели к учетным записям пользователя 478 AtomMind Servera (точно таким же образом, как другие объекты присваиваются пользователям).

Каждый экземпляр модели будет ссылаться на определенные устройства, предоставляя необходимые телеметрические данные и все необходимые переменные/функции/события, привязки и бизнес-правила, необходимые для обработки и визуализации собранных данных.

Для более подробной информации см. Использование экземплярных моделей 814.

12.10.2 Жизненный цикл модели

Как только создается или повторно инициализируется новая модель после запуска сервера, она проходит определенный цикл:

- Во-первых, находятся "пригодные" контексты 41 источника данных в соответствии с типом 811 модели:
  - Если модель абсолютная, ее источник данных - это сам контекст модели.
  - Если модель относительная, сервер рассчитывает ее выражение пригодности для каждого контекста в системе, чтобы определить, "совместим" ли он с моделью.
  - Если модель инстанцируемая, сервер рассчитывает ее выражение пригодности для каждого контекста в системе и добавляет дочерний контейнер для моделей этого типа в каждый контекст, для которого она пригодна. Эти контейнеры содержат экземпляры модели.
- Во-вторых, модель добавляет определения ее переменных, функций и событий к каждому пригодному контексту:
  - Контекст самой абсолютной модели.

- Все контексты, для которых пригодна относительная модель.
- Все экземпляры инстанцируемой модели, которые уже определены в контейнерах, найденных на первом этапе.

- В-третьих, модель также добавляет функции, отсылая наборы правил[813] модели ко всем пригодным контекстам.
- В-четвертых, модель инициализирует активаторы ее привязок[814], чтобы события и изменения в контекстах источника вызывали обработку привязок модели.

Нормальное функционирование модели

Как только заканчивается первичный запуск, модель переключается на ждущий режим. Ее последующие действия:

- Периодические привязки модели обрабатываются согласно графику, заставляя модель обрабатывать соответствующие привязки
- Некоторые события по обновлению значения переменной контекста сервера также вызывают активацию определенных привязок модели
- Другие компоненты системы читают/записывают свойства модели и выполняют ее функцию

12.10.3 Правила модели

Каждая модель может включать множество наборов *бизнес-правил,* охватывающих *машиночитаемую базу знаний,* помогая системе принимать решения и выполнять сложные расчеты во время автономного функционирования.

См. Правила[743] для более подробной информации

Использование наборов правил

Чтобы обработать набор правил из собственной привязки[821] модели или любого внешнего компонента системы, вызовите *функцию контекста набора правил*. Для получения более подробной информации см. ссылки на наборы правил[813].

Каждый набор правил выполняется отдельно, таким образом, можно вызвать один и тот же набор правил с различными параметрами одновременно. В этом случае, они будут работать параллельно.

12.10.3.1 Ссылки на наборы правил

Каждый набор правил, определенный моделью, представлен как отдельная контекстная функция[70]. Эта функция принимает Таблицу Данных[49] любого формата как входное значение и представляет ее в виде другой Таблицы Данных, чтобы вернуть в качестве функционального выхода. Имя контекстной функции набора правил соответствует имени самого набора правил.

Функция набора правил определяется в контексте модели (или каждым контекстом, к которому прикреплена относительная модель).

Поскольку функции контекста всегда возвращают Таблицы Данных, любая склярная величина, возвращенная "итоговым" выражением правила, будет представлена в виде Таблицы Данных в одну ячейку. Например, если выражение набора правил, нацеленных на **Итоговый Результат Набора Правил**, вернуло целое число, контекстная функция набора правил вернется в виде Таблицы Данных в одну ячейку со столбиком Целое и одной строкой, содержащей это целое число.

Если на эту функцию ссылаются из выражения, необходимо использовать функцию языка выражений[124] `cell()`, чтобы извлечь результат целого числа из таблицы функционального выхода.

12.10.4 Привязки модели

Механизм модели строится из контекстных привязок сервера 736. Эти привязки делают модель по-настоящему динамической, заставляя ее реагировать на контекстные события 73 сервера и изменения контекстных переменных 61 сервера.

Привязки определяют:

- Отношения между данными модели (т.е. собственные переменные 817, функции 818 и события 819 модели).
- Отношения между контекстными данными сервера (т.е. переменными 61, функциями 70 и событиями 73 контекста сервера).
- Отношения между данными сервера и данными модели.

Каждая привязка *оценивается* на разных стадиях функционирования модели.

Для получения более подробной информации см. контекстные привязки сервера 736.

12.10.5 Использование экземплярных моделей

Экземплярная модель позволяет определить новые бизнес-объекты с нуля и позволить системным операторам или администраторам более низкого уровня создавать экземпляры этих бизнес-объектов по запросу и управлять ими.

Управление контейнерами экземпляров модели

Подобно всем других объектам AtomMind Server (таким как запросы 829), экземпляры экземплярных моделей хранятся в контекстах 41 контейнера. Однако контейнеры запросов и другие стандартные объекты в большинстве случаев находятся в двух частях дерева контекстов сервера:

- В контексте пользователей 478, если пользователи владеют этими объектами (например: тревоги)
- В корневом контексте 1559, если это глобальные объекты (например: временные зоны)

В отличие от "классических" контейнеров, контейнеры экземпляров модели могут прикрепляться к **любой** части дерева контекстов сервера. Это контролируется в**ыражением пригодности** и **правилами обновления пригодности** экземплярной модели (см. пригодность ресурса 749). **Выражение пригодности** оценивается для каждого контекста в дереве контекстов сервера, и если его результат TRUE, контейнер экземпляра модели прикрепляется к этому контексту.

Так, чтобы эмулировать поведение "классических" объектов, выражение пригодности экземплярной модели конфигурируется следующим образом:

- `{.:#type} == "root"` для прикрепления одного контейнера экземпляра к корневому контексту (глобальные экземпляры)
- `{.:#type} == "user"` для прикрепления контейнера экземпляра к каждому контексту пользователя 1608 (экземпляры на каждого пользователя)

Контейнер экземпляра модели представлен иконкой  , в то время как отдельные экземпляры используют иконку  .

управление экземплярами моделей

Управлять экземплярами экземплярных моделей можно так же, как любыми другими ресурсами системы, например, тревогами 779 или отчетами 928.

Контейнеры экземпляров моделей поддерживают все типовые действий с контейнерами ресурсов - Создать на основе шаблона 105, Копировать в дочерние контексты 111, Import 108, Экспорт 108, Редактировать права доступа 106, Просмотр событий 109, Поиск/фильтрация 110, а также различные Групповые действия 101.

Экземпляры моделей поддерживают все типовые действий с ресурсами, включая Удалить 106, Создать копию 109, Реплицировать 110, Редактировать права доступа 106, Просмотр событий 109 и Показать статус 111.

Контексты экземпляров экземплярных моделей предлагают действие **Переместить в контейнер,** которое позволяет перемещать экземпляр в контейнер другого экземпляра данной модели.

Конфигурирование экземпляров моделей и их контейнеров

Вот некоторые свойства 815 экземплярной модели, контролирующие основные характеристики контекстов контейнеров экземпляров модели и контексты экземпляров модели:

- Тип контейнера
- Описание типа контейнера
- Имя контейнера
- Тип объекта
- Описание типа объекта
- Выражение именования объекта

Первые три контролируют контексты контейнеров экземпляров модели:

- **Тип контейнера** - это идентификатор, который может позже использоваться для нахождения или различения контейнеров экземпляров модели в пределах дерева контекстов сервера. Задайте имя объекта в горбатом регистре (camel case) во множественном числе, т.е."oilDerricks" ("нефтяные вышки"). Для получения более подробной информации см. типы контекстов 43.
- **Описание типа контейнера** - это удобочитаемая для человека версия вышеобозначенного, т.е. "Oil Derricks".
- **Имя контейнера -** это контекстное имя, которое будет использоваться для контекстов контейнеров модели. Должно удовлетворять соглашениям по наименованию 42 контекста. Имя требуется для ссылки на контейнеры из других частей системы. Установите имя "oilDerricks" (то же, что имя Типа Контейнера), и если вы прикрепили ваши контейнеры модели к контекстам пользователя (см. выше), полный путь контейнера будет выглядеть как `users.john.oilDerricks`.

Следующие три свойства контролируют контексты экземпляров модели:

- **Тип объекта -** это, подобно типу контейнера, идентификатор, который может позже использоваться для нахождения или различения экземпляров модели в пределах дерева контекстов сервера. Задайте имя вашего объекта в горбатом регистре (camel case) во множественном числе, т.е. "oilDerrick". Для получения более подробной информации см. типы контекстов 43.
- **Описание типа объекта** - это удобочитаемая для человека версия типа объекта, т.е. "Oil Derrick".
- **Выражение именования объекта** - это выражение используется для вычисления удобочитаемых для человека описаний экземпляров модели.

12.10.6 Конфигурирование модели

Этот раздел описывает свойства конфигурации модели.

Все описанные здесь свойства доступны при помощи действия Конфигурировать 105 в Контексте модели 1539.

12.10.6.1 Свойства модели

Определяет основные опции модели.

Описание поля	Имя поля
Имя. Имя контекста модели. Оно должно удовлетворять соглашениям по наименованию 42 контекста. Имя необходимо для ссылки на эту модель из других частей системы.	name
Описание. Текстуальное описание 43 контекста модели.	description
Тип. Тип 811 модели, **Относительная**, **Абсолютная** или **Экземплярная**.	type
Выражение пригодности. Действует по-разному в зависимости от типа модели 811, относительной или экземплярной: • В случае относительной модели определяет, к какому(им) контексту(ам) должна прикрепляться модель. См. Пригодность ресурса 749 для получения более подробной информации. • В случае экземплярной модели определяет, к какому(им) контексту(ам) должны прикрепляться контейнеры экземпляров модели.	validityExpression

Правила обновления пригодности. Список масок контекста и имен событий. Если событие, определенное полем **Событие** этой таблицы, происходит в любом контексте, который соответствует маске, определенной полем **Маска** в той же записи, **Выражение пригодности** для этого контекста пересчитывается. Это позволяет сделать модель пригодной/непригодной для определенного контекста, если происходят какие-либо изменения.	validityListeners
Контекст по умолчанию. Опция облегчает редактирование ссылок 117 относительных моделей 811. См. Контекст модели по умолчанию 822 для получения более подробной информации.	defaultContext
Тип контейнера. Применяется только для экземплярных 811 моделей. Определяет тип контекста 43 контейнеров модели. Строка типа может содержать только английские буквы, числа и нижние подчеркивания.	containerType
Описание типа контейнера. Применяется только для экземплярных 811 моделей. Определяет удобочитаемое для человека описание типа контекста контейнера модели.	containerTypeDescription
Имя контейнера. Применяется только для экземплярных 811 моделей. Определяет контекстное имя 42 контейнеров модели. Оно должно удовлетворять соглашениям по наименованию 42 контекста.	containerName
Тип объекта. Применяется только для экземплярных 811 моделей. Определяет тип контекста 43 экземпляров модели. Строка типа может содержать только английские буквы, числа и нижние подчеркивания.	objectType
Описание типа объекта. Применяется только для экземплярных 811 моделей. Определяет удобочитаемое для человека описание типа контекста экземпляра модели.	objectTypeDescription
Выражение именования объекта. Применяется только для экземплярных 811 моделей. Определяет выражение, используемое для вычисления удобочитаемых для человека описаний экземпляров модели	objectNamingExpression
Разрешенный. Если этот флаг снят, модель деактивируется и не обрабатывает какие-либо активные привязки. При этом все определения переменных/функций/событий, добавленные этой моделью, остаются доступны, если модель деактивирована. Экземпляры деактивированной инстанцируемой модели также не скрываются/убираются из дерева контекстов сервера.	enabled
Порог глубины стека вызовов наборов правил. Определяет максимальное количество вложенных вызовов наборов правил. Если количество рекурсивных вызовов наборов правил превысит данное пороговое значение, выполнение выдаст ошибку о переполнении стека.	ruleSetCallStackDepthThreshold
Стандартные одновременно обрабатываемые привязки. Определяет размер ядра пула потоков модели, т.е. стандартное количество привязок, которые обрабатываются одновременно.	normalConcurrentBindings
Максимальные одновременно обрабатываемые привязки. Определяет максимальный размер пула потоков модели, т.е. количество одновременно обрабатываемых привязок, разрешенных в случае переполнения очереди привязок.	maximumConcurrentBindings
Максимальная длина необработанной очереди привязок. Определяет, сколько необработанных операций привязки можно поставить в очередь, прежде чем размер пула потоков модели превысит ее размер ядра относительно максимального размера.	maximumBindingQueueLength
Журналирование привязок. Если включена эта опция, каждое выполнение привязки будет сопровождаться особым событием Выполнение привязки 1539.	logBindingsExecution

Доступ к этим свойствам открывается через переменную childInfo 1540.

Если относительная модель соединена с большим количеством устройств, или инстанцируемая модель имеет несколько экземпляров, вы должны увеличить значения параллельно работающих привязок, так как все данные перерасчитываются корневой моделью. Для лучшего понимания логики обработки привязок, см. описание параллельной работы привязок 742.

12.10.6.2 Переменные модели

Таблица переменных модели содержит свойства переменных, описанные моделью.

Описание поля	Имя поля
Имя. Имя переменной.	name
Описание. Описание переменной.	description
Формат. Формат переменной. См. раздел формат 50 для получения более подробной информации.	format
Перезаписываемый. Флажок, указывающий на то, что переменная может перезаписываться.	writable
Помощь. Подробное описание переменной.	help
Группа. Группа переменной или NULL, если переменная не принадлежит ни к одной группе.	group
Права доступа на чтение. Уровень 486 прав доступа в контексте модели (либо контекст, к которому прикрепляется относительная модель, либо контекст экземпляра экземплярной модели), необходимый для чтения переменной.	readPermissions
Права доступа на запись. Уровень 486 прав доступа в контексте модели (либо контекст, к которому прикрепляется относительная модель, либо контекст экземпляра экземплярной модели), необходимый для записи переменной.	writePermissions
Режим хранения. Определяет, постоянно ли значение переменной (тип хранения **База данных**), либо оно переходное (тип хранения **Память**).	storageMode
Время хранения истории. Определяет, как долго хранятся ранее зафиксированные значения этой переменной в базе данных сервера.	updateHistoryStorageTime
Режим записи истории. Управляет политикой сохранения истории и запуска обновлений. Если установлен режим – "Только измения", то выставление того же значения для переменной будет проигнорировано. Если установлен режим "Все значения", то любая операция для переменной будет сохранена, и запустится обновление.	historyRate
Время кэширования. Определяет минимальный период последовательного удаленного чтения значения переменной. Если удаленные компоненты пытаются прочитать переменную дважды во время данного периода, вторая попытка чтения вернет значение из кэша. Неопределенное значение отключает кэширование.	cacheTime
Кэширование значений в памяти сервера. Управляет кэшированием значений переменных на стороне сервера. 'Хранить всегда' - в кэш всегда будет помещаться последнее записанное значение переменной. 'Хранить, если есть место в памяти' -записанное значение переменной будет помещаться в кэш только при наличии свободной памяти.	serverCachingMode
Добавить предыдущее значение в событие обновления переменной. Флаг показывает, что событие обновления 84 переменной будет содержать предыдущее значение переменной.	addPreviousValueToVariableUpdateEvent

Доступ к этим свойствам открывается через переменную modelVariables 154.

Больше информации о свойствах определения переменной см. в Переменные 61.

12.10.6.3 Функции модели

Таблица функций модели содержит свойства функций, объявленных моделью.

Описание поля	Имя поля
Имя. Имя функции.	name
Описание. Описание функции.	description
Формат ввода. Формат ввода функции. См. раздел формат 50 для получения более подробной информации.	inputformat
Формат вывода. Формат вывода функции. См. раздел формат 50 для получения более подробной информации.	outputformat
Помощь. Подробное описание переменной.	help
Группа. Группа 61 переменных или НОЛЬ, если переменная не принадлежит ни одной группе.	group
Права доступа. Уровень 486 прав доступа в контексте модели (либо контекст, к которому прикреплена относительная модель, либо контекст экземпляра экземплярной модели), необходимый для выполнения функции.	permissions
Тип. Определяет тип реализации функции. Поддерживаются следующие типы: • **Код Java**. Реализация функции представлена Java классом, реализующим специальный интерфейс, который содержит метод "выполнить функцию". • **Выражение**. Реализация функции оценивает Выражение AtomMind 112, принимает ввод функции за таблицу по умолчанию 120. Выражение должно оценивать таблицу, которая представляет собой вывод функции. • **Запрос**. Реализация функции выполняет запрос AtomMind 829, используя значения яйчеек таблицы ввода в качестве значений параметров запроса. Таблица результатов запроса представляет собой вывод функции.	type
Реализация. Доступно, если выбран **Тип** Код Java. Исходный код класса Java, который реализует тело функции, т.е. делает то, что должна делать функция. Для получения более подробной информации см. определение и реализация функций 1388.	implementation
Выражение. Доступно если выбран **Тип** Выражение. Выражение AtomMind 112 для оценки во время выполнения функции. Среда вычисления 114 выражения функции: Контекст по умолчанию 119: • Для абсолютной модели: сам контекст модели • Для относительной модели: контекст, к которому прикреплена относительная модель • Для экземплярной модели: контекст экземпляра модели Таблица данных по умолчанию 120: Таблица данных ввода функции. Ряд по умолчанию 119: 0 Переменные среды 123: Только стандартные 123 переменные.	expression
Запрос. Доступно если выбран **Тип** Запрос. Запрос AtomMind 829 для выполнения в процессе выполнения функции.	query

Текст запроса может ссылаться на параметры функции, используя символ ? , например: `SELECT * FROM users.*.devices.*:temperature AS tpr WHERE tpr.temperature$celsius > ?` Номер ссылки параметров ? в тексте запроса должен соответствовать номеру ячейки в таблице ввода функции. Первая ссылка параметров ? будет заполнена значением из первой колонки первого ряда таблицы ввода, второе значение параметра будет взято из второй колонки, и т.д.	
Параллелизм. Делает возможным выполнение функции при каждом вызове в отдельном окружении. Так, можно вызвать одну и ту же функцию с различными входными параметрами одновременно. В этом случае они будут работать независимо и параллельно.	concurrent
Плагин. Идентификационный номер плагина 207 сервера, который определяет модель. Необходим, если реализация функции должна иметь доступ к внутренним классам Java плагина.	plugin

Доступ к этим свойствам открывается через переменную modelFunctions 1541.

Больше о полях определения функции см. в Функции 70.

12.10.6.4 События модели

Таблица событий модели содержит свойства типов событий, описанные моделью.

Описание поля	Имя поля
Имя. Имя события.	name
Описание. Описание события.	description
Формат. Формат данных события. См. раздел формат 50 для получения более подробной информации.	format
Помощь. Подробное описание события.	help
Уровень. Уровень события по умолчанию.	level
Группа. Группа события или НОЛЬ, если событие не принадлежит ни к одной группе.	group
Права доступа. Уровень 486 прав доступа в контексте модели (либо контекст, к которому прикреплена относительная модель, либо контекст экземпляра экземплярной модели), необходимый для подписки на тип события.	permissions
Права доступа для генерации. Уровень 486 прав доступа в контексте модели (либо контекст, к которому прикреплена относительная модель, либо контекст экземпляра экземплярной модели), необходимый для генерирования экземпляра этого типа события.	firePermissions

Доступ к этим свойствам можно получить через переменную modelEvents 1542.

Больше о полях определения события см. в События 73.

12.10.6.5 Наборы правил

Таблица наборов правил определяет наборы бизнес-правил 813, используемые моделью.

Описание поля	Имя поля

Имя. Имя набора правил. Используется для ссылки на этот набор из других компонентов системы. Поскольку каждый набор представлен функцией 70, его имя может содержать только буквы, числа и нижние подчеркивания.	name
Описание. Удобочитаемое для человека описание набора правил.	description
Тип. Тип набора правил: Последовательная обработка или Отслеживание зависимости.	type
Правила. Список правил. См. ниже описание полей правил.	rules

Доступ к этим свойствам можно получить через переменную ruleSets 1542.

Поля Правил

Каждое правило в наборе правил имеет следующие свойства:

Описание поля	Имя поля
Цель. Может быть как **Итоговым результатом набора правил** (т.е. результатом всего набора правил), так и именем переменной среды набора правил, которое определяется или повторно определяется правилом.	target
Выражение. Выражение 112, которое рассчитывает результат правила. Среда Вычисления 114 Выражения Правила: Контекст по умолчанию 119 — Контекст абсолютной модели или контекст, к которому прикреплена относительная модель. Таблица данных по умолчанию 120 — Данные, переданные функции набора правил 813. Строка по умолчанию 119 — 0 Переменные среды 123 — Любая переменная среды, которая была определена ранее выполненными правилами того же набора правил.	expression
Условие. Опциональное выражение 112, которое в результате дает логическое значение. Оценка правила пропускается, если выражение условия в результате дает false. Среда Вычисления 114 Условия Правила: Контекст по умолчанию 119 — Контекст абсолютной модели или контекст, к которому прикреплена относительная модель. Таблица данных по умолчанию 120 — Данные, переданные функции набора правил 813. Строка по умолчанию 119 — 0 Переменные среды 123 — Любая переменная среды, которая определена ранее выполненными правилами того же набора правил.	condition
Комментарии. Любые заметки о назначении правила.	comment

12.10.6.6 Привязки

Таблица привязок определяет привязки модели.

Описание поля	Имя поля
Цель. Цель привязки - это особый тип ссылки[117], который определяет, куда записывается результат оценки выражения привязки, когда обрабатывается привязка.	target
Выражение. Это выражение[112], которое оценивается каждый раз, когда обрабатывается привязка. Результат оценки сохраняется в **цели привязки**.	expression
Активатор. Это ссылка, указывающая на некоторое событие или свойство модели, которое вызывает обработку привязки. Параметр **Активатор** доступен только тогда, когда активен параметр **При событии**.	activator
Условие. Условие - это выражение[112], которое первым делом оценивается после активации привязки. Если это выражение дает результат false, выполнение привязки пропускается.	condition
При запуске. Когда активен этот параметр, привязка обрабатывается каждый раз, когда создается экземпляр модели.	onstartup
При событии. Если активен этот параметр и обозначен **Активатор**, привязка обрабатывается, если возникает изменение свойства, на которое ссылается активатор. Если **Активатор** ссылается на событие, привязка будет обрабатываться, когда срабатывает событие. Если активен параметр **При событии,** а **Активатор** не определен, привязка будет обрабатываться автоматически: **Выражение** привязки включает ссылки, указывающие на одну или более переменных. Изменение любой из этих переменных вызывает выполнение привязки.	onevent
Периодически. Если активен этот параметр, привязка будет обрабатываться периодически.	periodically
Период. Интервал между сессиями обработки привязок. Этот параметр можно изменить только тогда, когда активен параметр **Периодически**.	period

Доступ к этим свойствам открывается через переменную bindings[1543].

12.10.6.7 Статистические каналы

Переменные модели поддерживают статистические каналы[718], которые помогают сохранять на долгий срок историю полей переменных, представляющих числовые значения.

Таблица Статистические каналы самой модели должна использоваться только для абсолютных моделей!

Если вы хотите добавить статистические каналы к контекстам, к которым прилагается относительная модель, либо к контекстам экземплярной модели, используйте общую таблицу Statistics configuration.

Просмотр статистики

Чтобы посмотреть краткую статистику переменной модели, запустите действие Посмотреть Статус[111] () и перейдите во вкладку Статистика.

Просмотр подробной статистики

Чтобы посмотреть подробную статистику переменной модели, используйте глобальное действие Посмотреть Статистику[1562].

12.10.6.8 Грануляция

Переменные моделей поддерживают грануляцию 731, которая помогает объединять исторический значения переменных.

Более подробно о настройке гранулятеров см. раздел Грануляция 731.

12.10.7 Контекст модели по умолчанию

Относительная модель может включать относительные ссылки 117. Проще говоря, **Контекст по умолчанию -** это свойство, используемое для разрешения ссылок модели во время их редактирования, когда модель не прикреплена ни к какому контексту. Используется в трех случаях:

- Во время редактирования привязок модели 821
- Во время редактирования правил модели 813
- Во время редактирования выражения пригодности 749 модели

Когда относительная модель прикреплена к контексту, пути относительных контекстов в ссылках разрешаются относительно пути контекста, к которому была прикреплена модель. Свойство контекста по умолчанию в этом случае не используется.

12.10.8 Безопасность моделей

Все привязки 814 and правила 813 модели обрабатываются при наличии прав доступа 477 владельца модели. Это по сути означает, что:

- Выражения привязок рассчитываются при наличии прав доступа владельца модели
- Цель привязок также записывается при наличии прав доступа владельца модели
- Выражения правил тоже оцениваются при наличии прав доступа владельца модели

Таким образом, модель имеет доступ только к тем данным, к которым имеет доступ ее владелец.

Если вы создаете копию определенной модели в другой учетной записи пользователя 478, возможно, копия не будет нормально функционировать, если новый владелец модели не имеет прав доступа к данным, передаваемым привязками клонированной модели.

Модификация модели

Только пользователи с уровнем прав доступа 486 *Администратора* в контексте модели могут менять конфигурацию модели. Это обеспечивает максимальную безопасность для потенциально опасных операций модели.

Безопасность пользовательских функций модели

Функции 818 модели, имеющие пользовательские реализации, потенциально могут получить доступ к любым данным сервера. Таким образом, системные администраторы, разрабатывающие код реализации функции, должны всегда соблюдать модель защиты и использовать только экземпляр объекта `CallerController`, который был передан в `FunctionImplementation.execute()`.

Бесопасность экземпляров инстанцируемой модели

Если контейнеры экземплярной 811 модели присоединены к контекстам пользователя 1608 (см. Использование экземплярных моделей 814), может показаться, что каждый экземпляр наследует права доступа пользователя к тому контексту, к которому прикреплен контейнер. Но это не так! Каждый экземпляр наследует права доступа владельца модели.

12.10.9 Производительность моделей

Модели - это сложные активные ресурсы сервера, которые имею значительное влияние на производительность. "Механизм" модели - это ее привязки 814 и правила 813.

Влияние моделей на производительность - это сложение следующих показателей:

- Количество экземпляров модели. Абсолютная 811 модель имеет всего один экземпляр, в то время как новый экземпляр относительной 811 модели создается для каждого ресурса, к которому она прикреплена. То есть, одна относительная модель может иметь тысячи экземпляров.

- Количество привязок модели.
- Частота обработки привязок модели. Она может быть явно определена в опциях привязок (для Периодических привязок) или косвенно определена частотой событий или изменениями состояния, вызывающих выполнение привязок (для привязок При Событии).
- Частота чтения/записи переменных модели, вызовов функций модели и генерирования событий модели, выполняемые другими активными компонентами системы (такими как тревоги 779, датчики 218 или виджеты 943).
- Сложность и влияние выражений привязок и выражений правил модели 813. См. производительность выражений 141 для получения более подробной информации.
- Влияние записей целей привязок. Запись цели привязки - это в большинстве случаев запись переменной контекста или вызов функции контекста. См. производительность переменных 70 и производительность функций 73 для получения более подробной информации.
- Влияние пользовательского кода Java, реализующего функции модели.

Настройки параллельного выполнения привязок 816 модели можно установить, если модель не способна обрабатывать все свои привязки вовремя.

12.11 Планировщик задач

Основной целью планировщика является автоматическое выполнение в неинтерактивном режиме действий 87 согласно определенному пользователем расписанию, например:

- Проверка статуса устройства каждые два часа и запуск внешнего приложения в случае проблемы
- Очистка ОЗУ устройства каждое воскресенье в 4 часа утра
- Отправка журнала отчета посещаемости по электронной почте 2-го и 17-го числа каждого месяца в 4 часа вечера в течение 2009 и 2010 годов
- Обновление прошивки тысячи устройств в данный момент или в 3 часа утра следующей ночью

Любое действие 87 или функция 70, доступные внутри AtomMind, могут быть запланированы для переодического выполнения. Если действие интерактивное и требует ввода информации пользователем, параметры ввода могут быть установлены заранее в процессе планирования. При перезагрузке сервера планировщик может определить, есть ли задачи, которые были пропущены (не были запущены) пока сервер был выключен.

Существует два вида расписаний:

- **Простое расписание**. Согласно простому типу рассписаний, действие выполняется определенное количество раз с заданным интервалом. Также можно задать время начала и конца интервала.
- **Расширенное расписание**.Этот тип расписания позволяет создавать сложные шаблоны выполнения, такие как "выполнять каждую минуту начиная с 2:00 до 2:59 часов дня каждый день" или "выполнять в 10:15 утра в последнюю пятницу каждого месяца в течение 2009, 2011, 2013 и 2015 годов".

Конфигурация задачи

Для запуска задачи необходимо следующее:

- **Маска контекста** 44, совпадающая с контекстами, для которых должно быть выполнено действие
- **Имя** выполняемого действия
- Любой **параметр (параметры) входа**, необходимые для выполнения действия
- По крайней мере один **триггер** для определения времени выполнения действия

Планировщик постоянно сохраняет своё внутреннее состояние. Это означает, что при перезапуске сервера планировщик может определить, выполнение каких запланированных задач было пропущено при выключении сервера.

Планировщик выполняет действия в неинтерактивном режиме 99. Действия, предназначенные для интерактивного режима, не могут быть запланированы.

Основные свойства запланированных задач описаны здесь 827.

Каждый пользователь 478 обладает своим набором запланированных задач.

Сопутствующая документация:

- Отправка запланированного отчета по почте 1682
- Планирование времени простоя устройства 1697

Отслеживание истории выполнения

Используйте действие Отслеживать связанные события 1091 контекста задач 1586 для просмотра:

- истории выполнения задач
- ошибок, найденных выполненным действием

Администрирование планировщика

Для администрирования планировщика используются два контекста: общий контекст Запланированные задачи 1585, который выступает в роли контейнера, и контекст Запланированная задача 1586, который содержит информацию об одном запланированном действии.

Встроенные запланированные задачи

Некоторые запланированные задачи встроены в AtomMind Server:

- **Проверять входящую почту**. AtomMind Server использует данную запланированную задачу для просмотра почты с сервера входящих сообщений. Это необходимо для подтверждения тревог через e-mail сообщения 806.
- **Удалить просроченные события**. Данная задача отвечает за периодическое удаление просроченных событий из истории событий 73. Не рекомендуется ее удалять или изменять настройки.
- **Запланированный перезапуск сервера**. Эта задача перезапускает AtomMind Server в момент, когда операция перезапуска была определа администратором.
- **Запланированное выключение сервера**. Эта задача выключает AtomMind Server в момент, когда операция выключения была определена администратором.

12.11.1 Триггеры задач

Запланированные задачи могут быть активированы одним из двух типов *триггеров*.

Триггер - действие, запускающее некоторые события.

Простые триггеры

Согласно одному простому триггеру планировщика, действие выполняется определенное количество раз за определенный срок. Кроме того, можно указать начало и конец интервала времени, когда будет выполняться действие. Можно определить более одного триггера данного типа. Когда выполнение прекращается (это может произойти из-за окончания количества повторов или по достижению времени окончания), соответствующий триггер автоматически удаляется из расписания. Свойства простых триггеров описаны здесь 828.

Если простой триггер был настроен с ограниченным Количеством повторов, запись о триггере автоматически удаляется из таблицы триггеров сразу после завершения всех запланированных выполнений.

Расширенные триггеры

Расширенные триггеры планировщика позволяют выполнять сложные шаблоны задач, такие как "выполнять ежедневно, каждую минуту, начиная с 14.00 и заканчивая в 14.59" или "выполнять в 10:15 каждую пятницу каждого месяца в течение 2009, 2010, 2011 и 2012 года". Можно определить более одного триггера данного типа. Триггеры автоматически удаляются из расписания, если в будущем не ожидается выполнений никаких действий, но на практике это происходит, только если опция "Год" установлена на ограниченное количество лет. Свойства нестандартных триггеров описаны здесь 828.

Невозможно назначить запланированное выполнение задачи для комбинаций "День месяца" и "День недели" одновременно. Если шаблон выполнения для любого "Дня месяца" имеет значение "Не никогда", шаблоны для "День недели" должны быть установлены на "Никогда" и наоборот.

ПРИМЕРЫ РАСШИРЕННЫХ ТРИГГЕРОВ

День месяца	Все дни : Всегда (выполнять каждый день месяца)
Месяц	Все месяцы : Включено
День недели	Все дни : Никогда (отключено, в данном шаблоне используется "День месяца")
Год	
Время	09:00:00
Дата/время запуска	<Не установлено>
Дата/время окончания	<Не установлено>
Шаблон выполнения	Ежедневно в 9:00.

День месяца	Все дни : Всегда
Месяц	Все месяцы : Включено
День недели	Все дни : Никогда
Год	2005
Время	09:00:00
Дата/время запуска	<Не установлено>
Дата/время окончания	<Не установлено>
Шаблон выполнения	Ежедневно в 9:00 в течение 2005 года.

День месяца	Все дни : Никогда
Месяц	Март : Включено, другие месяцы : Отключено
День недели	Среда : Всегда, другие дни : Никогда
Год	
Время	17:55:00
Дата/время запуска	<Не установлено>
Дата/время окончания	<Не установлено>
Шаблон выполнения	Каждую среду марта в 17:55.

День месяца	Все дни : Никогда
Месяц	Все месяцы : Включено
День недели	Пн - Пт : Всегда, другие дни : Никогда
Год	
Время	05:15:00
Дата/время запуска	<Не установлено>
Дата/время окончания	<Не установлено>

Шаблон выполнения	Каждый понедельник, вторник, среду, четверг и пятницу в 5:15.

День месяца	15 числа : Всегда, другие дни : Никогда
Месяц	Все месяцы : Включено
День недели	Все дни : Никогда
Год	
Время	09:00:00
Дата/время запуска	<Не установлено>
Дата/время окончания	<Не установлено>
Шаблон выполнения	15 числа каждого месяца в 9:00.

День месяца	Все дни : Никогда
Месяц	Все месяцы : Включено
День недели	Пт: Последняя, другие дни : Никогда
Год	
Время	09:00:00
Дата/время запуска	<Не установлено>
Дата/время окончания	<Не установлено>
Шаблон выполнения	В последнюю пятницу каждого месяца в 9:00.

День месяца	Все дни : Никогда
Месяц	Январь - март : Включено, другие месяцы : Отключено
День недели	Пт: Последняя, другие дни : Никогда
Год	2006 - 2008
Время	09:00:00
Дата/время запуска	<Не установлено>
Дата/время окончания	<Не установлено>
Шаблон выполнения	В каждую последнюю пятницу января, февраля и марта в 9:00 в течение 2006, 2007 и 2008 года.

День месяца	Все дни : Всегда
Месяц	Все месяцы : Включено
День недели	Все дни : Никогда
Год	
Время	09:00:00
Дата/время запуска	01.07.2007 00:00:00
Дата/время окончания	01.08.2008 00:00:00
Шаблон выполнения	Ежедневно в 9:00:00, первое выполнение 01.07.2007, последнее выполнение 31.07.2007 09:00:00.

12.11.2 Обработка параметров действий

Входные параметры 827 запланированного действия заранее определяются на стадии создания запланированных задач и сохраняются в базе данных 692. Однако часто необходимо менять/обновлять эти параметры во время выполнения запланированной задачи. Для этого необходимо:

- Открыть любую таблицу параметров для редактирования. Возможно добавлять/менять ее привязки 397.
- Добавить в таблицу одну или более привязок.

Эти привязки рассчитываются во время выполнения задачи. Они обновляют значение заранее определенных параметров действия.

См. привязки входных параметров 100 для получения более подробной информации.

12.11.3 Настройка задач

Данный раздел посвящен свойствам настройки запланированных задач.

Все свойства, описанные в данном разделе, доступны через выполнение действия Настроить 105 контекста задачи 1586.

12.11.3.1 Свойства задач

Данное свойство определяет базовые опции запланированных задач.

Описание поля	Имя поля
Имя задачи. Имя контекста запланированной задачи. Должно соответствовать соглашениям о наименованиях 42. Имя необходимо для ссылки на данную задачу из других частей системы.	name
Описание задачи. Текстовое описание задачи. Является также описанием 43 контекста запланированной задачи.	description
Задача активна. Все триггеры включенных задач активны, и задача выполняется. При повторном запуске задачи после некоторого времени, когда она была отключена, выполняются все пропущенные триггеры.	enabled
Маска контекстов. Маска 44 контекста определяет количество контекстов, из которых будет выполняться действие. При запуске запланированной задачи начинается выполнение действия в последовательном порядке из каждого контекста, соответствующего маске.	mask
Тип. Тип задачи, функция или действие.	type
Функция. Имя запускаемой функции, если **Типом** задачи является функция.	function
Действие. Имя запускаемого действия, если **Типом** задачи является действие.	action
Параметры. Параметры действия. Эти параметры используются для замены пользовательского ввода, когда запланированная задача выполняет действие в неинтерактивном режиме. Например, если запланированное действие требует подтверждения (например, запрашивает у пользователя "Удалить запрос?" и позволяет нажать кнопку ОК или Отмена), это поле будет содержать параметр "Удалить запрос?" с двумя возможными вариантами: "ОК" и "Отмена".	input
Выполнять пропущенные задачи. Если включена эта опция, все пропущенные задачи (например, при остановке или перезапуске сервера AtomMind) будут выполнены после загрузки сервера. Если опция отключена, пропущенные задачи игнорируются.	executeMissed
Условие. Данное выражение вычисляется для каждого контекста, соответствующего заданной **маске контекстов**. Если выражение возвращает false, выполнение задания пропускается для текущего контекста.	condition

Среда вычисления 114 выражения условия:		
Контекст по умолчанию 119	Контекст, для которого предполагается выполнение задачи.	
Таблица данных по умолчанию 120	None.	
Ряд по умолчанию 119	0	
Переменные среды 123	Только стандартные 123 переменные.	
Остановить в случае успеха. Если данная опция включена, то при первом удачном непропущенном выполнении задания для целевого контекста, выполение задания прекращается для данного контекста.		stopOnSuccess
Количество повторов. Максимальное количество попыток выполнения задания для конкретного контекста, если предыдущие выполнения завершились с ошибкой.		retryCount
Количество потоков по умолчанию. Базовый размер пула потоков, выполняющих задание, т.е. количество параллельных заданий в случае их нормального выполнения.		jobThreads
Максимальное количество потоков. Максимальный размер пула потоков, выполняющих задание, т.е. количество параллельных заданий в случае их медленного выполнения.		maxJobThreads

Данные свойства доступны через переменную jobDetailsView 1587.

12.11.3.2 Простое расписание

Данное свойство определяет один или более простые триггеры для запланированной задачи и имеет многострочный формат.

Описание поля	Имя поля
Дата/время начала. Время, до которого выполнение согласно данному триггеру невозможно. Значение по умолчанию - *<Не установлено>,* поэтому выполнение начинается сразу же после создания триггера.	startTime
Дата/время окончания. Время, после которого выполнение согласно данному триггеру невозможно. После завершения выполнения триггер удаляется при достижении данного времени. Значение по умолчанию - *<Не установлено>,* поэтому выполнение заканчивается по истечении количества повторов или продолжается, если значение "Количество повторов" установлено на 0.	endTime
Количество повторов. Количество выполнений. Выполнение прекращается и триггер удаляется по окончании определенного количества повторов выполнения задачи согласно данному триггеру. Значение по умолчанию равно 0, что означает "выполнять неограниченное количество раз".	repeatCount
Интервал повтора. Задан в секундах.	repeatInterval

Простое расписание доступно через переменную triggersView 1588.

12.11.3.3 Расширенное расписание

Данное свойство определяет один или более расширенные триггеры для запланированной задачи и имеет многострочный формат.

Описание поля	Имя поля

День месяца. Определяет шаблон выполнения для каждого дня месяца: всегда, в ближайшие день недели или никогда.	dayOfMonth
Месяц. Определяет, для каких месяцев назначено выполнение.	month
День недели. Определяет шаблон выполнения для каждого дня недели: всегда, в первый, второй, третий, последний или никогда.	dayOfWeek
Год. Определяет количество лет, в течение которых должно осуществляться выполнение задачи. Представляет собой комбинацию разделенных запятой лет или годовых интервалов, т.е. "2003, 2004, 2006-2008". Пустое значение приводит к постоянному выполнению каждый год.	year
Время. Время выполнения. Задача выполняется ежедневно в данное время, для которого назначено выполнение другими опциями триггера. Единственным способом назначить дополнительное время выполнения является создание нескольких триггеров.	time
Дата/время начала. Время, до которого выполнение согласно данному триггеру невозможно. Значение по умолчанию - *<Не установлено>,* поэтому выполнение начинается сразу же, как только позволит комбинация других опций.	startTime
Дата/время окончания. Время, после которого выполнение согласно данному триггеру невозможно. Триггер удаляется автоматически при достижении данного времени.	endTime

Расширенное расписание доступно через переменную cronTriggersView 1586.

12.11.4 Безопасность задач

Любая запланированная задача выполняется при наличии прав доступа 477 ее владельца. Таким образом, задача может проверять и обрабатывать только те данные, которые доступны для ее учетной записи пользователя 478.

Пример: Задача, принадлежащая учетной записи пользователя Джо, не может останавливать или перезапускать AtomMind Server, если у Джо нет уровня прав доступа 486 **Администратора** в корневом контексте 1559. При этом задача администратора по умолчанию 479 всегда может остановить/перезапустить сервер, поскольку у него есть права доступа для осуществления любых действий.

12.11.5 Производительность задач

Задачи - это активные компоненты сервера, которые могут вызывать перегрузку процессора и памяти при каждом выполнении. Количество потребляемых ресурсов полностью определяется действием, вызванным задачей, от малозначительной (например, действие "отправить SMS") до высоко нагруженной (например, удаление просроченных событий из базы данных).

12.12 Запросы

Язык структурированных запросов SQL представляет собой мощный язык запросов для извлечения и обработки данных. ТВЭЛ AtomMind Server изначально поддерживает SQL, тем самым обеспечивая легкий доступ как к внутренним данным, так и к данным оборудования. Специальный диалект SQL, поддерживаемый AtomMind Server, называется "Язык запросов AtomMind".

Язык запросов AtomMind высоко интергрирован в AtomMind Server. Поскольку все внутренние данные AtomMind Server представляют собой таблицы данных, очень удобно использовать запросы SQL для модификации свойств (настроек, переменных) различных объектов системы. Таблицы данных содержат актуальные данные, правила проверки корректности данных, подробную информацию о каждом поле и т.д.

Запросы нужны для:

- Просмотра/редактирования нескольких свойств множества ресурсов/устройств в одной сводной таблице
- Нахождения/фильтрации данных и активации тревог, если выбранные табличные данные удовлетворяют некому условию
- Построения отчета
- Экспорта данных во внешнюю систему или файл

- Сортировки и фильтрации существующих табличных данных

Данный раздел не является руководством по языку SQL. Предполагается, что вы уже знакомы с SQL и его синтаксисом. Мы не пытаемся усложнить вашу задачу - язык запросов SQL представляет собой объемную тему, по которой написано достаточное количество книг и руководств. Поэтому, если вы не знакомы с SQL, вам необходимо получить общее представление о предмете. Как только вы этого добьетесь, возвращайтесь к данному разделу, чтобы узнать, как мы его выполняем в AtomMind. В разделе ссылки SQL 833 мы привели перечень источников, где вы можете найти более подробную информацию о языке запросов SQL.

Цель языка запроса

Язык запроса AtomMind имеет два применения:

1. Получение данных от объектов сервера или оборудования, фильтрация, сортировка, группировка и их представление пользователю в удобной форме (в виде таблицы).
2. Модификация свойств различных контекстов (например, конфигурация различных аппаратных средств) посредством ввода значений для настроек в таблице.

Первая задача выполняется, используя запрос выборки обычных данных ("ВЫБРАТЬ"), создавая таблицу данных, которую можно увидеть в пользовательском интерфейсе AtomMind Server (таком как AtomMind Client).

Например, вы можете написать запрос, который отображает трафик всех Device Server в системе в виде удобной таблицы:

	Owner Name	Device Server Name	Bytes transferred from LinkServer to Device Server	Bytes transferred from Device Server to LinkServer
1	admin	agent	336756744	1112355301
2	admin	c1	75	0
3	admin	c2	0	0
4	admin	c3	204	0
5	admin	c4	0	0
6	admin	c5	17190607705	33353381388
7	admin	c6	0	0
8	admin	c7	0	0
9	admin	c8	0	0
10	admin	ftest	4092428638	6743462742
11	admin	http	43083	953292
12	admin	te	0	0
13	test	c5	0	0
14	test	ftest	0	0
15	test	test	0	0
16	trew	fds	0	0

Запросы также представляют результаты, которые можно редактировать, позволяя пользователю вводить данные, например, для конфигурации нескольких экземпляров одного устройства. Итак, вы можете использовать запрос для доступа к настройкам всех Device Server в системе и редактировать их в одной таблице:

	Owner name	Device name	Network Password	IP-address	Destination IP-address	Destination port	Connection t...	Gateway IP-address	Subnet mask	DHCP	Auto-registration on Link Server	MAC-address	Po
1	admin	http		192.168.1.125	192.168.1.2	6450	10	127.0.0.1	0.0.0.0	0- Disabled	1- Enabled	<Not set>	<N
2	admin	c2	<Not set>	192.168.1.114	192.168.1.2	6450	10	127.0.0.1	255.255.255.0	0- Disabled	1- Enabled	0.2.3.4.20.225	<N
3	admin	c3	<Not set>	192.168.1.117	192.168.1.2	6450	10	127.0.0.1	255.255.255.0	0- Disabled	1- Enabled	0.2.3.4.41.162	<N
4	admin	c6	<Not set>	192.168.1.99	192.168.1.2	6450	10	0.0.0.1	255.255.0.0	0- Disabled	1- Enabled	0.2.3.5.20.128	<N
5	admin	c5	<Not set>	192.168.1.118	192.168.1.131	6450	10	127.0.0.1	255.255.0.0	0- Disabled	1- Enabled	0.2.3.5.21.39	<N
6	admin	ftest	<Not set>	192.168.1.120	192.168.1.131	6450	10	127.0.0.1	255.255.0.0	0- Disabled	1- Enabled	0.6.6.6.6.42	<N

Запросы также могут быть использованы в различных средствах AtomMind Server. Например, вы можете установить тревогу, которая будет активирована, если результат запроса удовлетворит некоторому условию (см. пример Проблемы оборудования 809 в главе Тревоги 779 для получения более подробной информации). Результаты запроса могут конвертироваться в пригодный для печатания отчет во время их просмотра (например, используя свойство "Сформировать отчет" редактора таблицы данных 382 в AtomMind Client). Также могут быть использованы виджеты 943 для представления данных, ранее собранных и обработанных запросом.

Отличия от классического SQL

Существует два главных отличия между "обычным" SQL, используемым большинством процессоров баз данных (таких как MySQL или Oracle) и языком запросов AtomMind.

ПЕРВОЕ ОТЛИЧИЕ: ТАБЛИЦЫ ПРОТИВ КОНТЕКСТОВ

Обычный SQL использует таблицы в качестве главного источника данных. Каждый запрос "ВЫБРАТЬ" ссылается на одну или более таблицы, из которых выбираются данные. Ссылка на данные указываются в разделе "ОТ".

Язык запросов AtomMind работает с данными, которые хранятся в *контекстах* 41 *AtomMind Server.* Доступ к этим данным осуществляется при получении контекстных переменных 61 (свойств) или при вызове контекстных функций 70. Значение каждой переменной контекста представляются таблицей данных 49, которая имеет поля,

записи и ячейки. Эти таблицы данных используются процессором языка запросов в качестве таблиц, из которых выбираются данные.

Тот же метод может использоваться для ссылки на таблицы, содержащие значения выхода функций контекста. Когда запрос "ВЫБРАТЬ" ссылается на такую таблицу (содержащую возвращаемые функцией значения), ссылка должна включать входные параметры для данной функции. Когда запрос выполнен, вызов функции осуществляется с данными параметрами. Данная функция затем формирует таблицу данных, содержащую возвращаемые значения, которые, в свою очередь, будет использоваться запросом.

В статье Синтаксис языка запросов 858 данные ссылки на переменные контекста или функции называются **contextReference** 859. Каждая **contextReference** может иметь альтернативное имя (**tableAlias**), которое используется для ссылки на нее из других частей запроса.

Подобно обычному SQL, язык запросов AtomMind поддерживает вложенные запросы, где раздел "ОТ" запроса "ВЫБРАТЬ" содержит другой запрос "ВЫБРАТЬ".

ВТОРОЕ ОТЛИЧИЕ: ССЫЛКИ НА ПОЛЯ В РАЗДЕЛЕ "ОТ"

Второе отличие касается ссылки на поля таблиц, указанные в разделе "ОТ".

В обычном SQL ссылка на поле таблицы состоит из двух частей: **tableAlias.fieldName**. Первая часть - имя таблицы, ее алтернативное имя в разделе "ОТ" или альтернативное имя для вложенного запроса "ВЫБРАТЬ". Вторая часть представляет собой имя поля.

в AtomMind ссылка на поле таблицы состоит из трех частей. Выглядит это таким образом: **tableAlias.fieldAlias**. **tableAlias** - альтернативное имя для таблицы**,** формируемой из данных контекста (см. выше Первое отличие 830). **fieldAlias в** форме **entityName$fieldName.** *entityName* - имя переменной контекста или функции (такой как **childInfo**, которая является переменной "Информация о пользователе" контекста **пользователь**). *fieldName* является специальным полем переменной или возвращаемым функцией значением (такой как **firstname**, имя пользователя для контекста **userInfo**).

Те, кто уже знаком с языком SQL, могут использовать следующую аналогию, чтобы быстро понять язык запросов AtomMind: запрос в AtomMind такой же, как и в обычном запросе SQL, с некоторыми исключениями:

1. Имена таблиц, используемые в обычном SQL, заменяются на контекстные ссылки 833. Однако каждая контекстная ссылка используется для построения *таблицы* до выполнения запроса. Поэтому процессор языка запросов AtomMind обрабатывает таблицы подобно обычному движку базы данных.
2. Имена полей, используемые в обычном SQL, заменяются на ссылки на поля 836. Но данные ссылки на поля служат так же, как и обычные имена полей в классическом запросе SQL. Просто у них немного отличается синтаксис, позволяя пользователю ссылаться на поля "виртуальных" таблиц, формируемых из контекстных ссылок. Чтобы увидеть точную разницу в синтаксисе и заложенной в нем логике, пожалуйста, прочитайте о ссылках на поля 836.
3. При выполнении запроса SQL вы получаете информативную таблицу, содержащую только значения данных. Однако при выполнении запроса AtomMind вы получаете то, что мы называем Таблицей данных 49, которая более рациональна, чем "просто" таблица - она содержит условия данных, типы полей и другие подобные метаданные.

Поток выполнения запросов

Данный раздел представляет пошаговое описание того, как используется текст запроса для формирования результатов запроса.

1. На первой стадии процессор запроса находит все контекстные ссылки 833 в разделе "ОТ" текста запроса. Данные ссылки затем используются для построения специально подобранных для этого таблиц, содержащих данные из действующих контекстов, для которых в дальнейшем будет выполнен запрос. Данный процесс описан здесь 833.
2. Классический запрос SQL выполняется с использованием таблиц, которые были созданы на предыдущей стадии. Запрос формирует таблицу, построенную согласно синтаксису запроса.
3. Далее эта таблица конвертируется в таблицу данных 49. Это облегчает понимание: конечная таблица данных содержит описания всех полей, правила форматирования и подтверждения соответствия, а также любые другие метаданные, которые могут быть выбраны из таблиц данных, для которых был выполнен запрос. Итак, в сущности, AtomMind использует запрос для создания таблицы данных. Мы упомянули здесь предыдущий шаг (2) только для того, чтобы дать вам краткое представление о внутреннем устройстве системы.
4. Результат запроса показывается пользователю. Если результат можно редактировать, пользователь может внести изменения в представленные данные.
5. Если пользователь сохраняет измененные результаты, процессор выполнения запроса снова записывает все измененные данные в переменные контекста, из которых они были получены.

Как было указано в предыдущей главе, различие между запросом AtomMind и обычным запросом SQL заключается в том, что он выполняется для "виртуальных" таблиц, формируемых из данных контекста, нежели чем из "простых" таблиц.

Обзор синтаксиса запроса

Единственной операцией, которую поддерживает язык запросов AtomMind, является операция выборки, которая выполняется с ключевым словом ВЫБРАТЬ. ВЫБРАТЬ получает данные из назначенной таблицы или из нескольких связанных таблиц, которые создаются для данного случая, когда обрабатываются контекстные ссылки 833 в запросе.

Каждый запрос имеет несколько разделов:

- Основной раздел **ВЫБРАТЬ**, определяющий, какие поля оригинальных таблиц должны быть включены в результат запроса.
- Раздел **ОТ**, указывающий исходную таблицу или таблицы, из которых извлекаются данные. Раздел ОТ может включать дополнительный раздел **ПРИСОЕДИНИТЬ**, присоединяя связанные таблицы друг к другу на основе пользовательских критериев.
- Раздел **ГДЕ** включает условное выражение, которое используется для фильтрации данных, полученных при выполнения запроса. Раздел ГДЕ применяется до раздела ГРУППИРОВАТЬ ПО. Технически раздел ГДЕ исключает все строки из результатного набора, где условное выражение не является true.
- Раздел **ГРУППИРОВАТЬ ПО** используется для комбинирования или группирования строк со связанными значениями. ГРУППИРОВАТЬ ПО часто используется для подсчета статистики (общей, средней и т.д.) схожих строк или исключения повторяющейся строки из результата запроса.
- Раздел **ИМЕТЬ** включает условное выражение, используемое для исключения строк после применения раздела ГРУППИРОВАТЬ ПО к результату.
- Раздел **ОБЪЕДИНИТЬ** позволяет совмещать результаты двух и более разделов запроса ВЫБРАТЬ в один результат.
- Раздел **УПОРЯДОЧИТЬ ПО** используется для определения столбцов, используемых для сортировки полученных данных, а также, должны ли они быть восходящими или нисходящими. Порядок строк, возвращаемых запросом, не определен до тех пор, пока не задан раздел УПОРЯДОЧИТЬ ПО.
- Раздел **ОГРАНИЧИТЬ** ограничивает итоговый результат предварительно заданным количеством строк. Это область числовых значений. Например, если запрос выдает результат в 100 записей, а вам нужно 10 произвольных записей (первые десять, последние десять, десять записей, начиная с 15 записи и т.д.), вы можете использовать раздел ОГРАНИЧИТЬ, чтобы получить столько строк, сколько вам необходимо.

Язык запросов AtomMind поддерживает большинство свойств стандартных запросов выборки SQL. Среди них вложенные запросы ВЫБРАТЬ, функции агрегации SQL, используемые для вычислений различных значений данных (COUNT, MIN, MAX, SUM, AVG и т.д.), и встроенные функции 860 (числовые, строковые, даты/времени и системы).

Более подробную информацию о синтаксисе языка запросов AtomMind вы найдете здесь 858.

Администрирование запросов

Для администрирования запросов используются два контекста: общий контекст Запросы 1546, который служит контейнером, и контекст Запрос 1548, который содержит информацию об одном запросе. См. ниже описания.

Каждый пользователь 478 имеет свой набор запросов.

Встроенные запросы

Некоторые запросы уже встроены в AtomMind Server и появляются под учетной записью администратора по умолчанию 479:

- **Все пользователи**. Данный запрос позволяет просмотр и редактирование настроек всех учетных записей пользователей 478 в одной таблице.
- **Сводная информация о статусах** Device**.** Отображает статус подключения и синхронизации каждого Device.
- **Отключенные Device.** Отображает все отключенные Device.

- **Активные тревоги**. Отображает ожидающие и эскалированные тревоги 779.

Данные запросы описаны в главе Примеры запросов 843.

SQL ссылки

Для получения более подробной информации о синтаксисе языка SQL перейдите по следующим ссылкам:

http://en.wikipedia.org/wiki/SQL

http://www.w3schools.com/sql/sql_select.asp

http://sqlzoo.net/

http://www.fluffycat.com/SQL/

http://www.baycongroup.com/tocsql.htm

12.12.1 Контекстные ссылки

Контекстная ссылка выглядит таким образом:

```
contextMask { :contextEntityReference [: ...] }
```

contextMask является маской 44 контекстов, чьи сущности (переменные или функции) будут использоваться для построения таблиц, по которым будет выполнен запрос. Данная маска может быть простым именем контекста (без каких-либо специальных символов) или разрешаться в некоторые контексты.

contextEntityReference указывает на переменную контекста, группу переменных или функцию. Выглядит это данным образом:

```
{ contextVariableName | contextVariableGroupName.* | contextFunctionName
( contextFunctionParameters ) }
```

Вертикальные черты указывают альтернативы - фактически они не включены в *contextEntityReference.* Далее приведен простой пример контекстной ссылки:

users.admin:childInfo

данная ссылка указывает значение переменной **childInfo** из контекста **users.admin**.

Двоеточие (:) используется для разделения маски контекста и сущности. "Сущность" - придуманное название переменной или функции. Итак вы задаете маску (т.е. "куда" вы направляетесь), затем ставите двоеточие, чтобы обозначить сущность (т.е. "что" вы хотите найти в месте назначения). Вы даже можете использовать специальный символ после двоеточия для ссылки на группу переменных (такую как **user.charlie.devices.sensor:remote.***, которая даст все настройки (переменные) в группе датчиков устройства, принадлежащего пользователю **charlie**).

Каждая контекстная ссылка может сочетать маски контекста и несколько ссылок на сущности (даже разных типов):

users.*:userInfo:variableGroup.*:status()

Данная сылка указывает значение переменной **userInfo.** Все переменные, принадлежащие группе **variableGroup** и выходу функции **status**, вызываются без параметров. Значения извлекаются из всех контекстов, относящихся к маске **users.***, т.е. контекстов всех видимых пользователей в системе с правами доступа пользователя, выполняющего запрос.

Построение исходных таблиц из контекстных ссылок

Во время выполнения запроса сперва каждая контекстная ссылка используется для построения одной таблицы, по которой будет выполнен запрос. Построение данной таблицы представляет собой сложный процесс из нескольких шагов:

1. Маска контекста разрешается в список контекстов к ней относящихся и доступных пользователю, выполняющему запрос (согласно его правам доступа 477).

2. Процессор запроса выбирает и кэширует значения переменных, на которые указывают контекстные ссылки. Он также вызывает все связанные функции с параметрами, заданными в тексте запроса, и кэширует выходные данные функций.

3. Формируется список полей для конечной таблицы согласно правилам. Если контекстная ссылка ссылается только на одну переменную или функцию, конечная таблица будет иметь столько же полей, сколько их содержит переменная или вывод функции. Если контекстная ссылка ссылается на несколько переменных или функций, формат конечной таблицы будет включать **все** поля, которые есть в переменных или в выводе функции. Исключение оставляет случай, когда значение или вывод функции содержит более одной записи. В такой ситуации таблица будет иметь только одно поле для данной переменной/функции. Данное поле будет содержать вложенную таблицу со значением переменной или вывода функции

4. Таблица заполняется данными. Обычно одна запись создается для каждого контекста, соответствующего маске контекстной ссылки. Но если ссылка ссылается только на одну переменную/функцию и ее значение/вывод содержит несколько записей, все эти записи включены в конечную таблицу (т.е. таблица будет содержать не одну запись, а запись будет представлена в виде вложенной таблицы, что не имеет смысла).

Нажмите сюда 859, чтобы узнать, где контекстные ссылки могут применяться в синтаксисе языка запросов.

ОПРЕДЕЛЕНИЕ ПАРАМЕТРОВ ФУНКЦИИ

Когда функция контекста вызывается во время разрешения контекстной ссылки, ей необходимо иметь входные параметры. Данные параметров определены в списке, содержащем строковые параметры 859. Параметры входят в одну длинную строку и разделяются запятыми (",").

Более подробную информацию о том, как список разделенных запятыми параметров конвертируются в таблицу данных, см. здесь. Таблица данных построена в соответствии с форматом 50 ввода функции.

КЛЮЧЕВЫЕ ПОЛЯ

При создании таблиц, соответствующих контекстной ссылке, система создает *индексы* для определенных полей. Данные индексы существенно ускоряют процесс обработки запроса. Индексы создаются автоматически для всех *ключевых полей*, т.е. для тех, у которых выстален флажок Ключевое 51 в формате таблицы.

Если для выполнения вашего запроса, такого как запрос, ссылающийся на соединения, требуется большое количество времени, или же этот запрос негативно влияет на производительность, попробуйте изменить структуру исходных значений, чтобы нужные поля стали ключевыми. В этом случае для них автоматически будут созданы индексы, а производительность запросов многократно возрастет.

СПЕЦИАЛЬНЫЕ ПОЛЯ

Каждая таблица, построенная из контекстной ссылки, содержит три дополнительных поля: **CONTEXT_ID**, **PARENT_ID** и **RECORD_INDEX**. Данные поля добавляются процессором обработки запроса.

CONTEXT_ID содержит полное имя контекста, в котором возникла определенная запись.

PARENT_ID содержит полное имя контекста-родителя того контекста, в котором возникла определенная запись.

RECORD_INDEX - это номер записи в таблице данных, из которой эта запись была взята.

Ссылка на данные поля осуществляется через раздел ГДЕ. Однако данные поля скрыты. Они невидимы в результатах запроса.

Ссылаться на данные из специальных полей можно в любом запросе. См. использование объединений 853 для примера, как это сделать.

Имейте в виду, что имена специальных полей должны точно определяться в запросе ВЫБРАТЬ при использовании редактируемых результатов запроса 839. Для получения более подробной информации обратитесь к статье поля обратной записи 840.

Использование алиасов таблиц в качестве переменных контекста запроса в контекстных ссылках

Алиас таблицы, определенный для ссылок на результаты запроса, может использоваться как переменная контекста в контекстной ссылке в другом запросе. В этом случае контекстная ссылка выглядит так:

```
queryContextPath.queryContextName:tableAlies
```

Например, вы создали контекст запроса с именем `sineHistory`. Допустим, цель данного контекста - извлекать историю переменной `sine` виртуального устройства `virtual`, которое расположено в `users.admin.devices` (обратите внимание, что должно быть указано время хранения истории для соответствующей переменной, чтобы активировать хранение ее истории). Текст запроса выглядит таким образом:

```
SELECT * FROM utilities:variableHistory("users.admin.devices.virtual", "sine") AS data1
```

Можно использовать созданный алиас `data1` для ссылки на результаты запроса `sineHistory` в другом запросе. Предположим, что вы создали другой контекст запроса с именем `sineHistoryLast20`. В результате следующего запроса `sineHistoryLast20` вернет последние 20 записей истории:

```
SELECT * FROM users.admin.queries.sineHistory:data1 AS data2 LIMIT 20
```

Таким образом, контекст запроса `sineHistory` используется в качестве внешнего "подзапроса" для контекста `sineHistoryLast20`.

Обратите внимание, что в отличие от алиасов таблицы, нельзя ссылаться на функцию выполнить 1548 одного контекста запроса в другом контексте запрос. Формат выхода функции `execute` динамический (не содержит каких-либо предопределенных полей), поэтому запрос, ссылающийся только на эту функцию, вернет пустую таблицу (см. п.3 в разделе Построение исходных таблиц из контекстных ссылок 833 выше).

Примеры контекстных ссылок

Предположим, что у вас два контекста, **path.name** и **path.name2**. Маска контекста **path.*** подходит для обоих.

path.name имеет две переменные:

var1:

stringField
"test string"

var2:

integerField	**booleanField**
123	TRUE

Данный контекст также определяет функцию **func1**, которая возвращает следующее значение:

floatField	**integerField**
45.6	456
78.9	789

path.name2 имеет те же имена переменных, но уже с другими значениями:

var1:

stringField
"string in 2nd context"

var2:

integerField	**booleanField**
555	FALSE

Данный контекст также определяет функцию **func1**, которая возвращает следующее значение:

floatField	**integerField**
11.1	666
22.2	777
33.3	888

Пример 1

Контекстная ссылка: `path.name:var2`

Таблица, построенная из простой контекстной ссылки, будет в точности совпадать со значением **var2**:

var2$integerField	**var2$booleanField**
123	TRUE

Имейте в виду, что имена полей в полученной таблице включают переменные, из которых было взято каждое поле (см. выше **var2).** Это упрощает ссылку на данные поля, используя ссылки на поля 836.

Пример 2

Контекстная ссылка: `path.name:var1:var2`

Данная ссылка разрешится в таблицу с тремя полями, одно из значения **var1**, остальные из значения **var2**. Они включены в одну строку, т.к. обе переменные принадлежат одному и тому же контексту.

var1$stringField	var2$integerField	var2$booleanField
"test string"	123	TRUE

Пример 3

Контекстная ссылка: `path.*:var1:var2`

Данная ссылка указывает на маску контекста, поэтому она разрешится в таблицу с тремя полями, как и в предыдущем примере, но уже с двумя записями, по одной на каждый контекст:

var1$stringField	var2$integerField	var2$booleanField
"test string"	123	TRUE
"string in 2nd context"	555	FALSE

Пример 4

Контекстная ссылка: `path.*:func1`

Данная ссылка указывает на **одну** сущность (функцию) со значением, содержащим множество записей, поэтому итоговая таблица будет иметь те же поля, что и вывод функции. Общее число записей в таблице равно 5, т.к. две записи получены из значения **func1** в **path.name**, а три остальные из значения **func1** в **path.name2.**

func1$floatField	func1$integerField
45.6	456
78.9	789
11.1	666
22.2	777
33.3	888

Пример 5

Контекстная ссылка: `path.*:var1:func1`

Поскольку ссылки на сущности (переменная **var1** и функция **func1**) ссылаются на многострочные значения **func1,** они будут находиться во вложенных таблицах. **Невозможно** ссылаться на поля в таких таблицах, используя ссылки на поля 836.

var1$stringField	func1
"test string"	[Nested table with the value of "func1" in "path.name"]
"string in 2nd context"	[Nested table with the value of "func1" in "path.name2"]

12.12.2 Ссылки на поля

Раздел Контекстные ссылки 833 описывает, как используются контекстные ссылки для построения таблиц, в которых выполняются запросы. Эти таблицы содержат множество полей, на которые могут ссылаться различные части запроса, например, раздел ГДЕ.

Ссылки на поля могут состоять из двух или трех частей:

1. Если в запросе только одна контекстная ссылка в разделе ОТ, ссылка на поля таблицы, построенной из этой ссылки, состоит из двух частей:

`contextEntityName$dataTableFieldName`

В данной ссылке на поле **contextEntityName** - имя переменной контекста или функции, используемой одной контекстной ссылкой в запросе. **dataTableFieldName** - имя поля в табличном формате 49 таблицы данных, содержащей значение сущности.

Пример:

```
SELECT userInfo$firstname FROM users.*:userInfo
```

В данном примере **userInfo$firstname** ссылается на поле **firstname** таблицы данных, в которой содержится значение переменной **userInfo** во всех контекстах, соответствующих маске **users.***.

2. Если в запросе не одна контекстная ссылка, то каждая из них должна иметь *альтернативное имя*, чтобы у вас была возможность ссылаться на поля таблиц, построенных их этих ссылок. Альтернативное имя назначается при помощи ключевого слова **AS**:

```
contextReference AS tableAlias
```

Пример:

```
users.*:userInfo as ui
```

Здесь мы имеем таблицу, построенную из контекстной ссылки **users.*:userInfo** и альтернативное имя **ui**. Теперь мы можем ссылаться на поля, используя альтернативное имя.

Типичная форма контекстной ссылки, в состав которой входит альтернативное имя, выглядит следующим образом:

```
tableAlias.contextEntityName$dataTableFieldName
```

Пример:

```
SELECT ui.userInfo$firstname FROM users.*:userInfo as ui
```

Принцип действия этого запроса такой же, как и в предыдущем примере, однако, здесь на поле таблицы, построенной из "**users.*:userInfo**", ссылается само альтернативное имя. Еще пример об использовании альтернативного имени см. далее, в примере 6 852.

Нажмите сюда 859, чтобы увидеть, где могут появляться ссылки на поля в синтаксисе языка запроса.

На первый взгляд может показаться, что ссылка на поля должна состоять из двух компонентов: **tableAlias** и **fieldName**. Но язык запроса AtomMind позволяет вам создавать одну исходную таблицу (таблицу, в которой будет выполняться запрос) из значений нескольких переменных контекста или из вывода нескольких функций контекста. Именно поэтому необходим дополнительный компонент **entityName**. Он указывает на имя переменной контекста или функции, которая содержит поле **entityName**.

12.12.3 Настройка запроса

Далее приведен список свойств запроса:

Описание поля	Имя поля
Имя запроса. Имя контекста запроса. Должно удовлетворять соглашениям о наименованиях 42 контекстов. Имя требуется для ссылки на данный запрос из других частей системы.	name
Описание запроса. Текстовое описание запроса. Также является описанием 43 контекста Запрос.	description
Параметризован. Указывает, что запрос параметризован. Если флажок установлен, текст запроса содержит праматризованный запрос в формате XML, а не только простой текст.	parameterized
Текст запроса. Текст запроса. Редактируется в текстовом редакторе.	query
Исходные данные параметризатора. Данное поле используется только тогда, когда флажок "**Параметризован**" включен. Исходные данные параметризатора используются процессором параметризации для построения **текста запроса** во время выполнения запроса для всех определенных пользователем параметров. Более подробную информацию см. в разделе Параметризованные запросы 841.	parameterizer
Описания полей. Представляет собой таблицу пользовательских описаний, предназначенных для полей результата запроса. Существует два поля: **Field**, имя поля,	fields

которое появляется в итоговой таблице запроса, и **Description**, пользовательское описание поля.	
Отключить редактируемые результаты. Включает/отключает редактируемые результаты 839 для данного запроса.	disableEditableResult

Данные свойства доступны через переменную childInfo 1549.

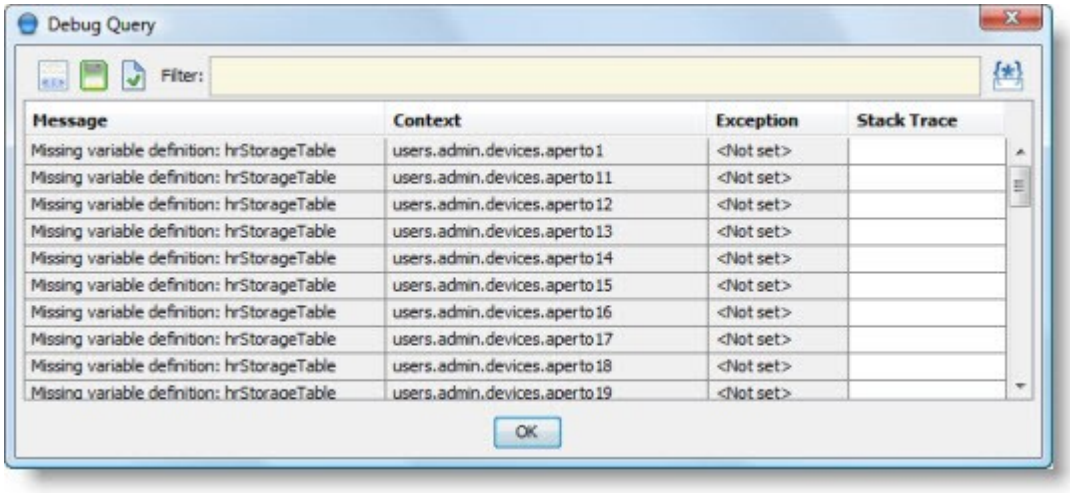
12.12.4 Отладка запросов

Некоторые запросы могут выполняться без ошибок, но возвращать непредсказуемые или неожиданные результаты. Такое поведение может возникнуть из-за некритичных ошибок, которые возникают во время выполнения запроса. Свойство отладки запросов помогает увидеть эти ошибки. Чтобы запустить запрос в режиме отладки, используйте действие Отладить запрос 1548, заданное в контексте запроса. Данное действие позволяет вам просматривать отчет отладки запроса до показа результата выполнения.

Отчет отладки состоит из следующих полей:

- **Сообщение**. Сообщение отладки.
- **Контекст (опционально)**. Контекст 41 сервера, обрабатывающий данные, послужившие причиной ошибки.
- **Исключение (опционально)**. Текст ошибки, формируемой контекстом, когда запрос получает доступ к его переменным или функциям.
- **Трассировка стека (опционально)**. Суммарная трассировка ошибки, формируемая AtomMind Server. Её может запрашивать команда технической поддержки AtomMind для решения сложных проблем. *Суммарная трассировка* является созданной компьютером информацией - она не удобна для чтения человеком.

Далее приведен пример отчета отладки (скриншот сделан в AtomMind Client):

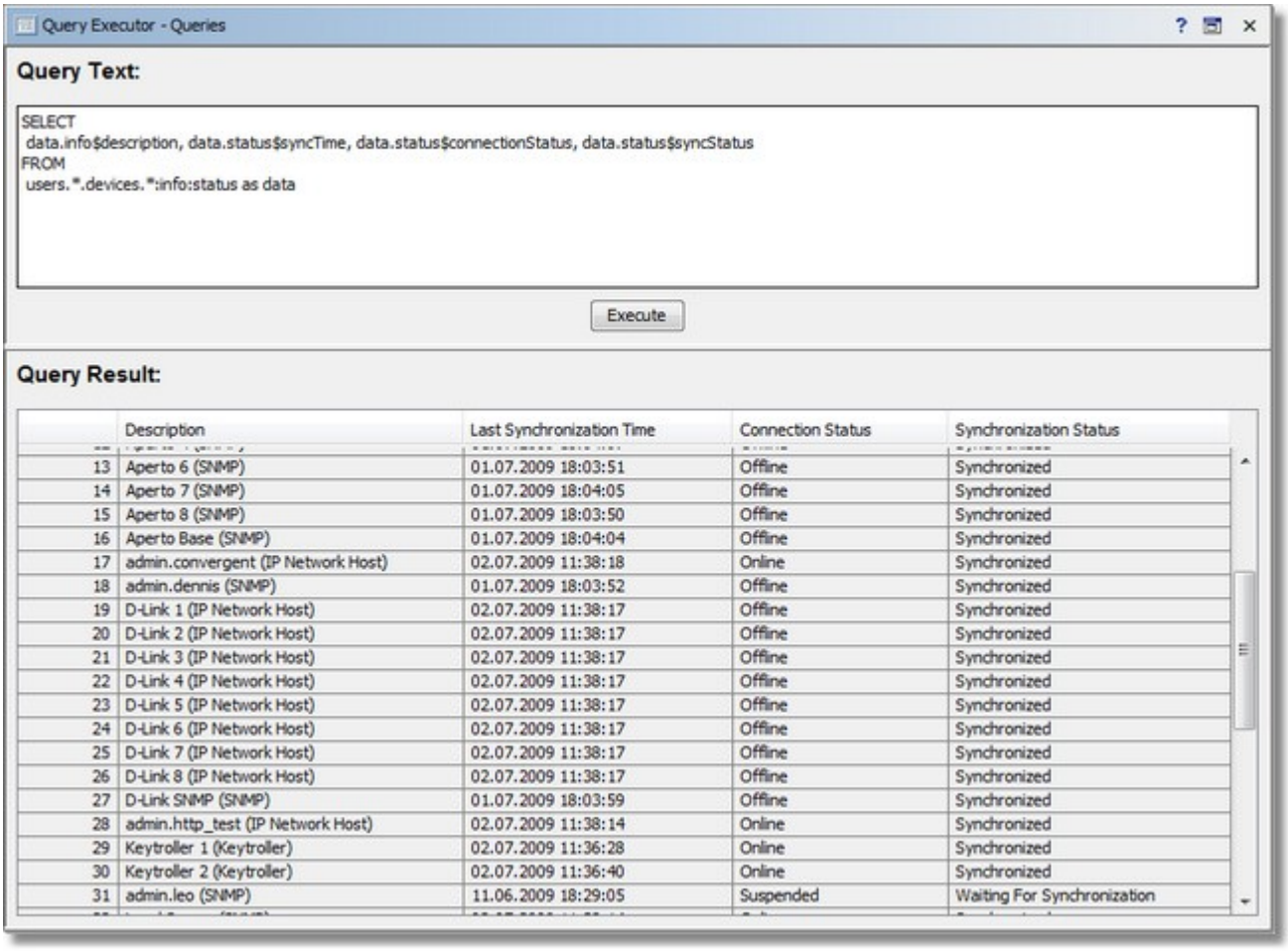


Отладка производительности запроса

Отчет выполнения запроса также показывает время, затраченное на различных этапах процесса выполнения. Эта информация необходима для оценки производительности запроса.

12.12.5 Виджет исполнитель запросов

Дистрибутив AtomMind Server по умолчанию включает виджет 943 **исполнитель запросов** для отладки запросов. Введите текст запроса и нажмите **Выполнить** для просмотра результатов запроса:



12.12.6 Редактируемые результаты запроса

Результаты выполнения некоторых запросов можно редактировать. Это уникальное свойство языка запросов AtomMind, которое помогает собирать свойства различных объектов или настроек оборудования в одну таблицу (включая только необходимые поля, упорядочивая и формируя по группам) и редактировать эти свойства в удобной форме.

Поле в результате запроса может быть изменено при выполнении следующих условий:

1. Поле должно принадлежать переменной контекста, а не выходу функции. Это очевидно, т.к. вы можете записать новое значение для переменной назад в контекст, из которого она была взята, но невозможно "записать обратно" выход функции.
2. Переменная, из которой извлечено данное поле, не должна быть только для чтения.
3. Значение данного поля не должно получаться при помощи вычислений (используя простые функции, функции агрегации или выражения языка запросов).
4. Поле не должно извлекаться из таблицы, построенной при помощи вложенного запроса.
5. Все поля исходной таблицы должны выбираться в разделе ВЫБРАТЬ при помощи ***** или **tableAlias.*** (это делать необязательно, если использовать поля обратной записи) 840.

Редактируемые результаты запроса доступны только при редактировании значения переменной **результатов запроса** (`data`) контекста запроса 1548.

Пример 1:

```
SELECT * FROM users.*:childInfo
```

Большинство полей в результате этого запроса могут быть изменены, потому что:

- Они происходят напрямую из значения переменной **userInfo**,
- Они доступны для записи в формате этой переменной,
- Они не конвертируются при помощи функции или выражения в результате запроса,
- Их отбор осуществляется при использовании * и без помощи ссылок на поля 836.

Некоторые поля (например, поле "Username") не доступны для редактирования, потому что они не определены как редактируемые в переменной **childInfo**.

Пример 2:

```
SELECT childInfo$firstname, childInfo$lastname FROM users.*:childInfo
```

Результат данного запроса не будет включать какие-либо редактируемые поля, потому что необходимые для выборки поля включаются в список прямо в разделе ВЫБРАТЬ, и поля обратной записи не определены в тексте запроса.

ПОЛЯ ОБРАТНОЙ ЗАПИСИ

Существует специальный способ выбрать определенное заранее количество полей, внося их в список раздела ВЫБРАТЬ и при этом сохраняя их доступными для редактирования. Для этого вам необходимо добавить три так называемых поля "обратной записи" к списку контекстных ссылок в запросе ВЫБРАТЬ. Данные поля помогут процессору запроса обнаружить, где должна быть сохранена информация при редактировании результатов запроса.

Формат ссылок на поля обратной записи выглядит следующим образом:

```
tableAlias.CONTEXT_ID, tableAlias.PARENT_ID, tableAlias.RECORD_INDEX
```

где **tableAlias** - альтернативное имя контекстной ссылки 833 в разделе ОТ, **CONTEXT_ID**, **PARENT_ID** и **RECORD_INDEX** - определенные заранее константы. Просто внесите их в текст запроса как есть, без всяких изменений.

Данные поля обратной записи должны всегда использоваться совместно.

Пример:

Предположим, что мы имеем запрос, который отображает статистику трафика всех серверов устройств 2087, доступных пользователю, выполняющему запрос:

```
SELECT
 info.deviceServerInfo$owner,
 info.deviceServerInfo$name,
 info.deviceServerInfo$description,
 info.deviceServerInfo$blocked,
 info.status$servertods,
 info.status$dstoserver,
FROM
 users.*.deviceservers.*:status:deviceServerInfo as info
```

Результат данного запроса не может быть изменен, потому что выбранные поля находятся прямо в разделе ВЫБРАТЬ:

	Owner Name	Device Server Name	Description	Blocked	Bytes transferred from LinkServer to Device Server	Bytes transferred from Device Server to LinkServer
1	admin	agent	Auto-registered Device Server	No	336756744	1112355301
2	admin	c1	Test DS	No	75	0
3	admin	c2	<Not set>	No	0	0
4	admin	c3	<Not set>	No	204	0
5	admin	c4	T1000	No	0	0
6	admin	c5	Auto-registered Device Server	No	17190619317	33353387814
7	admin	c6	Auto-registered Device Server 2	No	0	0
8	admin	c7	<Not set>	No	0	0
9	admin	c8	<Not set>	No	0	0
10	admin	ftest	Auto-registered Device Server	No	4092428638	6743462742
11	admin	http	<Not set>	No	43083	954012
12	admin	te	<Not set>	Yes	0	0
13	trew	fds	<Not set>	No	0	0

Чтобы сделать их редактируемыми, нужно добавить поля обратной записи к разделу ВЫБРАТЬ:

```
SELECT
 info.deviceServerInfo$owner,
```

```
 info.deviceServerInfo$name,
 info.deviceServerInfo$description,
 info.deviceServerInfo$blocked,
 info.status$servertods,
 info.status$dstoserver,
 info.CONTEXT_ID,
 info.PARENT_ID,
 info.RECORD_INDEX
FROM
 users.*.deviceservers.*:status:deviceServerInfo as info
```

Это позволит сделать описание сервера и результаты запроса редактируемыми. Остальные поля останутся доступными только для чтения, т.к. они заданы, как доступные только для чтения, в формате переменных **deviceServerInfo** ("Информация о сервере устройства") и **status** ("Статус сервера устройства"):

	Owner Name	Device Server Name	Description	Blocked	Bytes transferred from LinkServer to Device Server	Bytes transferred from Device Server to LinkServer
1	admin	agent	Auto-registered Device Server	☐	336756744	1112355301
2	admin	c1	Test DS	☐	75	0
3	admin	c2	<Not set>	☐	0	0
4	admin	c3	<Not set>	☐	204	0
5	admin	c4	T1000	☐	0	0
6	admin	c5	Auto-registered Device Server	☐	17190619317	33353387814
7	admin	c6	Auto-registered Device Server 2	☐	0	0
8	admin	c7	<Not set>	☐	0	0
9	admin	c8	<Not set>	☐	0	0
10	admin	ftest	Auto-registered Device Server	☐	4092428638	6743462742
11	admin	http	<Not set>	☐	43083	954012
12	admin	te	<Not set>	☑	0	0
13	trew	fds	<Not set>	☐	0	0

Если некоторые столбцы не редактируются из-за отсутствия полей обратной записи, это указывается в запросе отчет отладки 838:

	Message	Context	Exception	Stack Trace
1	Field has been made read-only since the query has no required metadata (CONTEXT_ID, PARENT_ID and RECORD_INDEX columns): INFO.DEVICESERVERINFO$OWNER	<Not set>	<Not set>	
2	Field has been made read-only since the query has no required metadata (CONTEXT_ID, PARENT_ID and RECORD_INDEX columns): INFO.DEVICESERVERINFO$NAME	<Not set>	<Not set>	
3	Field has been made read-only since the query has no required metadata (CONTEXT_ID, PARENT_ID and RECORD_INDEX columns): INFO.STATUS$SERVERTODS	<Not set>	<Not set>	
4	Field has been made read-only since the query has no required metadata (CONTEXT_ID, PARENT_ID and RECORD_INDEX columns): INFO.STATUS$DSTOSERVER	<Not set>	<Not set>	

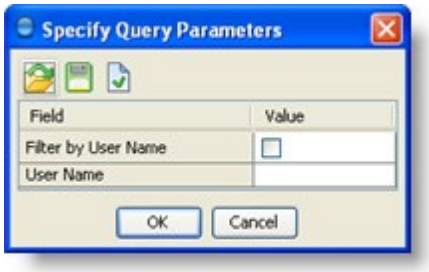
12.12.7 Параметризованные запросы

Параметризованные запросы используются для назначения некоторых параметров во время выполнения запроса. Исходные данные для процесса параметризации записаны в формате XML, а не на языке запросов AtomMind. Когда опция **Параметризован** включена в настройках запроса, процессор обработки запросов рассматривает текст в качестве параметризованных данных. Для получения более подробной информации см. Механизм параметризации 1444.

Далее приведен пример параметризованных данных с двумя полями в **Формате** (логический byusername and string username) и **Параметризованное выражение**:

```
SELECT * FROM users.*:childInfo <p enabled="{byusername}">WHERE childInfo$name LIKE '%
<e>{username}</e>%'</p>
```

Данный запрос будет запрашивать два параметра при выполнении:



Если пользователь указывает "Фильтровать по имени пользователя", то итоговый текст запроса будет следующим:

```
SELECT * FROM users.*:childInfo WHERE childInfo$name LIKE '%
text_entered_in_User_Name_field%'
```

т.е. будут отображаться только те пользователи, чьи имена включают данный текст.

Если "Фильтр по имени пользователя" не отмечен, то будут отображаться все пользователи, потому что параметризованный текст запроса будет следующим:

```
SELECT * FROM users.*:childInfo
```

Работая с параметризованными запросами, необходимо знать, что они будут обрабатываться как XML текст. Таким образом, использование таких символов как '<' and '>' для операций сравнения приведет к нарушению разметки XML. Необходимо избегать таких символов с < и > соответственно.

Пример 1: Выбор поддиапазона истории переменной

Этот параметризованный запрос позволяет операторам выбирать дату/время начала и окончания и показывает исторические значения переменной `batteryChargeCurrent`, которые были собраны за выбранный период времени.

```
SELECT * FROM
  utilities:variableHistory("users.admin.devices.site1", "batteryChargeCurrent",
"<e>{startDate}</e>", "<e>{endDate}</e>")
```

12.12.8 Производительность запросов

Время выполнения запроса и потребление ресурсов зависят от нескольких факторов:

- Количество и размер виртуальных таблиц, определенные контекстными ссылками[833] запроса. Например, в инсталляции 1000 устройств нагрузка процессора и памяти, обусловленные контекстной ссылкой `users.*.devices.*:temperature`, будет в 1000 раз больше, чем одна из ссылок `users.admin.devices.meter1:temperature`.
- Характер переменных[61] или функций[70] контекстов[41] сервера, на которые ссылается запрос. Для получения более подробной информации см. производительность переменных[70] и производительность функций[73].
- Сложность запроса (например, использование объединений, функций[860] и т.д.)

Отладка запроса[838] - это способ выяснить, какие этапы выполнения запроса замедлены.

Использование индексов

Подобно другим SQL запросам, производительность запросов AtomMind существенно зависит от наличия индексов.

Индексы создаются автоматически каждый раз при выполнении запроса для ключевых полей[51] переменных или результатов выполнения функций, являющихся исходными данными для запроса.

Обратитесь к разделу Ключевые поля[833] для получения дополнительной информации.

Функционирование в распределенной архитектуре

Запросы оптимизируются особым образом для работы в распределенной архитектуре[1332]. Запросы к данным контекстов, подключенных по распределённой архитектуре, выполняются параллельно при построении таблицы по контекстной ссылке[833]. Если многочисленные распределенные контексты расположены на разных серверах, все серверы будут параллельно готовить и отсылать свою часть данных.

Работа запросов в распределенной архитектуре имеет много общего с концепцией *MapReduce*:

- Удаленные серверы готовят данные частями, загружая, фильтруя, сортируя и агрегируя данные на своей стороне
- Изначальный запрос, работающий на сервере верхнего уровня (потребителе), объединяет входящие части данных и осуществляет дополнительную сортировку, фильтрацию и агрегирование
- Более того, запрос может объединять итоговый набор данных с другими наборами данных, поступающими с локального сервера или через распределённую архитектуру

12.12.9 Примеры запросов

Данная статья доступно объясняет некоторые реальные примеры запросов.

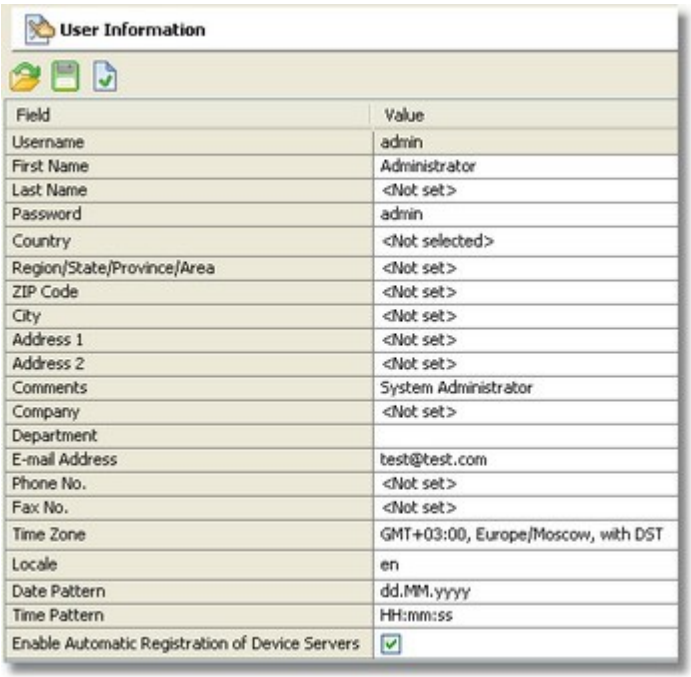
Пример 1: Просмотр/редактирование учетных записей пользователей

Просмотр или изменение различных настроек учетных записей[478] является общей административной задачей. Язык запроса может помочь администраторам AtomMind Server выполнить массовые изменения нескольких учетных записей пользователей или посмотреть определенные настройки множества учетных записей в одной упорядоченной таблице.

ВЫБОР ИНФОРМАЦИИ ОДНОГО ПОЛЬЗОВАТЕЛЯ

Для доступа к основным настройкам учетной записи пользователя мы используем переменную "userInfo" ("Информация о пользователе"), заданную в контексте Пользователь[1608]. Значение данной переменной имеет одну запись с различными полями, в которых записаны имя/фамилия пользователя, страна и т.д.

Ниже представлен типичный вид переменной информации о пользователе:



Field	Value
Username	admin
First Name	Administrator
Last Name	<Not set>
Password	admin
Country	<Not selected>
Region/State/Province/Area	<Not set>
ZIP Code	<Not set>
City	<Not set>
Address 1	<Not set>
Address 2	<Not set>
Comments	System Administrator
Company	<Not set>
Department	
E-mail Address	test@test.com
Phone No.	<Not set>
Fax No.	<Not set>
Time Zone	GMT+03:00, Europe/Moscow, with DST
Locale	en
Date Pattern	dd.MM.yyyy
Time Pattern	HH:mm:ss
Enable Automatic Registration of Device Servers	☑

Теперь мы используем один из самых простых возможных запросов, чтобы получить значение данной переменной только из одного контекста.

Текст запроса:

```
SELECT * FROM users.admin:childInfo
```

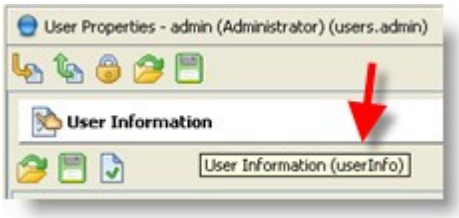
Результатом данного запроса будет являться таблица, отображающая значение переменной **childInfo**, заданной в контексте **users.admin**.

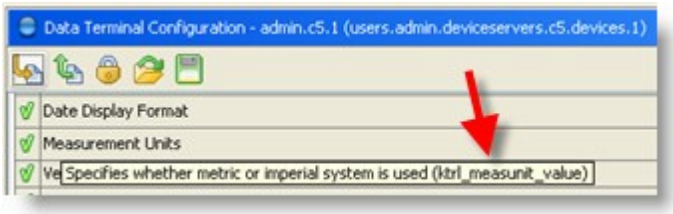
При просмотре или изменении свойств некого системного объекта или устройства его имя контекста обычно показывается в заголовке окна:

Данный запрос состоит из двух разделов: ВЫБРАТЬ и ОТ. Раздел ВЫБРАТЬ предлагает нам выбрать все поля ("*"). Раздел ОТ содержит одну контекстную ссылку 833 ("`users.admin:childInfo`"), которая используется для построения таблицы, в которой будет выполняться запрос. Содержание и формат данной таблицы в точности совпадут с переменной **childInfo**.

Все поля в результате запроса будут доступны для редактирования 839, кроме тех полей, которые определены как доступные только для чтения в формате переменной **childInfo**.

При просмотре или изменении свойств какого-либо контекста (такого как контекст "Пользователь") вы увидите *описания* переменных, а не их *имена*. Однако в тесте запроса нам нужно использовать *имена* переменных: они обычно показываются во всплывающих окнах при наведении мыши на описание переменных:





Результат запроса:

Query Results

Field	Value
Username	admin
First Name	Administrator
Last Name	<Not set>
Password	admin
Country	<Not selected>
Region/State/Province/Area	<Not set>
ZIP Code	<Not set>
City	<Not set>
Address 1	<Not set>
Address 2	<Not set>
Comments	System Administrator
Company	<Not set>
Department	
E-mail Address	test@test.com
Phone No.	<Not set>
Fax No.	<Not set>
Time Zone	GMT+03:00, Europe/Moscow, with DST
Locale	en
Date Pattern	dd.MM.yyyy
Time Pattern	HH:mm:ss
Enable Automatic Registration of Device Servers	☑

ВЫБОР ИНФОРМАЦИИ ВСЕХ ПОЛЬЗОВАТЕЛЕЙ

Теперь мы будем выбирать свойства **всех** пользователей в системе:

Текст запроса:

```
SELECT
    *
FROM
```

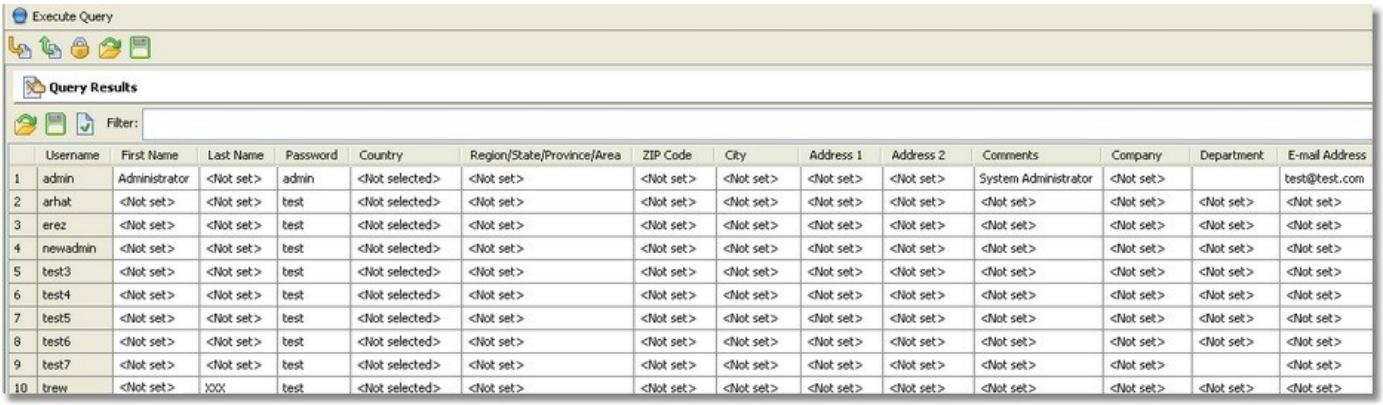
```
users.*:childInfo
```

Контекстная ссылка (**users.*:childInfo**) в тексте данного запроса по сути является маской 44 контекстов. Во время выполнения запроса данная маска преобразуется в список всех контекстов пользователей, доступных пользователю, выполняющему запрос. Для получения более подробной информации обратитесь к главе Контекстные ссылки 833.

Результатом данного запроса станет таблица со значениями переменной **childInfo** каждой учетной записи, доступной пользователю, выполняющему запрос. Возможно, она будет содержать множество записей, по одной на каждую учетную запись. Результат запроса также доступен для редактирования.

Данный запрос встроен в дистрибутив AtomMind Server. Он называется **Все пользователи**.

Результат запроса:

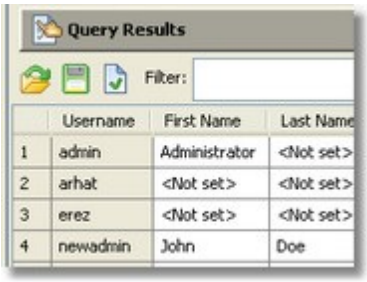

Execute Query

Query Results

Filter:

	Username	First Name	Last Name	Password	Country	Region/State/Province/Area	ZIP Code	City	Address 1	Address 2	Comments	Company	Department	E-mail Address
1	admin	Administrator	<Not set>	admin	<Not selected>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	System Administrator	<Not set>		test@test.com
2	arhat	<Not set>	<Not set>	test	<Not selected>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>
3	erez	<Not set>	<Not set>	test	<Not selected>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>
4	newadmin	<Not set>	<Not set>	test	<Not selected>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>
5	test3	<Not set>	<Not set>	test	<Not selected>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>
6	test4	<Not set>	<Not set>	test	<Not selected>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>
7	test5	<Not set>	<Not set>	test	<Not selected>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>
8	test6	<Not set>	<Not set>	test	<Not selected>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>
9	test7	<Not set>	<Not set>	test	<Not selected>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>		<Not set>
10	trew	<Not set>	XXX	test	<Not selected>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>

Например, вы изменяете имя и фамилию некого пользователя и сохраняете изменения:


Query Results

Filter:

	Username	First Name	Last Name
1	admin	Administrator	<Not set>
2	arhat	<Not set>	<Not set>
3	erez	<Not set>	<Not set>
4	newadmin	John	Doe

Изменения сразу же отображаются во всей системе:

СОРТИРОВКА СПИСКА ПОЛЬЗОВАТЕЛЕЙ

Теперь мы произведем сортировку таблицы.

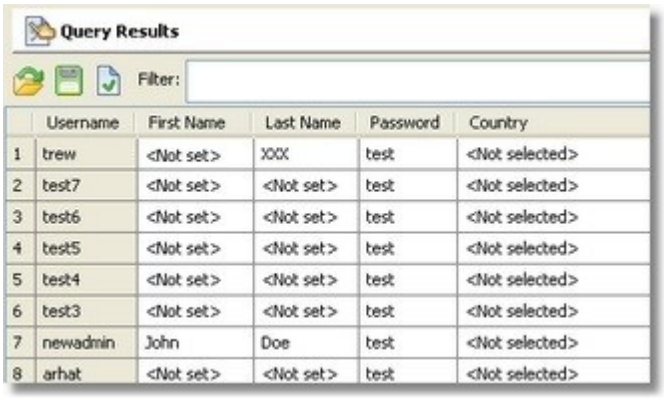
Текст запроса:

```
SELECT
  *
FROM
  users.*:childInfo order by childInfo$name desc
```

Данный запрос выдает таблицу, схожую с указанной выше, но строки в данной таблице упорядочиваются по значению поля name в формате переменной **childInfo** в нисходящем порядке. Запрос содержит состоящую из двух частей ссылку на поле 836 (**childInfo$name**), используемую для ссылки на поле таблицы, построенной при разрешении контекстной ссылки **users.*:childInfo**.

При изучении значения переменной "childInfo" и других переменных AtomMind Client в пользовательском интерфейсе AtomMind Server *описания* полей отображаются в заголовке таблицы, а не в *именах* полей. Однако мы должны использовать *имя* поля, чтобы ссылаться на него в тексте запроса. Имена полей обычно отображаются во всплывающих окнах, которые появляются при наведении мыши на заголовок поля:

Результат запроса:



	Username	First Name	Last Name	Password	Country
1	trew	<Not set>	XXX	test	<Not selected>
2	test7	<Not set>	<Not set>	test	<Not selected>
3	test6	<Not set>	<Not set>	test	<Not selected>
4	test5	<Not set>	<Not set>	test	<Not selected>
5	test4	<Not set>	<Not set>	test	<Not selected>
6	test3	<Not set>	<Not set>	test	<Not selected>
7	newadmin	John	Doe	test	<Not selected>
8	arhat	<Not set>	<Not set>	test	<Not selected>

ИСПОЛЬЗОВАНИЕ АЛЬТЕРНАТИВНЫХ ИМЕН ТАБЛИЦ

Мы можем редактировать запрос и задать альтернативное имя **info** для нашей контекстной ссылки, а затем использовать трехкомпонентную контекстную ссылку, которая будет включать данное альтернативное имя:

```
SELECT
  *
FROM
  users.*:childInfo as info
ORDER BY
  info.childInfo$name desc
```

Результат запроса будет такой же, как и предыдущий.

ОГРАНИЧЕНИЕ ВЫБОРКИ ПОЛЕЙ

Теперь выберем ограниченное количество полей при помощи последующего изменения того же запроса.

Текст запроса:

```
SELECT
  childInfo$name, childInfo$firstname, childInfo$lastname
FROM
  users.*:childInfo
ORDER BY
  childInfo$name desc
```

Данный запрос содержит несколько ссылок на поля в разделе ВЫБРАТЬ. Так как поля задаются в явной форме, результат запроса будет нередактируемым, если мы не добавим поля обратной записи[840]. Добавлять их мы будем в следующих примерах.

Результат данного запроса будет содержать только три столбца.

Результат запроса:

	Username	First Name	Last Name
1	trew	<Not set>	XXX
2	test7	<Not set>	<Not set>
3	test6	<Not set>	<Not set>
4	test5	<Not set>	<Not set>
5	test4	<Not set>	<Not set>
6	test3	<Not set>	<Not set>
7	newadmin	John	Doe
8	arhat	<Not set>	<Not set>
9	admin	Administrator	<Not set>

ДОБАВЛЕНИЕ ВЫРАЖЕНИЙ

В данном случае мы будем использовать выражение языка запроса для отображения имен и фамилий пользователей в одном столбце, чтобы упростить процесс обработки данных. Мы также добавим новый столбец в запрос - Country (страна пользователя).

Текст запроса:

```
SELECT
  childInfo$name,
  CASEWHEN((childInfo$firstname IS NULL), '', childInfo$firstname)  || ' ' ||
CASEWHEN((childInfo$lastname IS NULL), '', childInfo$lastname),
  childInfo$country
FROM
  users.*:childInfo
ORDER BY
  childInfo$name desc
```

Данный запрос выбирает три поля в исходной таблице. Первое и третье содержат имя пользователя и страну соответственно. Значение второго поля рассчитывается при помощи *выражения*. Данное выражение получает имя пользователя, добавляет в конце место и заполняет его фамилией пользователя. Двойные вертикальные черты (||) являются стандартным оператором SQL, используемым для соединения строк. Имя и фамилия изменяются при помощи встроенной функции **CASEWHEN**, чтобы отображалась пустая строка в том случае, если значение имени или фамилии является NULL ("<*Не установлено*>"). Имейте в виду, что второй столбец формируется автоматически.

Результат запроса:

	Username	column_2	Country
1	trew		<Not selected>
2	test7		<Not selected>
3	test6		<Not selected>
4	test5		<Not selected>
5	test4		<Not selected>
6	test3		<Not selected>
7	newadmin	John Doe	<Not selected>
8	arhat		<Not selected>
9	admin	Administrator	<Not selected>

ИСПОЛЬЗОВАНИЕ АЛЬТЕРНАТИВНЫХ ИМЕН ПОЛЕЙ

Теперь мы зададим название для второго столбца, чтобы сделать результат более удобным для чтения.

Текст запроса:

```
SELECT
  childInfo$name,
  CASEWHEN((childInfo$firstname IS NULL), '', childInfo$firstname)  || ' ' ||
CASEWHEN((childInfo$lastname IS NULL), '', childInfo$lastname) as name,
  childInfo$country
FROM
  users.*:childInfo
ORDER BY
  childInfo$name desc
```

Второй столбец в результате запроса отныне называется "name" (имя).

Результат запроса:

	Username	name	Country
1	trew		<Not selected>
2	test7		<Not selected>
3	test6		<Not selected>
4	test5		<Not selected>
5	test4		<Not selected>
6	test3		<Not selected>
7	newadmin	John Doe	<Not selected>
8	arhat		<Not selected>
9	admin	Administrator	<Not selected>

ВОЗМОЖНОСТЬ РЕДАКТИРОВАНИЯ РЕЗУЛЬТАТА

Чтобы было возможно изменить страну пользователя, добавьте поле обратной записи[840], таким образом делая отчет редактируемым. Для этого нам нужно назначить альтернативное имя для контекстной ссылки ("table alias"), потому как поля обратной записи (CONTEXT_ID, PARENT_ID и RECORD_INDEX) должны быть заданы для определенной контекстной ссылки. Чтобы определить, какая запись должна быть записана обратно, необходимо использовать альтернативное имя. Внимательно прочтите приведенный ниже пример, чтобы полностью в этом разобраться.

Текст запроса:

```
SELECT
  info.childInfo$name,
  CASEWHEN((info.childInfo$firstname IS NULL), '', info.childInfo$firstname)  || ' '
|| CASEWHEN((info.childInfo$lastname IS NULL), '',   info.childInfo$lastname) as name,
  info.childInfo$country,
  info.CONTEXT_ID,
  info.PARENT_ID,
  info.RECORD_INDEX
FROM
  users.*:childInfo as info
ORDER BY
  info.childInfo$name desc
```

В результате данного запроса столбец "country" будет редактируемым. Столбец "username" останется доступным для чтения, потому что он так определен в исходной переменной ("userInfo"). Столбец "name" доступен только для чтения, т.к. он рассчитывается при помощи выражения.

Результат запроса:

	Username	name	Country
1	trew		<Not selected>
2	test7		<Not selected>
3	test6		Antarctica
4	test5		<Not selected>
5	test4		<Not selected>
6	test3		<Not selected>
7	newadmin	John Doe	<Not selected>
8	arhat		<Not selected>
9	admin	Administrator	<Not selected>

ФИЛЬТРАЦИЯ СПИСКА ПОЛЬЗОВАТЕЛЕЙ

Теперь добавим к запросу несколько правил фильтра. Для примера выберем только тех пользователей, чьи имена пользователей (usernames) не содержат слово "test".

Текст запроса:

```
SELECT
  info.childInfo$name,
```

```
  CASEWHEN((info.childInfo$firstname IS NULL), '', info.childInfo$firstname)  || ' '
|| CASEWHEN((info.childInfo$lastname IS NULL), '',   info.childInfo$lastname) as name,
  info.childInfo$country,
  info.CONTEXT_ID,
  info.PARENT_ID,
  info.RECORD_INDEX
FROM
  users.*:childInfo as info
WHERE
  info.childInfo$name NOT LIKE '%test%'
ORDER BY
  info.childInfo$name desc
```

Данный запрос содержит дополнительный раздел ГДЕ, определяющий, какие записи должны попадать в результат.

Результат запроса:

	Username	name	Country
1	trew		<Not selected>
2	newadmin	John Doe	<Not selected>
3	arhat		<Not selected>
4	admin	Administrator	<Not selected>

ОГРАНИЧЕНИЕ КОЛИЧЕСТВА СТРОК В РЕЗУЛЬТАТЕ ЗАПРОСА

Последняя строка следующего примера включает раздел ОГРАНИЧИТЬ, используемый для ограничения количества строк в результате запроса.

Текст запроса:

```
SELECT
  info.childInfo$name,
  CASEWHEN((info.childInfo$firstname IS NULL), '', info.childInfo$firstname)  || ' '
|| CASEWHEN((info.childInfo$lastname IS NULL), '',   info.childInfo$lastname) as name,
  info.childInfo$country,
  info.CONTEXT_ID,
  info.PARENT_ID,
  info.RECORD_INDEX
FROM
  users.*:childInfo as info
WHERE
  info.childInfo$name NOT LIKE '%test%'
ORDER BY
  info.childInfo$name desc
LIMIT 2 OFFSET 1
```

Результат данного запроса содержит только две (`LIMIT 2`) записи (2-ую и 3-ю) из результата предыдущего запроса. Здесь мы используем `OFFSET 1`. Это означает, что мы получим 2 записи (`LIMIT 2`), начиная со второй записи (OFFSET 1). Оператор OFFSET дает команду AtomMind брать записи, начиная со второй (номер первой записи - 0).

Результат запроса:

	Username	name	Country
1	newadmin	John Doe	<Not selected>
2	arhat		<Not selected>

Пример 2: Просмотр/изменение учетных записей Device Server

Данный пример похож на выборку основных свойств всех пользователей, описанную в предыдущем примере, однако, здесь мы используем переменную **deviceServerInfo** (Информация устройства), заданную в контексте"deviceServer"[2102].

Текст запроса:

```
SELECT
  *
FROM
  users.*.deviceservers.*:deviceServerInfo
```

Текст данного запроса включает одну контекстную ссылку (**users.*.deviceservers.*:deviceServerInfo**) с маской контекста, которая распространяется на все контексты deviceServer, доступные при наличии прав доступа[477] у пользователя, выполняющего запрос. Результаты данного запроса редактируемы, за исключением полей, которые определены как доступные только для чтения в переменной **deviceServerInfo**.

Данный запрос встроен в дистрибутив AtomMind Server. Он называется **Все Device Servers**.

Результат запроса:

	Owner Name	Device Server Name	Password	Description	Register in DNS	Blocked	Inban...	Data Transfer/Device Plugin	Time Zone
1	admin	agent		Auto-registered Device Server	☐	☐	☐	Agent	GMT+03:00, Eu
2	admin	c1		Test DS	☐	☐	☐	Dynamic DNS	GMT+03:00, Eu
3	admin	c2		<Not set>	☐	☐	☐	Dynamic DNS	GMT+03:00, Eu
4	admin	c3		<Not set>	☐	☐	☐	Transparent data routing (Link Service)	GMT+03:00, Eu
5	admin	c4	test	T1000	☐	☐	☐	Dynamic DNS	GMT+03:00, Eu
6	admin	c5		Auto-registered Device Server	☐	☐	☐	Device based on Standard Communication Protocol	GMT+03:00, Eu
7	admin	c6		Auto-registe	☐	☐	☐	Transparent data routing (Link Service)	GMT+03:00, Eu
8	admin	c7		<Not set>	☐	☐	☐	Device based on Standard Communication Protocol	GMT+03:00, Eu
9	admin	c8	c8	<Not set>	☐	☐	☐	Device based on Standard Communication Protocol	GMT+03:00, Eu
10	admin	ftest		Auto-registered Device Server	☐	☐	☐	Device based on Standard Communication Protocol	GMT+03:00, Eu
11	admin	http		<Not set>	☐	☐	☐	HTTP Proxy	GMT+03:00, Eu
12	admin	te	TE	<Not set>	☐	☑	☐	Dynamic DNS	GMT+03:00, Eu
13	trew	fds	fds	<Not set>	☐	☐	☐	Device based on Standard Communication Protocol	GMT+03:00, Eu

Value: Auto-registered Device Server
Type: String

Пример 3: Просмотр статистики трафика Device Server

Данный пример показывает применение контекстных ссылок, включающих в себя множество переменных/полей.

Текст запроса:

```
SELECT
 info.deviceServerInfo$owner,
 info.deviceServerInfo$name,
 info.status$servertods,
 info.status$dstoserver
FROM
 users.*.deviceservers.*:status:deviceServerInfo as info
ORDER BY
 info.deviceServerInfo$owner,
 info.deviceServerInfo$name
```

Контекстная ссылка в данном запросе ("**users.*.deviceservers.*:status:deviceServerInfo**") ссылается на две переменные, определенные в контексте "deviceServer"[2102]: на указанную выше "deviceServerInfo" и "status" ("Статус устройства"), которая содержит реальные значения относительно работы Device Server. Значения обеих переменных включают только одну запись, поэтому итоговая таблица, которая сформируется после

преобразования этой контекстной ссылки, будет содержать все поля, появляющиеся в в значениях обеих переменных, и одну запись на каждое Device Server доступное пользователю, выполняющему запрос.

Из этой таблицы мы выбираем четыре столбца: два, которые появляются в значении **deviceServerInfo** (Владелец и имя Device Server) и два из **status** (Входящий и исходящий трафик). Таблица упорядочивается по двум полям: сначала по владельцу Device Server, затем по имени Device Server (потому что некоторые Device Server могут принадлежать одному пользователю).

Данный запрос встроен в дистрибутив AtomMind Server. Он называется **Статистика трафика Device Server.**

Результат запроса:

	Owner Name	Device Server Name	Bytes transferred from LinkServer to Device Server	Bytes transferred from Device Server to LinkServer
1	admin	agent	336756744	1112355301
2	admin	c1	75	0
3	admin	c2	0	0
4	admin	c3	204	0
5	admin	c4	0	0
6	admin	c5	17190656536	33353459489
7	admin	c6	0	0
8	admin	c7	0	0
9	admin	c8	0	0
10	admin	ftest	4092428638	6743462742
11	admin	http	43083	954012
12	admin	te	0	0
13	trew	fds	0	0

Пример 4: Вызов серверов устройств

Данный пример показывает, как функции контекста могут быть использованы в данном запросе.

Текст запроса:

```
SELECT
  *
FROM
  external_device_servers.*:buzz()
```

Данный запрос очень интересен, потому что у него нет выхода, однако, он дает побочный эффект на сервере. Он содержит одну контекстную переменную ("**external_device_servers.*:buzz()**"), которая включает в себя функцию **buzz()** ("Вызвать Device Server"), заданную в контексте "deviceServer"2112. Данная функция вызывается без параметров, что обозначено пустыми скобками. Функция "Вызвать устройство" заставляет Device Server мерцать диодами для его физической идентификации. Таким образом, данный запрос помогает определить местонахождение всех устройств в сегменте локальной сети. Важно запустить действие **Обнаружить <%DS%** до выполнения запроса. Данное действие позволяет AtomMind Server найти все локальные Device Server, отправляя широковещательные сетевые запросы. Для получения более подробной информации обратитесь к разделу Внешние серверы устройств2096.

Если обнаружение не было выполнено, у контекста Внешние серверы устройств2096 не будет дочерних, маска контекста **external_device_servers.*** не преобразуется ни в какой контекст, и при выполнении запроса не будут совершаться действия.

Выход функции **buzz()** не содержит полей, поэтому данный запрос не возвращает никаких данных. Но его выполнение приводит к тому, что все внешние Device Server начинают "мигать", таким образом становится проще их обнаружить.

Данный запрос встроен в дистрибутив AtomMind Server. Он называется "**Вызвать все внешние Device Servers**".

Пример 5: Расчет общего трафика всех серверов устройств

Данный пример показывает, как можно использовать функцию агрегации.

Текст запроса:

```
SELECT
  SUM(status$servertods) as server_to_ds,
  SUM(status$dstoserver) as ds_to_server
FROM
  users.*.deviceservers.*:status
```

В одном из предыдущих примеров мы упоминали переменную **status** ("Статус Device Server"), заданную в контексте "deviceServer"2102, которая содержит статистику трафика между AtomMind Server и Device Server. Данный запрос рассчитывает, сколько байт было отправлено на видимые в данный момент Device Server и сколько было принято от них. Результат запроса содержит одну запись, потому что используется функция агрегации SUM.

Результат запроса:

Field	Value
server_to_ds	21 619 886 080
ds_to_server	41 210 232 832

Поскольку результат запроса содержит только одну строку, он отображается в виде "вертикального" шаблона из двух столбцов, где имена полей записаны в первом столбце, а значения полей -- во втором.

Данный пример можно изменить для использования других функций агрегации:
- AVG поможет рассчитать средний трафик Device Server
- MIN и MAX могут применяться для расчета минимального и максимального трафика соответственно
- COUNT может использоваться для обнаружения нескольких Device Server (хотя это не самый лучший способ)

Пример 6: Выполнение запроса по нескольким таблицам

Данный пример иллюстрирует выполнение запросов по нескольким таблицам (построенным из нескольких контекстных ссылок).

Текст запроса:

```
SELECT
  d.deviceServerInfo$owner || '.' || d.deviceServerInfo$name as device_server,
  d.deviceServerInfo$blocked,
  u.userInfo$city,
  u.userInfo$country
FROM
  users.*:childInfo as u, users.*.deviceservers.*:deviceServerInfo as d
WHERE
  u.childInfo$name = d.deviceServerInfo$owner
```

Данный запрос имеет две контекстные ссылки в разделе ОТ. Первая используется для построения таблицы, содержащей все базовые настройки всех пользователей, вторая - для формирования таблицы базовых настроек всех Device Server. Эти таблицы затем объединяются, основываясь на формуле в разделе ГДЕ ("**u.childInfo$name = d.deviceServerInfo$owner**"). Результат запроса имеет четыре поля: первое содержит полное имя устройства в виде "Имя владельца"."Имя Device Server". Второе отображает статус Device Server "Блокировано". Два остальных столбца показывают город и страну пользователя, к которому относятся Device Server (которые, вероятно, могут оказаться городом и страной, где расположено Device Server).

Результат запроса:

	device_server	Blocked	City	Country
1	admin.agent	No	Bobruisk	Belarus
2	admin.c1	No	Bobruisk	Belarus
3	admin.c2	No	Bobruisk	Belarus
4	admin.c3	No	Bobruisk	Belarus
5	admin.c4	No	Bobruisk	Belarus
6	admin.c5	No	Bobruisk	Belarus
7	admin.c6	No	Bobruisk	Belarus
8	admin.c7	No	Bobruisk	Belarus
9	admin.c8	No	Bobruisk	Belarus
10	admin.ftest	No	Bobruisk	Belarus
11	admin.http	No	Bobruisk	Belarus
12	admin.te	Yes	Bobruisk	Belarus
13	arhat.DS1	No	<Not set>	<Not selected>
14	test5.ds1	No	Penguin City	Antarctica
15	test5.ds2	No	Penguin City	Antarctica
16	test5.ds3	No	Penguin City	Antarctica
17	trew.fds	No	<Not set>	<Not selected>

Пример 7: Использование ОБЪЕДИНИТЬ

Данный пример показывает, как использовать ОБЪЕДИНИТЬ SQL и скрытые поля 834 для сочетания данных из двух таблиц, образованных из двух разных контекстных ссылок 833.

Когда AtomMind используется для сетевого управления, AtomMind Server работает с устройствами по протоколу SNMP. Каждое устройство SNMP обладает так называемой таблицей интерфейса (переменная `ifTable` контекста устройства), в которой содержится статистика сетевых интерфейсов устройства. Приведенный ниже запрос находит все сетевые интерфейсы всех устройств SNMP, чей статус равен "*down*" (т.е. поле `ifOperStatus` равно 2).

Запрос добавляет описание контекста устройства (т.е. имя устройства) к описанию сетевого интерфейса, чтобы в дальнейшем отображать их в одной таблице.

```
SELECT
  info.info$description, ift.ifTable$ifDescr, ift.ifTable$ifOperStatus
FROM
  users.*.devices.*:ifTable as ift LEFT OUTER JOIN users.*.devices.*:info as info ON
ift$CONTEXT_ID = info$CONTEXT_ID
WHERE
  ift.ifTable$ifOperStatus = 2
```

Результат запроса:

	Description	ifDescr	ifOperStatus
1	admin.ant (SNMP)	Marvell Yukon 88E8038 PCI-E Fast Ethernet Controller - Минипорт планировщика пакет...	down
2	Server (SNMP)	sit0	down
3	Mail Server (SNMP)	sit0	down
4	This PC (SNMP)	Intel(R) 82566MM Gigabit Network Connection	down
5	This PC (SNMP)		down
6	This PC (SNMP)	Teredo Tunneling Pseudo-Interface	down
7	This PC (SNMP)	isatap.{46816747-3137-44BE-86E7-B92384F28536}	down
8	This PC (SNMP)	isatap.{CB8B0BFE-FDFF-418B-95CC-D7465D0923BD}	down
9	This PC (SNMP)	isatap.{ED1EBC11-A95A-4C43-BBE9-5C1D3FCC5532}	down
10	This PC (SNMP)	Intel(R) 82566MM Gigabit Network Connection-QoS Packet Scheduler-0000	down
11	Gateway Router (SNMP)	sit0	down
12	Web Server (SNMP)	sit0	down

Дополнительные примеры

ПОЛУЧЕНИЕ СОБЫТИЯ ВХОДА ПОЛЬЗОВАТЕЛЯ В СИСТЕМУ

Данный запрос вызывает функцию get 1516() и фильтрует полученную таблицу по имени пользователя (username).

```
SELECT
 *
FROM
 events:get("users", "login") as logins
WHERE
 logins.get$username = 'operator'
```

Имейте в виду, что те же результаты можно получить, добавив выражение фильтра в вызов функции get():

```
SELECT
 *
FROM
 events:get("users", "login", "{username} == 'operator'") as logins
```

ОБНАРУЖЕНИЕ СЕТЕВЫХ УСТРОЙСТВ С МИНИМАЛЬНЫМ ВРЕМЕНЕМ ОТКЛИКА

Данный запрос обнаруживает все сетевые устройства, время отклика которых превышает 500 мсек и отображает их имена вместе с временем отклика.

```
SELECT
  info.info$description AS device,
```

```
  ping.ping$averageTime AS average_round_trip_time
FROM
  users.*.devices.*:info AS info
RIGHT OUTER JOIN
  users.*.devices.*:ping AS ping
ON
  ping$CONTEXT_ID = info$CONTEXT_ID
WHERE
  ping.ping$averageTime > 500
ORDER BY
  ping.ping$averageTime DESC
```

ОБНАРУЖЕНИЕ УСТРОЙСТВ ВЫСОКОЙ ЗАГРУЗКИ ЦП

Данный запрос обнаруживает все сетевые устройства SNMP, у которых средняя загрузка всех процессоров превышает 90%. Он превосходно работает с мультипроцессорными устройствами.

```
SELECT
  info.info$description AS device,
  avg(processors.hrProcessorTable$hrProcessorLoad) AS processor_utilization_percentage
FROM
  users.*.devices.*:hrProcessorTable AS processors
LEFT OUTER JOIN
  users.*.devices.*:info as info
ON
  processors$CONTEXT_ID = info$CONTEXT_ID
GROUP BY
  device
HAVING
  avg(processors.hrProcessorTable$hrProcessorLoad) > 90
ORDER BY
  processor_utilization_percentage DESC
```

ГРУППИРОВАНИЕ УСТРОЙСТВ SNMP ПО ТИПУ

Данный запрос рассчитывает количество устройств SNMP каждого типа (тип определяется переменной **sysObjectID**, предоставляемой самим устройством) и выводит результат в таблицу. Имейте в виду, что описания каждого типа устройства берутся из общей таблицы 2178, которая называется **deviceTypes**.

```
SELECT
  coalesce(types.value$description, snmp.sysObjectID$sysObjectID) AS device_type,
  COUNT(*) AS device_count
FROM
  users.*.devices.*:status:sysObjectID AS snmp
LEFT JOIN
  common.deviceTypes:value AS types
ON
  snmp.sysObjectID$sysObjectID = types.value$type
WHERE
```

```
  snmp.status$driver = 'com.tibbo.linkserver.plugin.device.snmp'
GROUP BY
  device_type
ORDER BY
  device_count DESC
```

ПОЛУЧЕНИЕ СТАТИСТИКИ НАСТРОЕК УСТРОЙСТВА

Данный запрос использует функцию статистика 1616, заданную в контексте Утилиты 1613 для извлечения последних значений канала статистики 718. Канал настроен для подсчета ошибок интерфейса сетевых устройств, которые возникают каждый час/день/месяц.

Запрос группирует таблицы интерфейсов всех устройств в одну и использует ее как основу (`FROM users.*.devices.*:ifTable AS interfaces`). Затем он слева присоединяет данные последнего часа из двух каналов статистики -- "ifInOctets" и "ifOutOctets" (`LEFT OUTER JOIN utilities:statistics("users.*.devices.*", "ifInErrors", null, "hour") AS in_errors`). И, наконец, слева таблицы присоединяется информация об устройстве. Запрос также осуществляет фильтрацию, сортировку и ограничение количества рядов.

```
SELECT
 info.info$description AS device,
 interfaces.ifTable$ifDescr AS interface,
 in_errors.statistics$average * 3600 AS incoming_errors,
 out_errors.statistics$average * 3600 AS outgoing_errors
FROM
 users.*.devices.*:ifTable AS interfaces
LEFT OUTER JOIN
 utilities:statistics("users.*.devices.*", "ifInErrors", null, "hour") AS in_errors
ON
 interfaces.ifTable$ifIndex = in_errors.statistics$key
 AND
 interfaces$CONTEXT_ID = in_errors.statistics$context
LEFT OUTER JOIN
 utilities:statistics("users.*.devices.*", "ifOutErrors", null, "hour") AS out_errors
ON
 interfaces.ifTable$ifIndex = out_errors.statistics$key
 AND
 interfaces$CONTEXT_ID = out_errors.statistics$context
LEFT OUTER JOIN
 users.*.devices.*:info AS info
ON
 info$CONTEXT_ID = interfaces$CONTEXT_ID
WHERE
  length(interfaces.ifTable$ifDescr) > 1
ORDER BY
  incoming_errors + outgoing_errors DESC
LIMIT
  10
```

СОСТАВЛЕНИЕ СПИСКА СОТРУДНИКОВ ПО ОТДЕЛАМ

В AtomMind Time and Attendance система обрабатывает данные посещения для *держателей карт,* образуя сложную иерархию, которая включает организации, отделы и, возможно, подразделения. Поскольку путь контекста каждого сотрудника начинается с пути контекста подразделения, к которому он относится, мы можем создать запрос, который создал бы список сотрудников по подразделениям:

```
SELECT
 divisions.childInfo$name,
 cardholders.childInfo$name
FROM
 organizations.Organization1.divisions.*.cardholders.*:childInfo as cardholders,
 organizations.Organization1.divisions.*:childInfo as divisions
WHERE
 substring(cardholders.CONTEXT_ID, 1, length(divisions.CONTEXT_ID)) =
divisions.CONTEXT_ID
```

ПОЛУЧЕНИЕ ТРЕВОГ, АКТИВНЫХ ДЛЯ МНОЖЕСТВА УСТРОЙСТВ

В приведенном ниже примере каждое устройство имеет табличное пользовательское свойство 2177 **custom_childContext**, которое приводит список всех устройств, *зависимых* от него. Это свойство имеет поле **contextPath**, которое содержит пути зависимых устройств.

Запрос проверяет триггеры, представленные в таблицах `activeInstances` всех тревог. Выбираются только триггеры, чье поле **источник** равно контекстному пути выбранного устройства (`i.CONTEXT_ID`) или представлено в таблице **custom_childContext** выбранного устройства.

Таким образом, запрос возвращает триггеры тревог, активные как для данного устройства (**dev1**), так и для его зависимых устройств.

```
SELECT
 *
FROM
 users.*.alerts.*:activeInstances AS at
WHERE
 at.activeInstances$source IN
 (
 SELECT
  i.CONTEXT_ID
 FROM users.admin.devices.dev1:info AS i
 UNION SELECT
  c.custom_childContexts$contextPath
 FROM users.admin.devices.dev1:custom_childContexts AS c
 )
```

ОБЪЕДИНЕНИЕ ИСТОРИЧЕСКИХ ЗНАЧЕНИЙ МНОЖЕСТВА ПЕРЕМЕННЫХ

Функция variableHistory 1613 может загружать исторические значения любой переменной контекста. Это подходит для отображения этих значений в отчете, на панели инструментов и т.д. Но что если вы хотите показать множество переменных в одной таблице? Например, вы измеряете температуру и историю и хотите, чтобы измерения были представлены в виде единой таблицы. Вот решение.

Приведенный ниже запрос загружает десятидневную историю двух отдельных переменных в двух виртуальных таблицах и объединяет эти таблицы, соединяя вместе значения с одинаковыми временными метками с точностью до одной секунды.

```
SELECT
  sineTimeString, sineTime,  sineValue, triangleTime, triangleValue
FROM
  (
```

```
  SELECT
    TO_CHAR(sine.variableHistory$vUpdateTime, 'DD.MM.YYYY H:MI:SS') as sineTimeString,
sine.variableHistory$vUpdateTime as sineTime, sine.variableHistory$value as sineValue
  FROM
    utilities:variableHistory("users.admin.devices.demoVirtualDevice", "sine",
'dateAdd(now(), -10, \"m\")') as sine
  )
JOIN
  (
  SELECT
    TO_CHAR(triangle.variableHistory$vUpdateTime, 'DD.MM.YYYY H:MI:SS') as
triangleTimeString, triangle.variableHistory$vUpdateTime as triangleTime,
triangle.variableHistory$value as triangleValue
  FROM
    utilities:variableHistory("users.admin.devices.demoVirtualDevice", "triangle",
'dateAdd(now(), -10, \"m\")') as triangle
  )
ON
sineTimeString = triangleTimeString
```

ОБЪЕДИНЕНИЕ РЕЗУЛЬТАТОВ НЕСКОЛЬКИХ ПОДЗАПРОСОВ

Приведенный ниже запрос ссылается на три других запроса, представляя список их переменных `data` в разделе ОТ основных и вложенных утверждений ВЫБРАТЬ. Результаты трех запросов соединяются вместе, используя операторы ОБЪЕДИНИТЬ.

```
SELECT
  hpux.data$icon AS icon,
  hpux.data$device AS device,
  hpux.data$context AS context,
  hpux.data$cpu AS cpu,
  hpux.data$max_threshold as max_threshold,
  hpux.data$min_threshold as min_threshold,
  hpux.data$max_threshold - hpux.data$cpu as margin,
  CASEWHEN(hpux.data$max_threshold - hpux.data$cpu < 0,
    2,
    CASEWHEN(hpux.data$max_threshold - hpux.data$cpu < 10, 1, 0) ) as condition
FROM
  users.admin.queries.cpu_HPUX:data as hpux
UNION ALL
(SELECT
  standard.data$icon AS icon,
  standard.data$device AS device,
  standard.data$context AS context,
  standard.data$cpu AS cpu,
  standard.data$max_threshold as max_threshold,
  standard.data$min_threshold as min_threshold,
  standard.data$max_threshold - standard.data$cpu as margin,
```

```
  CASEWHEN(standard.data$max_threshold - standard.data$cpu < 0,
    2,
    CASEWHEN(standard.data$max_threshold - standard.data$cpu < 10, 1, 0) ) as
condition
FROM
  users.admin.queries.cpu_standard:data as standard
UNION ALL
(SELECT
  sun.data$icon AS icon,
  sun.data$device AS device,
  sun.data$context AS context,
  sun.data$cpu AS cpu,
  sun.data$max_threshold as max_threshold,
  sun.data$min_threshold as min_threshold,
  sun.data$max_threshold - sun.data$cpu as margin,
  CASEWHEN(sun.data$max_threshold - sun.data$cpu < 0,
    2,
    CASEWHEN(sun.data$max_threshold - sun.data$cpu < 10, 1, 0) ) as condition
FROM
  users.admin.queries.cpu_Sun:data as sun))
ORDER BY
  margin
```

12.12.10 Синтаксис языка запроса

Данная статья не дает никаких *объяснений*. В ней просто *представлены* конструкции и функции SQL, которые могут использоваться в <%AG %>. Дело в том, что обучение читателя языку запросов SQL не является целью данного руководства. Если вы недостаточно хорошо знакомы с этой темой, мы крайне рекомендуем вам обратиться к ссылкам на источник 833, указанным в статье Запросы 829.

Каждый запрос, сформированный при помощи языка запросов AtomMind, имеет следующий синтаксис:

selectStatement

```
SELECT [ALL | DISTINCT]
{ selectExpression | tableAlias.fieldReference | tableAlias.* | * } [, ...]
FROM tableList
[WHERE expression]
[GROUP BY expression [, ...]]
[HAVING expression]
[{ UNION [ALL] | MINUS | INTERSECT } selectStatement]
[ORDER BY orderExpression) [, ...]]
[LIMIT <limit> [OFFSET <offset>]]
```

Раздел LIMIT может использоваться, только если все контекстные ссылки 833 в разделе OT имеют альтернативные имена (например, "SELECT ... FROM context_reference AS **alias** LIMIT n, m").

tableList

```
table [joinedTables] [, ...]
```

joinedTables

```
joinedTable [joinedTable] [...]
```

joinedTable

```
{CROSS | INNER | LEFT OUTER | RIGHT OUTER | FULL OUTER} JOIN table ON expression
```

table

```
{ (selectStatement) [AS tableAlias] | contextReference [AS tableAlias]}
```

contextReference

См. Контекстные ссылки 833.

```
contextMask { :contextEntityReference [: ...] }
```

contextEntityReference

```
{ contextVariableName | contextVariableGroupName.* | contextFunctionName
( contextFunctionParameters ) }
```

contextFunctionParameters

```
{ null | 'value' | "value"} [, ...]
```

orderExpression

```
{ [tableAlias.]fieldReference | selectExpression } [ASC | DESC]
```

selectExpression

```
{ expression | COUNT(*) | {  COUNT | MIN | MAX | SUM | AVG }  ([ALL | DISTINCT]
expression) } [[AS] label]
```

expression

```
[NOT] condition [{ OR | AND } condition]
```

condition

```
{
   value [|| value]
 | value { = | < | <= | > | >= | <> | != } value
 | value IS [NOT] NULL
 | EXISTS(selectStatement)
 | value BETWEEN value AND value
 | value [NOT] IN ( {value [, ...] | selectStatement } )
 | value [NOT] LIKE value [ESCAPE] value
}
```

value

```
[+ | -] literal   [{ + | - | * | / | || }literal  ]
 | ( condition )
 | function ( [parameter] [,...] )
 | selectStatement giving one value
 | {ANY|ALL} (selectStatement giving single column  )
 | INTERVAL literal {YEAR | MONTH | DAY | HOUR | MINUTE}
}
```

fieldReference

См. Ссылки на поля 836.

```
{ { contextVariableName | contextFunctionName} $ dataTableFieldName |
writebackFieldName }
```

writebackFieldName

"CONTEXT_ID" | "PARENT_ID" | "RECORD_INDEX"

Литералы

Язык запросов AtomMind определяет несколько типов литералов:

- Литерал Null: **NULL**
- Логические литералы: **TRUE** или **FALSE**
- Десятичные литералы (**0**, **1**, **123**, **-1234567890**, ...)
- Шестнадцатиричные литералы (**X'0A'**, **X'FFFF'**, ...)
- Двоичные литералы (**B'01'**, **B'00110011'**)
- Литералы с плавающей точкой (**3.1**, **-44.5**, **1.3E12**, ...)
- Строковые литералы (**'This is a String'**, **'test'**, ...)

12.12.10.1 Функции языка запроса

Раздел содержит все функции, доступные на языке запроса AtomMind.

Встроенные числовые функции

Функция	Описание
ABS(Number d)	Возвращает абсолютное значение *d*
ACOS(Number d)	Возвращает арккосинус угла
ASIN(Number d)	Возвращает арксинус угла
ATAN(Number d)	Возвращает арктангенс угла
ATAN2(Number a,Number b)	Возвращает тангенс *a/b*
BITAND(Number a, Numberb)	Возвращает *a & b*
BITNOT(Number a)	Возвращает !*a*.
BITOR(Number a, Number b)	Возвращает a | b
BITXOR(Number a, Number b)	Возвращает *a ^ b*.
CEILING(Number d)	Возвращает наименьшее целое число не менее *d*
COS(Number d)	Возвращает косинус угла
COT(Number d)	Возвращает котангенс угла
DEGREES(Number d)	Конвертирует радианы в градусы
EXP(Number d)	Возвращает e (2.718...), в степени *d*
FLOOR(Number d)	Возвращает наибольшее целое число не более *d*
LOG(Number d)	Возвращает натуральный логарифм по основанию e
LOG10(Number d)	Возвращает натуральный десятичный логарифм
MOD(Number a, Number b)	Возвращает a по модулю b
PI()	Возвращает число Пи (3.1415...)
POWER(Number a, Number b)	Возвращает a, в степени b
RADIANS(Number d)	Конвертирует градусы в радианы

RAND()	Возвращает случайное число x от 0.0 до 1.0
ROUND(Number a, Number b)	Округляет a до b знаков после запятой
ROUNDMAGIC(Number d)	Решает проблемы округления, такие как 3.11-3.1-0.01
SIGN(Number d)	Возвращает -1, если d менее 0, 0, если d==0 и 1, если d более 0
SIN(Number d)	Возвращает синус угла
SQRT(Number d)	Возвращает квадратный корень
TAN(A)	Возвращает тангенс угла
TRUNCATE(a,b)	Сокращает a до b знаков после запятой

Строковые встроенные функции

Функция	**Описание**
ASCII(String s)	Возвращает ASCII код самого левого символа *s*
BIT_LENGTH(String str)	Возвращает длину строки в битах
CHAR(Integer c)	Возвращает символ, который содержит ASCII код c
CONCAT(String s1, String s2)	Возвращает *s1* + *s2*.
DIFFERENCE(String s1, String s2)	Возвращает разницу между s1 и s2
HEXTORAW(String s)	Возвращает переведенную строку
INSERT(String s, Integer start, Integer length, String replacement)	Возвращает строку, где *length* числа символов, начинающихся со *start*, была заменена на *replacement*
LENGTH(String s)	Возвращает количество символов в строке s
LOCATE(String search, String s[, Integer start])	Возвращает первый индекс (1=левый, 0=не найден), где *search* найден в строке *s*, начинающейся со *start*.
LOWER(String s)	Конвертирует *s* в строчный регистр
LPAD(String s1, Integer length[, String s2])	Возвращает строку символов длиной *length* символов. Строка содержит символы *s1*, перенесенные пробелами налево. Если *length* меньше, чем длина аргумента *s1*, аргумент отсекается. Если определена опциональная строка *s2*, эта строка используется для заполнения поля вместо пробелов.
LTRIM(String s)	Убирает все пробелы в начале строки *s*
OCTET_LENGTH(String str)	Возвращает длину строки в битах (вдвое больше числа символов)
RAWTOHEX(String s1)	Возвращает переведенную строку
REGEXP_MATCHES(String s, String regex)	Возвращает true, если *s* соответствует *regex* как целое. *Regex* определяется в соответствии с регулярными правилами выражения 2142.
REGEXP_SUBSTRING(String s, String regex)	Возвращает первую область в *s*, которая соответствует *regex*. *Regex* определяется в соответствии с регулярными правилами выражения 2142.
REPEAT(String s, Integer count)	Возвращает *s*, повторяемое определенное количество раз
REPLACE(String s, String replace, String s2)	Заменяет все случаи *replace* в *s* на *s2*
REVERSE(String s)	Возвращает строку символов на основе *s* с символами в обратном порядке.
RPAD(String s1, Integer length[, String s2])	Возвращает строку символов длиной *length* символов. Строка содержит символы строки *s1*, перенесенные пробелами направо. Если *length* меньше, чем длина аргумента *s1*, аргумент отсекается. Если определена опциональная строка *s2*, эта строка используется для заполнения поля вместо пробелов.

RTRIM(String s)	Убирает все пробелы в конце *s*
SOUNDEX(String s)	Возвращает четырехсимвольный код, представляющий звук *s*
SPACE(Integer count)	Возвращает строку, состоящую из определенного количества пробелов
SUBSTRING(String s, Integer start[, String len])	Возвращает подстроку, начинающуюся с *start* (1=первый символ слева) длиной *len*
UPPER(String s)	Конвертирует s в верхний регистр

Встроенные функции даты/времени

Функция	**Описание**
CURDATE()	Возвращает текущую дату
CURTIME([<time precision>])	Возвращает текущее время как значение типа ВРЕМЯ И ВРЕМЕННАЯ ЗОНА.
LOCALTIME()	Возвращает текущее время как значение типа ВРЕМЯ.
CURRENT_TIMESTAMP [(<timestamp precision>)]	Возвращает текущую дату/время как значение типа ВРЕМЕННАЯ МЕТКА И ВРЕМЕННАЯ ЗОНА.
LOCALTIMESTAMP [(<timestamp precision>)]	Возвращает текущую дату/время как значение типа ВРЕМЕННАЯ МЕТКА.
DATEADD(String string, Integer number, Date datetime)	Строка указывает единицу времени и может принимать следующие значения 'ms'='millisecond', 'ss'='second','mi'='minute','hh'='hour', 'dd'='day', 'mm'='month', 'yy' = 'year'. Можно использовать как длинную, так и короткую форму обозначения временного интервала.
DATEDIFF(String string, Date datetime1, Date datetime2)	Возвращает количество единиц времени, прошедших с datetime1 до datetime2. Строка указывает единицу времени и может иметь следующие значения 'ms'='millisecond', 'ss'='second','mi'='minute','hh'='hour', 'dd'='day', 'mm'='month', 'yy' = 'year'. Можно использовать как длинную, так и короткую форму строки.
DAYNAME(Date date)	Возвращает название дня
DAYOFMONTH(Date date)	Возвращает день месяца (1-31)
DAYOFWEEK(Date date)	Возвращает день недели (1 означает воскресенье)
DAYOFYEAR(Date date)	Возвращает день года (1-366)
HOUR(Date time)	Возвращает час (0-23)
MINUTE(Date time)	Возвращает минуту (0-59)
MONTH(Date date)	Возвращает месяц (1-12)
MONTHNAME(Date date)	Возвращает название месяца
NOW()	Эквивалент LOCALTIMESTAMP().
QUARTER(Date date)	Возвращает квартал (1-4)
SECOND(Date time)	Возвращает секунду (0-59)
SECONDS_SINCE_MIDNIGHT(Date time)	Возвращает целое число в пределах 0 - 86399, определяющее количество секунд, истекших после полуночи.
TIMESTAMP(Object value)	Эта функция переводит аргумент во ВРЕМЕННУЮ МЕТКУ БЕЗ ВРЕМЕННОЙ ЗОНЫ. Когда единственный аргумент представляет собой числовое значение, он интерпретируется как временная метка Unix в секундах. Когда аргумент представляет собой строку, его значение переводится во временную метку.
TIMESTAMP_WITH_ZONE(Object value)	This function translates the arguments into a TIMESTAMP value. When the single argument is a numeric value, it is interpreted as a Unix timestamp in seconds. When the argument is a string, it's parsed to a timestamp.

TO_CHAR(Date date, String format)	Форматирует *date* в строку *format*, представленную во втором аргументе. Строка формата описана здесь 2146.
TO_TIMESTAMP(String value, String format)	Преобразует *datetime* во временную метку, используя формат. Строка формата описана здесь 2146.
UNIX_MILLIS(Date date)	Возвращает количество миллисекунд, начиная с 1970-01-01 (the Epoc).
UNIX_TIMESTAMP(Date date)	Возвращает количество секунд, начиная с 1970-01-01 (the Epoc).
WEEK(Date date)	Возвращает неделю года (1-53)
YEAR(Date date)	Возвращает год

Компоненты поддерживаемого формата для функций TO_CHAR и TO_TIMESTAMP в верхнем регистре:

BC | B.C. | AD | A.D.	Возвращает AD для нашей эры и BC для до нашей эры.
RRRR	Год в четырехзначном формате.
YYYY	Год в четырехзначном формате.
IYYY	Год в четырехзначном формате, сответствует неделе ISO в году. Отчетный год за последние несколько дней календарного года может быть следующим годом.
YY	Год в двузначном формате.
IY	Год в двузначном формате, сответствует неделе ISO в году.
MM	Месяц (01-12).
MON	Краткое название месяца из трех букв.
MONTH	Название месяца.
WW	Неделя года (1-53), где неделя 1 начинается в первый день года и продолжается до седьмого дня года (не календарная неделя).
W	Неделя месяца (1-5), где неделя 1 начинается в первый день месяца и заканчивается на седьмой (не календарная неделя).
IW	Неделя года (1-52 или 1-53) на основе стандарта ISO. Неделя начинается с понедельника. Первая неделя может начинаться под конец предыдущего года.
DAY	Название дня.
DD	День месяца (01-31).
DDD	День года (1-366).
DY	Краткое название дня из трех букв.
HH	Час суток (00-11).
HH12	Час суток (00-11).
HH24	Час суток (00-23).
MI	Минута (00-59).
SS	Секунда (00-59).
FF	Дробные секунды.

Встроенные функции выражения

Функция	**Описание**
EVALUATE_TO_STRING(String expression, String defaultContext, String defaultTable [, String agrument0, String argument1, ...])	Вычисляет выражение AtomMind 112 (указанное параметром *выражение*). Контекст по умолчанию 119 для вычисления задан параметром *defaultContext*. Таблица по умолчанию 120 задана параметром *defaultTable*. Любая ссылка на поле 836, указывающая на поле Таблицы данных, может быть использована как *defaultTable*, так как в языке запросов поля Таблицы данных переменных/функций контекста представлены полями строк.

	Аргументы (*argument0*, *argument1*, ...) доступны через переменные среды, названные 0, 1, и так далее. Например, для доступа к *argument3* выражения используйте следующую ссылку: `{env/3}`. Эта функция преобразует результат выражения AtomMind в строку и возвращает эту строку.
EVALUATE_TO_INT(String expression, String defaultContext, String defaultTable [, String agrument0, String argument1, ...])	Действует как EVALUATE_TO_STRING, но преобразует результат выражения AtomMind в целое число.
EVALUATE_TO_LONG(String expression, String defaultContext, String defaultTable [, String agrument0, String argument1, ...])	Действует как EVALUATE_TO_STRING, но преобразует результат выражения AtomMind длинное число (целое число большого диапазона).
EVALUATE_TO_DOUBLE(String expression, String defaultContext, String defaultTable [, String agrument0, String argument1, ...])	Действует как EVALUATE_TO_STRING, но преобразует результат выражения AtomMind в цисло с плавающей запятой.
EVALUATE_TO_BOOLEAN(String expression, String defaultContext, String defaultTable [, String agrument0, String argument1, ...])	Действует как EVALUATE_TO_STRING, но преобразует результат выражения AtomMindв логическое.
EVALUATE_TO_DATE(String expression, String defaultContext, String defaultTable [, String agrument0, String argument1, ...])	Действует как EVALUATE_TO_STRING, но преобразует результат выражения AtomMind в дату.

Встроенные системные функции

Функция	**Описание**
CONVERT(Object exp, String type)	Конвертирует *exp* в другой тип данных. Поддерживаемые типы данных: • целочисленные:SQL_TINYINT, SQL_SMALLINT, SQL_INTEGER, SQL_BIGINT, SQL_NUMERIC, SQL_DECIMAL, SQL_BOOLEAN • с плавающей точкой: SQL_REAL, SQL_FLOAT, SQL_DOUBLE • строковые: SQL_CHAR, SQL_VARCHAR, SQL_LONGVARCHAR • даты/времени: SQL_DATE, SQL_TIME, SQL_TIMESTAMP Пример: `CONVERT(10, SQL_STRING)`
CASEWHEN(Object exp, Object v1, Object v2)	Если выражение exp является true, возвращается v1, в противном случае - v2. Функция поддерживает поля, которые могут иметь знаечние null.
COALESCE(Object expr1, Object expr2,Object expr3,...)	Если выражение exp 1 имеет непустое значение (т.е. не null), оно возвращается, или же вычисляется exp 2, и если оно не имеет пустое значение, оно также возвращается и т.д.
NULLIF(Object v1, Object v2)	Если v1 равно v2, возвращается пустое значение, в противном случае - v1

12.13 Машинное обучение

Машинное обучение - подход к искусственному интеллекту, который позволяет компьютерной программе выполнять задачу без прямого программирования.

Обычно выделяют две основные категории задач машинного обучения: обучение с учителем и без учителя.

Для **обучения с учителем** требуется размеченный набор данных для обучения модели машинного обучения. Метка - это значение целевой переменной (переменной, которая считается зависимой от одной или более независимых переменных, также называемых признаками), которая принимается моделью за достоверные данные.

Обучение без учителя не требует размеченных данных и используется для того, чтобы выделить структуру из входных данных.

Модуль машинного обучения платформы AtomMind разработан для решения трех основных типов задач машинного обучения с учителем: регрессия, классификация и обнаружение аномалий.

Регрессия

Регрессия - это задача машинного обучения, в которой предсказываемая переменная является действительным числом.

Типичным примером задачи регрессии является предварительная оценка стоимости дома на основе нескольких пизнаков, таких как жилая площадь, количество комнат и этажей, район города и т.д.

Классификация

При классификации выходные значения делятся на два или более классов. Задача заключается в отнесении новых входных данных к одному или нескольким из этих классов.

Примером классификации может служить задача по определению, содержит ли картинка текст (пример бинарной классификации), или задача оптического распознавания символов (многоклассовая классификация).

Обнаружение аномалий

Обнаружение аномалий можно рассматривать как одноклассовую классификацию. Модель машинного обучения учится на наборе данных, содержащем только неаномальные ("нормальные") данные, а затем используется для предположения аномальности нового экземпляра данных. Иногда одноклассовая классификация считается задачей машинного обучения без учителя, так как все метки в обучающем наборе одинаковые ("нормальные") и становятся в определенном смысле ненужными.

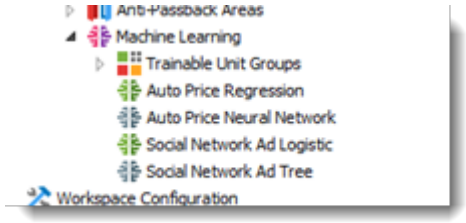
Обучаемый модуль

Экземпляр контекста машинного обучения[1535] называется обучаемым модулем[1599]. Обучаемый модуль предоставляет функции и действия, необходимые для построения и оценки качества модели машинного обучения.

Каждый пользователь[478] имеет свой собственный набор обучаемых модулей.

Администрирование обучаемых модулей

Два контекста используются для администрирования обучаемых модулей: один - это общий контекст машинного обучения[1535], который служит контейнером. Другой -это контекст обучаемого модуля[1599], который содержит информацию для отдельного обучаемого модуля.



12.13.1 Стадии работы

Этот раздел описывает типичный жизненный цикл обучаемого модуля.

Создание и конфигурирование

Прежде всего, обучаемый модуль необходимо создать. Для этого используется действие Создать контекста машинного обучения[1535].

После этого, обучаемый модуль необходимо сконфигурировать[869]. Свойства обучаемого модуля описываются в соответствующем разделе[870]. При конфигурировании свойств обучаемого модуля убедитесь, что выбор задачи и

алгоритма 867 соответствует вашей бизнес-задаче. Обязательно укажите имя поля с зависимой переменной, которое будет указывать на колонку с зависимой переменной вашего набора данных. Обратите внимание, что все наборы данных, передаваемые функциям обучаемого модуля, должны содержать поле с таким же именем, иначе будет отображено сообщение об ошибке. Обратите внимание, что невозможно создать обучаемый модуль, если Имя и Имя поля с зависимой переменной остались незаполненными.

Если у экземпляров данных есть веса, соответствующая настройка должна быть выставлена на true и предоставлено Имя поля с весами. Обратите внимание, что это имя не должно совпадать с Именем поля с зависимой переменной во избежание ошибки. Тип поля, соответствующего весам, должен быть числовым.

Обучение

Когда обучаемый модуль создан и сконфигурирован, его задача и алгоритм установлены, гиперпараметры 870 алгоритма заданы, внутренние фильтры 878 (при необходимости) добавлены, обучаемый модуль может быть обучен на обучающем наборе данных. Для этой цели используется функция Обучить 1599.

Обучающий набор должен быть передан функции как таблица данных 49, формат 51 которой должен удовлетворять требованиям, приведенным в соответствующем разделе 869.

Как только процесс обучения обучаемого модуля окончен, его иконка меняется с  (указывает на необученный обучаемый модуль) на  (указывает на обученный обучаемый модуль). Кроме того, состояние обучаемого модуля сохраняется в хранилище конфигураций. Если необходим перезапуск AtomMind Server, каждый обучаемый модуль восстанавливает свое состояние после перезагрузки.

Результат вызова функции Обучить на уже обученном обучаемом модуле зависит от того, основан ли этот обучаемый модуль на алгоритме, поддерживающем дообучение. Если да, то обучаемый модуль будет обновлен (а не обучен заново). Если обучаемый модуль использует недообучаемый алгоритм, то он будет обучен заново.

Если необходимо сбросить состояние обучаемого модуля, можно использовать функцию Сбросить 1604. Эта функция также представлена одноименным действием.

При изменении любой опции обучаемого модуля, его статус будет сброшен и, как следствие, потребуется повторное обучение.

Оценка

После пройденного обучения обучаемый модуль может использоваться для предсказания на новых данных. Однако, желательно проводить оценку качества работы модели машинного обучения до начала ее применения для предсказаний. Функция Оценить 1602 проводит оценку работы обученного модуля на тестовом наборе данных и возвращает набор оценочных метрик. Тестовый набор должен передаваться функции как таблица данных, формат которой должен удовлетворять требованиям, приведенным в соответствующем разделе 869.

Оценочная статистика сохраняется в хранилище конфигураций. Таким образом, если функция Оценить вызывается более одного раза, вновь полученные метрики оценки добавляются к общей статистике. При необходимости сброса статистики вычисления, можно использовать функцию Сбросить оценку 1604. Эта функция также представлена одноименным действием.

При вызове функции Оценить на необученном обучаемом модуле, появится сообщение об ошибке.

Перекрестная проверка

Еще один способ провести оценку обучаемого модуля - использовать функцию Провести перекрёстную проверку 1604. Функция осуществляет перекрестную проверку по K частям данных (folds) на заданном наборе данных и возвращает тот же набор оценочных метрик, что и функция Оценить. Значения метрик оценки усредняются по K частям данных. Число блоков K и начальное число для генератора случайных чисел определяются в соответствующих параметрах конфигурирования. Количество частей данных не должно превышать количество записей в тестовом наборе. Требования к входному набору данных такие же, как для функции Оценить.

Имейте в виду, что функция Провести перекрёстную проверку может использоваться даже на необученном обучаемом модуле. Эта функция не меняет статус обучаемого модуля (если он был необученным, то остается необученным; если он был обучен, переобучение не проводится).

Вам может понадобиться использовать перекрёстную проверку, если вы не хотите разбивать ваш набор данных на обучающий набор и тестовый набор (например, в случае если набор данных слишком мал). Если оценочные метрики, возвращенные функцией Провести перекрёстную проверку, доказывают, что модель функционирует желаемым образом, весь набор данных может быть использован как обучающий набор для обучения модели.

Использование обучаемого модуля для предсказаний

Теперь можно использовать обученный и прошедший оценку обучаемый модуль для предсказаний. Этой цели служит функция Использовать 1601. Важно, чтобы формат таблицы данных, передаваемой в качестве аргумента функции Использовать, соответствовал формату таблицы данных, которая использовалась для обучения обучаемого модуля. Функцию Использовать можно применять на данных, как содержащих так и не содержащих значения зависимой переменной. При использовании на данных, не содержащих значений зависимой переменной, столбец с зависимой переменной должен быть заполнен значениями NULL. Использование функции во втором случае (на данных со значениями зависимой переменной) может служить еще одним инструментом оценки функционирования модели машинного обучения. Предсказанные и фактические значения (включая ошибку для задач регрессии) будут включены в итоговую таблицу данных совместно с соответствующими признаками (независимыми переменными).

При вызове функции Использовать на необученном обучаемом модуле, появится сообщение об ошибке.

12.13.2 Алгоритмы

Количество алгоритмов, поддерживаемых модулем машинного обучения AtomMind, постоянно увеличивается. Текущий набор алгоритмов представлен ниже.

Алгоритмы для задач регрессии:

Алгоритм	Описание
Линейная регрессия	Множественная линейная регрессия с L2-регуляризацией, отбором признаков и возможностью исключать коллинеарные признаки. Поддерживает числовые, категориальные признаки и признаки даты. Поддерживает экземпляры набора данных с весами.
Регрессия опорных векторов (SVR)	Реализация метода опорных векторов (SVM) для регрессии. Поддерживает несколько известных ядер. Входные значения могут быть нормализованы и стандартизированы по необходимости. Поддерживает числовые и категориальные признаки. Поддерживает экземпляры набора данных с весами.
Дерево Решений с REP	«Быстрое дерево решений» с отсечением ветвей по приведенной погрешности (Reduced Error Pruning). Поддерживает числовые, категориальные признаки и признаки даты. Поддерживает экземпляры набора данных с весами.
Случайный лес	Лес случайных деревьев. Поддерживает числовые, категориальные признаки и признаки даты. Поддерживает экземпляры набора данных с весами.
Многослойный перцептрон	Реализация нейронной сети прямого распространения, которая обучается с использованием обратного распространения ошибки. Функция активации узлов во всех скрытых слоях является сигмоидой. Узлы в выходном слое являются невыпрямляющими линейными элементами. Входные значения могут быть нормализованы по необходимости. Поддерживает числовые, категориальные признаки и признаки даты. Поддерживает экземпляры набора данных с весами.
Стохастический градиентный спуск	Применяет стохастический градиентный спуск для обучения различных линейных моделей (бинарная классификация методом опорных векторов, бинарная логическая регрессия, квадратичная функция потерь, функция потерь Хьюбера и линейная регрессия эпсилон-нечувствительной функции потерь). Замещает все отсутствующие значения и преобразует категориальные признаки в бинарные. Алгоритм также нормализует все признаки таким образом, что коэффициенты на выходе основаны на нормализованных данных. Является дообучаемым.

Алгоритмы для задач классификации:

Алгоритм	Описание
Логистическая регрессия	Многоклассовая логистическая регрессия с гребневой (ridge) регуляризацией. Поддерживает числовые, категориальные признаки и признаки даты. Поддерживает экземпляры набора данных с весами.
Метод опорных векторов (SVM)	Реализация алгоритма последовательной минимальной оптимизации для метода опорных векторов. Значения признаков могут быть нормализованы или стандартизированы по необходимости. Многоклассовые задачи решаются с использованием попарной классификации. Поддерживает числовые и категориальные признаки. Поддерживает экземпляры набора данных с весами.
Дерево Решений с REP	«Быстрое дерево решений» с отсечением ветвей по приведенной погрешности (Reduced Error Pruning). Поддерживает числовые, категориальные признаки и признаки даты. Поддерживает экземпляры набора данных с весами.

Случайный лес	Лес случайных деревьев. Поддерживает числовые, категориальные признаки и признаки даты. Поддерживает экземпляры набора данных с весами.
Многослойный перцептрон	Реализация нейронной сети прямого распространения. Функцией активации всех узлов является сигмоида. Входные признаки могут быть нормализованы по необходимости. Поддерживает числовые, категориальные признаки и признаки даты. Поддерживает экземпляры набора данных с весами.
Наивный байесовский классификатор	Наивный байесовский классификатор с использованием классов оценки. Поддерживает числовые икатегориальные признаки. Поддерживает экземпляры набора данных с весами.
Стохастический градиентный спуск	Применяет стохастический градиентный спуск для обучения различным линейным моделям (бинарная классификация методом опорных векторов, бинарная логическая регрессия, квадратичная функция потерь, функция потерь Хьюбера и линейная регрессия эпсилон-нечувствительной функции потерь). Замещает все отсутствующие значения и преобразует категориальные признаки в бинарные. Алгоритм также нормализует все признаки таким образом, что коэффициенты на выходе основаны на нормализованных данных. Является дообучаемым.
Дерево Хёфдинга	Дерево Хёфдинга (VFDT) является инкрементным индукционным алгоритмом обучения дерева решений, способным обучаться на массивных потоках данных при условии, что распределение, по которому распределены образцы данных, остается неизменным во времени. Деревья Хёфдинга используют тот факт, что часто бывает достаточно малой выборки для выбора оптимального признака расщепления. Является дообучаемым.
Многоклассовый дообучаемый классификатор	Метаклассификатор для обработки многоклассовых наборов данных при помощи двухклассового классификатора. Этот классификатор также способен применять выходные коды с исправлением ошибок для большей точности. Базовый классификатор должен быть дообучаемым. Является дообучаемым.

Алгоритмы для задач обнаружения аномалий:

Алгоритм	**Описание**
Одноклассовый метод опорных векторов	Реализация одноклассового метода опорных векторов для обнаружения аномалий. Поддерживает числовые, категориальные признаки и признаки даты.

Метаалгоритмы:

Алгоритм	**Описание**
Алгоритм с предварительной фильтрацией	Реализация выбираемого классификатора на данных, которые прошли через выбираемый фильтр. Поддерживает экземпляры набора данных с весами. Является дообучаемым в случае, если базовый алгоритм является дообучаемым.
Многоклассовый дообучаемый классификатор	Метаклассификатор для обработки многоклассовых наборов данных при помощи двухклассового классификатора. Этот классификатор также способен применять выходные коды с исправлением ошибок для большей точности. Базовый классификатор должен быть дообучаемым. Является дообучаемым.

Дообучаемые алгоритмы:

Алгоритм	**Описание**
Алгоритм с предварительной фильтрацией	Реализация выбираемого классификатора на данных, которые прошли через выбираемый фильтр. Поддерживает экземпляры набора данных с весами. Является дообучаемым в случае, если базовый алгоритм является дообучаемым.
Дерево Хёфдинга	Дерево Хёфдинга (VFDT) является инкрементным индукционным алгоритмом обучения дерева решений, способным обучаться на массивных потоках данных при условии, что распределение, по которому распределены образцы данных, остается неизменным во времени. Деревья Хёфдинга используют тот факт, что часто бывает достаточно малой выборки для выбора оптимального признака расщепления. Является дообучаемым.

Многоклассовый дообучаемый классификатор	Метаклассификатор для обработки многоклассовых наборов данных при помощи двухклассового классификатора. Этот классификатор также способен применять выходные коды с исправлением ошибок для большей точности. Базовый классификатор должен быть дообучаемым. Является дообучаемым
Стохастический градиентный спуск	Применяет стохастический градиентный спуск для обучения различным линейным моделям (бинарная классификация методом опорных векторов, бинарная логическая регрессия, квадратичная функция потерь, функция потерь Хьюбера и линейная регрессия эпсилон-нечувствительной функции потерь). Замещает все отсутствующие значения и преобразует категориальные признаки в бинарные. Алгоритм также нормализует все признаки таким образом, что коэффициенты на выходе основаны на нормализованных данных. Является дообучаемым.

Каждый алгоритм имеет свой набор гиперпараметров. Для более подробной информации см. Гиперпараметры алгоритмов 870.

12.13.3 Аргументы функций обучаемого модуля

Типы полей, поддерживаемые функциями обучаемого модуля, включают: Integer, Long, Float, Double, Boolean, String, и Date.

Поле типа **Integer**, **Long**, **Float**, или **Double,** не имеющее допустимых значений, представляет собой числовой признак.

Для задач регрессии поле с зависимой переменной должно быть одного из четырех числовых типов. Заметим, что независимо от типа поля с зависимой переменной, предсказываемые значения будут типа Double.

Поле любого поддерживаемого типа (включая числовые), которое имеет **допустимые значения**, а также поле типа **Boolean**, представляет собой категориальный признак.

Для задач классификации колонка с зависимой переменной должна содержать категориальные данные. Лучший способ представления категориального признака или поля с зависимой переменной - использовать **поле типа String с допустимыми значениями,** которые содержат все возможные классы. Заметим, что независимо от типа поля с зависимой переменной, предсказываемый класс будет типа String (с допустимыми значениями, которые включают все возможные классы).

Если задача определена как классификация, и поле с зависимой переменной не имеет допустимых значений, обучаемый модуль будет пытаться преобразовать значения в поле с зависимой переменной в категориальные, используя все уникальные значения как допустимые. Однако предполагается, что тестовая выборка не будет содержать значений, которых не было в обучающей выборке (т.е. тестовая выборка не должна содержать класс, которого не было в обучающей выборке).

Поле типа **Date** представляет собой признак соответствующего типа (Date).

Значения Null обрабатываются как отсутствующие. Кроме того, при невозможности предсказания для определенного экземпляра данных, функция Operate вернет Null в качестве предсказания.

Формат таблицы данных, переданной функции Train, сохраняется как ссылка для валидации формата. Таким образом, таблица данных, переданная функции Operate 1601 или Evaluate 1602, должна иметь абсолютно такой же формат, как и таблица данных, используемая для обучения (включая допустимые значения, возможность содержать значения Null и пр.). Несоответствие форматов приведет к ошибке.

Таблицы данных, переданные функциям Train 1599, Evaluate 1602 или Cross Validate 1604, должны содержать как минимум один экземпляр. Количество экземпляров также не может быть меньше, чем количество частей данных для перекрестной проверки.

12.13.4 Конфигурирование обучаемого модуля

Описывает свойства конфигурации обучаемого модуля.

Все описанные здесь свойства доступны при помощи действия Конфигурировать 105 в контексте Обучаемого модуля 1599.

12.13.4.1 Свойства обучаемого модуля

Описывает основные опции обучаемого модуля.

Описание поля	Имя поля
Имя. Имя обучаемого модуля. Оно должно удовлетворять соглашениям по наименованию 42 контекста. Имя необходимо для ссылки на этот обучаемый модуль из других частей системы.	name
Описание. Текстуальное описание 43 обучаемого модуля.	description
Задача. Задача машинного обучения, **регрессия**, **классификация** или **обнаружение аномалий**. После создания обучаемого модуля задача не может быть изменена для этого обучаемого модуля.	task
Алгоритм. Алгоритм машинного обучения для решения конкретной бизнес-задачи. Перечень доступных алгоритмов зависит от выбранной задачи. Дополнительную информацию можно найти в алгоритмы 867.	algorithm
Гиперпараметры. Параметры, которые устанавливаются до начала процесса обучения. Дополнительную информацию можно найти в гиперпараметры алгоритмов 870.	hyperparameters
Имя поля столбца с зависимой переменной. Имя поля столбца с зависимой переменной (целевая переменная). Дополнительную информацию можно найти в аргументы функций обучаемых модулей 869.	labelFieldName
У набора данных есть веса. Определяет, есть ли у экземпляров набора данных веса.	hasWeights
Имя поля с весами. Имя поля, содержащего веса, если экземпляры набора данных имеют значение веса. Отображается только если опция **У набора данных есть веса** установлена на true.	weightFieldName
Количество частей данных. Количество частей данных, на которые будет разбит набор данных для перекрестной проверки (при вызове функции crossValidate).	cvNumFolds
Случайное начальное число. Начальное число для генератора случайных чисел, используемого при разбиении данных на части для перекрестной проверки.	cvRandomSeed

12.13.4.2 Гиперпараметры алгоритмов

Гиперпарамерты являются параметрами алгоритма машинного обучения, которые устанавливаются до начала процесса обучения.

Параметры, общие для всех алгоритмов:

Описание поля	Имя поля
Размер пакета данных. Предпочтительное количество экземпляров данных для обработки в случае пакетного режима предсказания. Большее или меньшее количество экземпляров может быть предоставлено, но данное число даёт алгоритмам возможность задать предпочтительный размер пакета данных.	batchSize
Число десятичных знаков. Число десятичных знаков, которое будет использовано для отображения численных результатов в информации, возвращаемой функцией "Обучить".	numDecimalPlaces

Параметры, специфичные для алгоритма:

ЛИНЕЙНАЯ РЕГРЕССИЯ

Описание поля	Имя поля

Метод отбора признаков. Определяет метод, используемый для отбора признаков для линейной регрессии. Доступны следующие методы: без отбора признаков, отбор признаков с помощью метода M5 (прохождение по атрибутам, удаляя атрибут с наименьшим стандартизованным коэффициентом, до тех пор, пока не перестанет наблюдаться улучшение в оценке ошибки, задаваемой информационным критерием Акаике) и жадный метод отбора, основанный на информационной метрике Акаике.	attributeSelectionMethod
Исключать коллинеарные признаки. Определяет, исключать ли коллинеарные признаки.	eliminateCollinearAttributes
Minimal. Если включено, то средние значения и стандартные отклонения будут отброшены для сохранения памяти. Также, информация об обученном модуле, возвращаемая функцией "Обучить", будет сокращена.	minimal
Параметр регуляризации. Значение параметра регуляризации.	ridge
Отображать дополнительные статистические данные. Определяет, следует ли отображать дополнительные статистические данные (такие как стандартное отклонение коэффициентов и t-статистика) в информации об обученном модуле для регрессионного анализа.	outputAdditionalStats

ЛОГИСТИЧЕСКАЯ РЕГРЕССИЯ

Описание поля	Имя поля
Максимальное количество итераций. Максимальное количество итераций, которые будут выполнены.	maxIts
Параметр регуляризации. Значение параметра регуляризации.	ridge
Использовать метод сопряженных градиентов. Использовать метод сопряжённых градиентов вместо BFGS - быстрее для задач с большим количеством параметров.	useConjugateGradientDescent

МНОГОСЛОЙНЫЙ ПЕРЦЕПТРОН

Описание поля	Имя поля
Затухание. Эта настройка (decay) будет уменьшать скорость обучения: начальная скорость обучения будет поделена на номер эпохи для определения того, какой должна быть текущая скорость обучения. Это может помочь предотвратить расхождение нейронной сети от заданной выходной величины, а также улучшить качество работы нейросети в целом.	decay
Скрытые слои. Задает скрытые слои нейросети. Этот параметр должен состоять из списка целых чисел (одно число для каждого скрытого слоя), разделенных запятой. Если скрытых слоев нет, поместите сюда одно число 0. Также имеются несколько символов подстановки: 'a' = (признаки + классы)/2, 'i' = признаки, 'o' = классы, 't' = признаки + классы.	hiddenLayers
Скорость обучения. Определяет, насколько обновляются весовые коэффициенты.	learningRate
Моментный параметр. Моментный параметр, который применяется к коэффициентам во время обновления.	momentum
Фильтр из категориального типа в двоичный. Фильтр для преобразования массива данных. Может улучшить качество работы, если массив данных содержит категориальные данные.	nominalToBinaryFilter
Выполнить нормализацию признаков. Включает нормализацию признаков, что может улучшить качество работы нейросети. Категориальные признаки также будут нормализованы (после преобразования их фильтром из категориального типа в двоичный, если этот фильтр включен) так, чтобы значения категориальных признаков лежали в пределах от -1 до 1.	normalizeAttributes

Выполнить нормализацию значений зависимой переменной. Включает нормализацию зависимой переменной, если она является численной. Это может улучшить качество работы нейросети. Нормализация выполняется от -1 до 1. Выходной результат преобразуется обратно к оригинальному масштабу.	normalizeLabelValues
Перезапуск. Если нейронная сеть не сходится к ответу, данная настройка перезапустит процесс обучения с меньшей скоростью обучения. Если нейронная сеть расходится, а перезапуск не разрешен, то процесс обучения завершится неудачей, и будет выведено сообщение об ошибке.	reset
Случайное начальное число. Начальное число, используемое для инициализации генератора случайных чисел. Случайные числа используются для задания первоначальных весовых коэффициентов, а также для перемешивания обучающих данных.	seed
Время обучения. Количество эпох для обучения. Если контрольная выборка не ноль, то обучение может быть прекращено раньше.	trainingTime
Размер контрольной выборки. Процентная доля контрольной выборки. Обучение будет продолжаться до тех пор, пока не будет наблюдаться последовательного ухудшения ошибки на контрольной выборке, либо пока не будет достигнуто установленное время обучения. Если этот параметр установлен на ноль, контрольная выборка использоваться не будет. В этом случае нейросеть будет обучаться заданное количество эпох.	validationSetSize
Порог валидации. Используется для прекращения контрольного тестирования. Значение указывает сколько раз подряд ошибка на контрольной выборке может ухудшаться, пока обучение не будет прекращено.	validationThreshold

НАИВНЫЙ БАЙЕСОВСКИЙ КЛАССИФИКАТОР

Описание поля	Field Name
Использовать оценку ядра. Использовать оценку ядра для численных признаков вместо нормального распределения.	useKernelEstimator
Использовать управляемую дискретизацию. Использовать управляемую дискретизацию для конвертации численных признаков в категориальные.	useSupervisedDiscretization

ОДНОКЛАССОВЫЙ МЕТОД ОПОРНЫХ ВЕКТОРОВ

Описание поля	Имя поля
Не осуществлять замену отсутствующих значений. Определяет, следует ли отключить автоматическую замену отсутствующих значений. Предупреждение: отключайте автозамену только в том случае, если данные не содержат отсутствующих значений.	doNotReplaceMissingValues
Ядро. Ядро, которое будет использовано.	svmKernel
Параметры ядра. Параметры выбранного ядра.	svmKernelParameters
Выполнить нормализацию. Определяет, следует ли выполнять нормализацию данных..	normalize
Ню. Значение параметра ню.	nu
Случайное начальное число. Начальное число, используемое для инициализации генератора случайных чисел.	seed
Сжатие. Определяет, следует ли использовать эвристику сжатия.	shrinking

Параметр допустимого отклонения. Параметр допустимого отклонения для критерия завершения.	toleranceParameter

СЛУЧАЙНЫЙ ЛЕС

Описание поля	Имя поля
Процентное отношение размера подмножества данных. Процентное отношение размера подмножества данных к размеру обучающего набора данных.	bagSizePercent
Разрывать связи в случайном порядке. Разрывать связи в случайном порядке, когда несколько признаков выглядят одинаково значимыми.	breakTiesRandomly
Вычислять "out-of-bag" ошибку. Определяет, следует ли вычислять "out-of-bag" ошибку.	calcOutOfBag
Вычислять значимость признаков. Вычислять значимость признаков посредством уменьшения усредненного коэффициента Джини.	computeAttributeImportance
Максимальная глубина дерева. Максимальная глубина дерева (0, если не ограничена).	maxDepth
Число нитей выполнения. Число нитей выполнения для создания ансамбля.	numExecutionSlots
Количество признаков. Устанавливает количество случайно выбранных признаков (features). Если 0, используется int(log_2(num_predictors) + 1).	numFeatures
Количество итераций. Количество итераций, которые будут выполнены.	numIterations
Отображать статистические данные "out-of-bag". Определяет, следует ли отображать статистические данные по сложности, когда осуществляется оценка "out-of-bag".	outputOutOfBagComplexityStats
Отображать информацию о классификаторах. Определяет, следует ли отображать информацию об отдельных классификаторах в информации об обученном модуле	outputClassifiers
Случайное начальное число. Начальное число, используемое для инициализации генератора случайных чисел**.**	seed

ДЕРЕВО РЕШЕНИЙ С REP

Описание поля	Имя поля
Предварительная оценка. Предварительная оценка для зависимой переменной.	initialCount
Максимальная глубина дерева. Максимальная глубина дерева (-1, если не ограничена).	maxDepth
Минимальное количество экземпляров данных. Минимальный суммарный вес экземпляров данных в листе.	minNum
Минимальная доля от дисперсии. Минимальная доля от дисперсии по всем данным, которая должна присутствовать в узле для осуществления расщепления (только для задач регрессии).	minVarianceProp
Без отсечения ветвей. Определяет, следует ли осуществлять отсечение ветвей.	noPruning
Количество частей данных. Определяет количество данных, используемых для отсечения ветвей. Одна часть данных используется для отсечения ветвей, остальные для создания правил.	numFolds

Случайное начальное число. Начальное число для случайного перемешивания данных.	seed
Распределить предварительную оценку. Распределить предварительную оценку по всем значениям классов вместо того, чтобы использовать заданную предварительную оценку для одного класса.	spreadInitialCount

МЕТОД ОПОРНЫХ ВЕКТОРОВ (SVM)

Описание поля	Имя поля
C. Параметр сложности C.	c
Тип фильтра. Определяет способ преобразования данных.	filterType
Ядро. Ядро, которое будет использовано.	kernel
Параметры ядра. Параметры выбранного ядра.	kernelParameters
Эпсилон. Эпсилон для погрешности округления.	epsilon
Параметр допустимого отклонения. Параметр допустимого отклонения	toleranceParameter
Построить калибровочные модели. Определяет, следует ли подгонять калибровочные модели к результатам метода опорных векторов (для надлежащих оценок вероятности).	buildClibrationModels
Калибратор. Определяет, какой калибровочный метод использовать. Отображается только если установлен параметр **buildClibrationModels**.	calibrator
Параметры калибратора. Параметры калибратора. Отображается только если установлен параметр **buildClibrationModels**.	calibratorParameters
Количество частей данных. Количество частей данных для перекрестной проверки, используемое для создания обучающего набора данных для калибровочной модели (-1 означает использование всего обучающего набора данных). Отображается только если установлен параметр **buildClibrationModels**.	calibNumFolds
Случайное начальное число. Случайное начальное число для перекрестной проверки, используемое для создания обучающего набора данных для калибровочной модели. Отображается только если установлен параметр **buildClibrationModels**.	calibRandomSeed

РЕГРЕССИЯ ОПОРНЫХ ВЕКТОРОВ (SVR)

Описание поля	Имя поля
C. Параметр сложности C.	c
Тип фильтра. Определяет способ преобразования данных.	filterType
Ядро. Ядро, которое будет использовано.	kernel
Параметры ядра. Параметры выбранного ядра.)	kernelParameters
Оптимизатор. Обучающий алгоритм.	regOptimizer
Параметры оптимизатора. Параметры оптимизатора	regOptimizerParameters

НАИВНЫЙ БАЙЕСОВСКИЙ КЛАССИФИКАТОР

Описание поля	Имя поля
Использовать оценку ядра. Использовать оценку ядра для численных признаков вместо нормального распределения.	useKernelEstimator
Использовать управляемую дискретизацию. Использовать управляемую дискретизацию для конвертации численных признаков в категориальные.	useSupervisedDiscretization

СТОХАСТИЧЕСКИЙ ГРАДИЕНТНЫЙ СПУСК

Описание поля	Имя поля
Не осуществлять нормализацию. Определяет, следует ли отключить нормализацию.	doNotNormalize
Не осуществлять замену отсутствующих значений. Определяет, следует ли отключить глобальную замену отсутствующих значений.	doNotReplaceMissingValues
Количество эпох. Количество эпох для обучения (в пакетном режиме обучения). Общее количество итераций равно количеству эпох, умноженному на количество экземпляров данных.	epochs
Лямбда. Коэффициент регуляризации.	lambda
Скорость обучения. Определяет скорость обучения. Если нормализация выключена, то значение скорости обучения должно быть уменьшено (например, установлено на значение 0.0001).	learningRate
Функция потерь. Функция потерь, которая будет оптимизироваться.	lossFunction
Эпсилон. Параметр эпсилон для эпсилон-нечувствительной функции потерь и функции потерь Хьюбера. Ошибка с абсолютным значением меньшим, чем это пороговое значение (эпсилон), даёт ноль для эпсилон-нечувствительной функции потерь. Для функции потерь Хьюбера эпсилон - это граница между квадратичной и линейной частями функции потерь.	epsilon
Случайное начальное число. Начальное число, используемое для инициализации генератора случайных чисел.	seed

АЛГОРИТМ С ПРЕДВАРИТЕЛЬНОЙ ФИЛЬТРАЦИЕЙ

Описание поля	Имя поля
Алгоритм. Базовый алгоритм, который будет использоваться.	algorithm
Гиперпараметры базового алгоритма. Определяет параметры выбранного алгоритма.	baseAlgorithmParameters
Фильтр. Фильтр, который будет использоваться.	filter
Параметры фильтра. Определяет параметры выбранного фильтра.	filterParameters
Случайное начальное число. Начальное число, используемое для инициализации генератора случайных чисел.	seed

ДЕРЕВО ХЁФДИНГА

Описание поля	Имя поля
Грейс-период. Количество экземпляров данных (или суммарный вес экземпляров данных), которые должны быть "увидены" листом между попытками расщепления.	gracePeriod

Порог Хёфдинга. Порог, ниже которого расщепление будет разрывать связи.	hoeffdingTieThreshold
Принцип предсказания. Определяет, какой принцип предсказания будет использован.	leafPredictionStrategy
Порог для предсказания наивным байесовским классификатором. Количество экземпляров данных (вес), которые должен "увидеть" лист до того, как (адаптивному) наивному байесовскому классификатору будет позволено делать предсказания.	naiveBayesPredictionThreshold
Отображать модели по листам. Определяет, следует ли отображать информацию о моделях по листам в информации об обученном модуле (применимо только к "наивным байесовским" листам).	outputLeafModels
Допустимая ошибка при расщеплении. Допустимая ошибка при принятии решения о расщеплении. Чем ближе это значение к нулю, тем больше времени занимает принятие решения.	splitConfidence
Критерий расщепления. Определяет, какой критерий расщепления будет использован.	splitCriterion
Минимальная доля веса по информационному выигрышу. Минимальная доля веса, требуемая по меньшей мере двум ветвям для расщепления по информационному выигрышу.	minimumFractionOfWeightInfoGain

МНОГОКЛАССОВЫЙ ДООБУЧАЕМЫЙ КЛАССИФИКАТОР

Описание поля	Имя поля
Базовый алгоритм. Базовый алгоритм, который будет использоваться.	baseAlgorithm
Гиперпараметры базового алгоритма. Определяет параметры выбранного алгоритма.	baseAlgorithmParameters
Метод. Определяет метод, который будет использоваться для приведения мультиклассовой задачи к нескольким бинарным.	method
Декодирование логарифмической функции потерь. Определяет, применять ли декодирование логарифмической функции потерь для случайных и исчепывающих кодов.	logLossDecoding
Множитель ширины. Устанавливает множитель ширины при использовании случайных кодов. Количество кодов равно этому числу, умноженному на количество классов.	randomWidthFactor
Использовать попарное сопряжение. Определяет, следует ли использовать попарное сопряжение.	usePairwiseCoupling
Случайное начальное число. Начальное число, используемое для инициализации генератора случайных чисел.	seed

Параметры ядра

Параметры, общие для всех ядер:

Описание поля	Имя поля
Размер кэша. Размер кэша (простое число), 0 для полного кэша, -1 для того, чтобы отключить его.	kernelCacheSize

Параметры, специфичные для ядра:

РАСПРЕДЕЛЕНИЕ ПИРСОНА ТИПА VII

Описание поля	Имя поля
Омега. Значение омега.	kernelOmega
Сигма. Значение сигма.	kernelSigma

ПОЛИНОМИАЛЬНЫЕ И НОРМАЛИЗОВАННЫЕ ПОЛИНОМИАЛЬНЫЕ ЯДРА

Описание поля	Имя поля
Показатель степени. Значение показателя степени.	kernelExponent
Использовать младший разряд. Определяет, использовать ли младший разряд.	kernelUseLowerOrder

РАДИАЛЬНАЯ БАЗИСНАЯ ФУНКЦИЯ

Описание поля	Имя поля
Гамма. Значение гамма.	kernelGamma

Параметры ядра, используемые алгоритмом одноклассового метода опорных векторов

Параметры, общие для всех ядер:

Описание поля	Имя поля
Размер кэша. Размер кэша в МБ.	kernelSvmCacheSize

ПОЛИНОМИАЛЬНОЕ ЯДРО

Описание поля	Имя поля
Коэффициент0. Независимый коэффициент функции ядра.	kernelSvmCoefficient0
Показатель степени. Значение показателя степени.	kernelSvmDegree
Гамма. Коэффициент гамма. Если 0, то используется значение 1/max_index.	kernelSvmGamma

РАДИАЛЬНАЯ БАЗИСНАЯ ФУНКЦИЯ

Описание поля	Имя поля
Gamma. Коэффициент гамма. Если 0, то используется значение 1/max_index.	kernelSvmGamma

СИГМОИДА

Описание поля	Имя поля
Коэффициент0. Независимый коэффициент функции ядра.	kernelSvmCoefficient0

Гамма. Коэффициент гамма.	kernelSvmGamma

Параметры оптимизатора

Параметры, общие для всех оптимизаторов:

Описание поля	Имя поля
Эпсилон. Эпсилон для погрешности округления.	epsilon
Параметр эпсилон. Параметр эпсилон для эпсилон-нечувствительной функции потерь.	epsilonParameter
Случайное начальное число. Начальное число, используемое для инициализации генератора случайных чисел.	seed

ПАРАМЕТРЫ УЛУЧШЕННОГО REGSMO

Описание поля	Имя поля
Допустимое отклонение. Параметр допустимого отклонения (tolerance parameter), используемый для проверки критерия остановки (b_up меньше, чем b_low + 2*tol).	tolerance
Использовать вариант 1. Определяет, использовать первый вариант из статьи, указанной ниже, либо второй вариант. S.K. Shevade, S.S. Keerthi, C. Bhattacharyya, K.R.K. Murthy: Improvements to the SMO Algorithm for SVM Regression. In: IEEE Transactions on Neural Networks, 1999	useVariant1

12.13.4.3 Внутренние фильтры

Количество фильтров, поддерживаемых модулем машинного обучения AtomMind, постоянно увеличивается. Текущий набор фильтров представлен ниже.

Фильтр	Описание
Заместить отсутствующие значения	Замещает числовые значения соответствующими средними значениями по атрибуту (для численных признаков) или модами по атрибуту (для категориальных признаков). Если требуется, может игнорировать отсутствующие значения в колонке с зависимой переменной.
Удалить строки с отсутствующими значениями	Удаляет экземпляры данных, содержащие как минимум одно отсутствующее значение. Отсутствующие значения в колонке с зависимой переменной игнорируются.

Следует отметить, что фильтры (даже одного типа) могут добавляться неоднократно и комбинироваться по необходимости.

Некоторые фильтры имеют параметры. Для более подробной информации см. Параметры внутренних фильтров 878.

12.13.4.4 Параметры внутренних фильтров

Параметры фильтра Заместить отсутствующие значения

Описание поля	Имя поля
Игнорировать поле с зависимой переменной. Если установлена эта настройка, индекс поля с зависимой переменной будет временно сброшен перед применением фильтра.	ignoreLabelField

12.14 Скрипты

Интегрированные языки выражений 112 и запросов 829 AtomMind помогают в решении большинства повседневных задач по обработке данных. Однако иногда этого недостаточно. Бывает, становятся необходимыми возможности полноценного языка программмирования с циклами, переменными, объявлениями методов и пр. С этой целью, AtomMind Server может обрабатывать скрипты, написанные на Java и R. Эти скрипты выполняются внутри виртуальной машины Java (JVM) сервера, разрешая им доступ ко всем внутренним данным сервера в добавок к входным параметрам самого скрипта. Таким образом, скрипты - это очень мощный инструмент, дающий возможность полностью контролировать сервер в режиме реального времени.

Скрипты можно вызывать из выражений, что делает их доступными для:

- Активирования тревог 779
- Фильтрации событий 762
- Запросов 829
- Обработки данных настраиваемых форм и интерфейсов 943
- Подготовки исходных данных для отчета 928

Существует два способа запустить скрипт:

- Выполнение вручную путем вызова действия выполнить 1591 контекста Скрипт 1591. Входная таблица данных 49 функции (которая может быть любого формата), будет передана прямо в скрипт.
- Автоматическое выполнение при запуске %ls%>. Выполнение скрипта происходит автоматически при запуске, если включен флажок **Запускать автоматически при включении сервера** в свойствах 880 скрипта. В этом случае, входные данные не передаются в скрипт.

Скрипт может вернуть некоторые данные в форме объекта таблица данных 49. Данная таблица возвращается функцией выполнить 1593 контекста Скрипт. Если скрипт возвращает `NULL`, функция **выполнить** возвращает таблицу из одной ячейки с пустым (nullable) строковым полем.

Как и когда писать скрипты

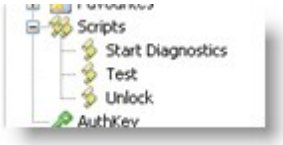
Скрипты имеют неограниченные возможности по использованию сред разработки и доступу к ресурсам сервера. Однако разработка скриптов, имеющих доступ к машинным ресурсам нижнего уровня или недокументированным классам AtomMind, часто могут нарушать работоспособность и производительность системы.

Следующие правила позволяют написать хороший скрипт:

- Лучший скрипт - тот, который берет входные данные и использует математическую/логическую обработку для преобразования их в выходные данные, не затрагивая и не влияя на среду выполнения.
- Если необходим доступ к окружению, его следует осуществлять через формализованные программные интерфейсы и классы унифицированной модели данных AtomMind. К этим интерфейсам относятся `Context`, `ContextManager`, `DataTable`, и др. Для большей информации см. главу Разработка 1339.
- У скриптов не должно быть доступа к сокетам и они не должны выполнять никакого сетевого взаимодействия. Сетевые операции, используемые для получения внешних данных или для показа локальных данных внешним системам, должны выполняться драйверами устройств 518.
- Скрипты не должны иметь прямого доступа к файловой системе сервера. Это можно осуществить через драйверы устройств Файл 576 и Папка 577.
- Скрипты не должны создавать потоки и запускать сторонние процессы. Многопоточность осуществляется должным образом во всех "активных" объектах системы (таких как модели 810), которые могут вызвать ваш скрипт как обратный вызов.
- Хороший скрипт должен иметь ограниченное и предсказуемое время выполнения. Он не должен надолго зависать ни при каких обстоятельствах.
- Если скрипт вызывает методы сторонних библиотек, они должны отвечать вышеперечисленным правилам (иметь минимум взаимодействия со средой, не иметь доступа к ресурсам системы и т.д.)

Администрирование скриптов

Для администрирования скриптов используются два контекста 41: общий контекст Скрипты 1590, который служит в качестве контейнера, и контекст Скрипт 1591, который содержит информацию об определенном скрипте.



Каждый пользователь 478 имеет свой набор скриптов.

Встроенные скрипты

В настоящее время всего один скрипт входит в ядро дистрибутива AtomMind Server. Этот скрипт называется **Расчет даты нарушения SLA** и описан в руководстве Прогнозирование нарушений SLA 1705.

12.14.1 Конфигурация скрипта

Каждый скрипт характеризуется несколькими свойствами:

Описание поля	Имя поля
Имя скрипта. Имя контекста скрипта 1591, необходимое для ссылки на данный скрипт из других частей системы. Должно удовлетворять соглашениям по наименованию 42 контекстов.	name
Описание скрипта. Текстовое описание скрипта. Также является описанием 43 контекста "Скрипт".	description
Тип языка. Тип языка программирования скрипта.	scriptLanguage
Текст скрипта. Источник скрипта на языке программирования Java.	text
Текст скрипта R. Источник скрипта на языке программирования R.	rText
Запускать автоматически при запуске сервера. Флажок, согласно которому сервер запускает скрипт при загрузке.	autorun

Данные свойства доступны через переменную childInfo 1471.

12.14.2 Java скрипты

Скрипты написаны полностью на языке Java, поэтому могут иметь доступ ко всем ресурсам сервера, на котором запущен AtomMind Server, включая:

- Любые объекты в оперативной памяти
- Файловую систему
- Сетевой и последовательный интерфейсы ввода/вывода
- Консоли, журналирование и даже графический интерфейс пользователя
- Многопоточную обработку

Скрипты выполняются в среде JVM, и их права доступа никак не ограничены. Обратитесь к разделу безопасность скриптов 885 для получения подробной информации.

Интерфейс скрипта

Скрипты написаны на языке Java. Каждый скрипт - это отдельный Java класс, который должен реализовать интерфейс скрипта:

```
public interface Script
{
  public DataTable execute(ScriptExecutionEnvironment environment, DataTable parameters) throws Sc
}
```

Этот интерфейс объявляет единственный метод `execute()`, который вызывается AtomMind Server при выполнении скрипта.

Среда выполнения

У каждого скрипта есть доступ к объекту, описывающему интерфейс `ScriptExecutionEnvironment`, который передается в качестве аргумента методу **execute()**. Ниже описан интерфейс `ScriptExecutionEnvironment`:

```
public interface ScriptExecutionEnvironment
{
  public abstract CallerController getCallerController();

  public abstract Context getScriptContext();
}
```

Экземпляр `ScriptExecutionEnvironment` предоставляет доступ к объекту, описывающему интерфейс `CallerController` (он получен в результате вызова метода `getCallerController()`). Этот объект включает в себя права доступа пользователя, который инициировал выполнение скрипта. Объект `CallerController` передается в качестве аргумента большинству контекстных операций (Get Context, Get/Set Variable, Call Function, Add/Remove Event Listener и т.д.)

Когда скрипт запускается автоматически при запуске сервера, его среда выполнения включает в себя систему `CallerController`, которая подавляет все проверки прав доступа.

`Context`, возвращаемый методом `getScriptContext()`, позволяет получить доступ к дереву контекстов и всем его объектам.

Разработка скриптов

Изучите руководство по программированию, чтобы получить общую информацию о разработке скриптов AtomMind.

Шаблон скрипта

Когда создается новый скрипт, он содержит автоматически сгенерированную заглушку класса реализации интерфейса **Скрипта** с пустым методом `execute()`. Ниже представлен текст скрипта по умолчанию:

```
import com.tibbo.aggregate.common.context.*;
import com.tibbo.aggregate.common.datatable.*;
import com.tibbo.aggregate.common.script.*;

import com.tibbo.linkserver.*;
import com.tibbo.linkserver.context.*;
import com.tibbo.linkserver.script.*;

public class %ScriptClassNamePattern% implements Script
{
  public DataTable execute(ScriptExecutionEnvironment environment, DataTable parameters) throws Sc
  {
  }
}
```

Обратите внимание, что `%ScriptClassNamePattern%` будет заменен автоматически сгенерированным именем Java класса во время компиляции. Не изменяйте эту часть скрипта, чтобы избежать ошибок компиляции.

После того, как вы добавили необходимый код внутрь тела метода `execute()`, вы можете попробовать выполнить скрипт. Обратите внимание, что скрипт перекомпилируется при каждом выполнении. Система сообщает пользователю о любых ошибках компиляции. Используйте описание действия Выполнить для дополнительной информации.

Пример скрипта

Пример Java скрипта, который меняет IP адрес аппаратного Device Server:

```
import com.tibbo.aggregate.common.context.*;
```

```
import com.tibbo.aggregate.common.datatable.*;
import com.tibbo.aggregate.common.script.*;

import com.tibbo.linkserver.*;
import com.tibbo.linkserver.context.*;
import com.tibbo.linkserver.script.*;

public class %ScriptClassNamePattern% implements Script
{
  public DataTable execute(ScriptExecutionEnvironment environment, DataTable parameters) throws Sc
  {
       try
       {
              // Getting context of a Device Server
              Context con = environment.getScriptContext().getContextManager().get("users.admin.de

              if (con == null)
              {
                     throw new ScriptException("Device Server not available");
              }

              // Getting IP address variable
              DataTable val = con.getVariable("IP_setting", environment.getCallerController());
              // Changing IP address
              val.rec().setValue("IP_setting", "192.168.1.235");
              // Writing new value of variable
              con.setVariable("IP_setting", environment.getCallerController(), val);
              // Rebooting Device Server
              con.callFunction("reboot", environment.getCallerController());
              return null;
       }
       catch (ScriptException ex)
       {
              throw ex;
       }
       catch (Exception ex)
       {
              throw new ScriptException(ex);
       }
  }
}
```

12.14.3 Скрипты на языке R

R - это язык программирования с открытым исходным кодом и программная среда для статистического вычисления, которые поддерживаются организацией R Foundation для статистического вычисления. R - это язык интерпретации. Интерпретатор выполняет программу напрямую, переводя каждое выражение в последовательность одной или более подпрограмм, которые уже скомпилированы в машинный код.

Интерфейс скрипта

Скрипты написаны на языке R. Каждый скрипт должен содержать:

```
dataSetMatrixInput
dataSetMatrixOutput
```

где **dataSetMatrixInput** - это R-матрица, конвертируемая из входящей таблицы данных 49.

- Все числовые типы данных AtomMind (целочисленные, двойные или длинные целочисленные) и даты конвертируются в число двойной точности R.
- Все строчные и цветовые типы AtomMind конвертируются в символьный тип R.
- Каждая вложенная таблица данных будет конвертироваться во вложенную R матрицу.

Во время трансформации таблицы данных в R-матрицу все названия столбцов остаются прежними. В R-матрице количество рядов и столбцов соответствует количеству записей и полей таблиц данных.

Результат выполнения должен быть передан в **dataSetMatrixOutput.**

Использование некоторых R-библиотек может стать причиной непредсказуемого поведения системы при выполнении скрипта.

Шаблон скрипта

Когда создается новый скрипт, он содержит автоматически сгенерированный код с двумя переменными.

```
dataSetMatrixInput #Input Parameters (Matrix)

dataSetMatrixOutput = dataSetMatrixInput #Output Parameters (Matrix)
```

Чтобы выполнить скрипты на языке R, ваша ОС должна иметь переменную среды "R_HOME". В ОС Windows должна быть настроена переменная "PATH", указывающая путь к выбранной версии бинарных файлов библиотеки R.

Руководство по интеграции R и Linux

Следуйте данной инструкции для скачивания и установки R. Для связи между AtomMind R, необходима установка дополнительных библиотек. Используйте данную инструкцию для установки и подтовки rJava.

Для конфигурирования R перед установкой, используйте команду `R CMD javareconf`.

Скопируйте файл "libjri.so" из "$R_HOME/library/jri/" в директорию "lib" AtomMind Server.

Пример скрипта

Пример скрипта на языке R:

```
dataSetMatrixInput #Input Parameters (Matrix)
dp = double(5)
dp[1] = 1
dp[2] = 2
dp[3] = 3
dp[4] = 4
dp[5] = 5
strs = c("str1", "str2", "str3")
rMatrix = matrix(list(),nrow = 1,ncol = 2)
rMatrix[[1,1]] = dp
rMatrix[[1,2]] = strs
colnames(rMatrix) = c("numericField","characterField")
df = as.data.frame(rMatrix)
dataSetMatrixOutput = rMatrix #Output Parameters (Matrix)
```

12.14.4 Скрипты на языке Python

Python - универсальный интерпретируемый язык программирования высокого уровня. Python отличается системой динамической типизации и автоматическим управлением памятью. Язык поддерживает множество парадигм программирования, включая объектно-ориентированную, императивную, функциональную и процедурную, а также имеет обширную и универсальную стандартную библиотеку.

Руководство по интеграции Python

Для загрузки и установки Python в вашу систему воспользуйтесь следующей инструкцией. Рекомендуется версия 3.6.X.

Скопируйте папку "jep" из директории "lib" AtomMind Server в папку Python "site-packages", если скрипт на языке Python будет использоваться только для текущего пользователя.

Чтобы найти местоположение директории "site-packages", используйте команду `python -m site --user-site`.

Скопируйте папку "jep" из директории "lib" AtomMind Server в папку Python "dist-packages", если скрипт на языке Python будет использоваться для всех пользователей.

Чтобы найти местоположение директории "dist-packages", используйте команду: `python -m site`.

Интерфейс скрипта

Скрипты пишутся на языке Python. Каждый скрипт должен содержать:

```
dataSetDataFrameInput
dataSetDataFrameOutput
```

где **dataSetMatrixInput** - это pandas.DataFrame, конвертируемый из входящей таблицы данных 49.

- Все числовые типы данных AtomMind (целочисленные, двойные или длинные целочисленные) и даты конвертируются в число двойной точности Python.
- Все строчные и цветовые типы AtomMind конвертируются в символьный тип Python.
- Каждая вложенная таблица данных будет конвертироваться во вложенный pandas.DataFrame.

Во время трансформации таблицы данных в pandas.DataFrame все названия столбцов остаются прежними. В pandas.DataFrame количество рядов и столбцов соответствует количеству записей и полей таблиц данных.

Результат выполнения должен быть передан в **dataSetDataFrameOutput.**

Библиотека "pandas" не включена в стандартный набор библиотек Python. Необходима отдельная установка "pandas".

Шаблон скрипта

Когда создается новый скрипт, он содержит автоматически сгенерированный код с двумя переменными:

```
dataSetDataFrameInput #Input Parameters (pandas.DataFrame)

dataSetDataFrameOutput #Output Parameters (pandas.DataFrame)
```

Чтобы выполнить скрипты на языке Python, ваша ОС должна иметь переменные среды Python.

Пример скрипта

Пример скрипта на языке Python:

```
import pandas
dataSetDataFrameInput #scrScriptPyDefaultDataFrame
pythonList = list()
pythonList.append(1)
pythonList.append(2)
pythonList.append(3)

innerStrList = list()
innerStrList.append("newStr")
innerStrList.append("someStr")
innerStrList.append("superStr")
pythonList.append(innerStrList)

df = pandas.DataFrame(pythonList)

dataSetDataFrameOutput = df #scrScriptPyDefaultDataFrame
```

12.14.5 Использование скриптов в выражениях

Вызвать скрипт из выражения 112 можно, вставив ссылку 117 в функцию выполнить 1591 контекста "Скрипт". Параметры функции, определенные 120 в ссылке, будут переданы в скрипт. Функция **Выполнить скрипт** возвращает значение, на которое можно ссылаться из других частей выражения.

Пример. Далее следует выражение, ссылающееся на скрипт **calculator** и передающее в него некоторое значение ввода:

```
{users.admin.scripts.calculator:execute("123", "456", "789")}
```

Скрипт получит три строки "123", "456" и "789" в качестве входа (для более подробной информации см. определение входных параметров функции в ссылке 120). Он может сконвертировать данные строки в числа, выполнить по ним необходимые вычисления и вернуть результат.

12.14.6 Безопасность скриптов

Скрипты выполняются на сервере виртуальной машины Java и их права доступа никак не ограничены. Таким образом, одна ненамеренная ошибка в скрипте или некий вредоносный код могут вызвать неправильное функционирование сервера, зависание, 100% время загрузки процессора, повреждение его базы данных или данных на рабочей машине. По умолчанию только администратор по умолчанию 479 может изменять и выполнять скрипты. Обеспечьте правом изменять и выполнять скрипты только пользователей с полномочиями высшего уровня.

Если скрипт можно заменить выражением 112, всегда рекомендуется осуществить эту замену, поскольку механизм безопасности выражения 141 позволяет всем операциям выражения наследовать права доступа вызывающей стороны.

Выполнение скрипта

Чтобы другой пользователь мог выполнить скрипт, присвойте ему уровень прав доступа 486 Администратора в контексте Скрипты 1590. Это позволит пользователю выполнять все скрипты в этом контексте.

Чтобы разрешить выполнение одного скрипта, присвойте уровень прав доступа Администратора в контексте этого Скрипта 1591.

Изменение скрипта

Только пользователи с уровнем прав доступа 486 *Администратора* в контексте "Скрипт" могут изменять конфигурацию и код скрипта. Это обеспечивает максимальную безопасность для операций со скриптами.

12.15 Классы

Классы созданы для хранения большого числа сходных объектов в реляционной или графической базе данных. Каждый экземпляр класса занимает одну запись таблицы базы данных или узел в графической БД. Классы используют заданные пользователем поля 887, которые напрямую преобразуются в колонки таблицы базы данных, содержащей экземпляры определенного класса.

Между собой классы могут иметь отношения "один к одному", "один ко многим" и "многие ко многим". Эти отношения помогают установить умную навигацию между экемплярами класса. Отношения экземпляров классов сопоставляются с внешними ключами таблиц значений в реляционной базе данных и в связях графической базы данных.

Инструментальные панели 912 принимают классы к сведению, позволяя обрабатывать и изменять поля отдельных экземпляров, также как отображать табличный список экземпляров с оперативной сортировкой, поиском и фильтрацией.

Экземпляры классов могут иметь конфигурируемые состояния и жизненные циклы, также как и конфигурируемые разрешения для каждого экземпляра.

Несколько примеров использования классов:

- В системе CRM, типичные классы - это Контактное лицо, Клиент, Потенциальный клиент и Поненциальная сделка
- В системе инвентаризации активов классами могут быть Актив, Тип актива, Субъект и Местоположение
- Система службы технической поддержки может иметь такие классы как Агент, Инцидент, Сервис и Соглашение о качестве предоставляемых услуг (Service Level Agreement, SLA)
- Система управления информацией центра обработки и хранения данных обычно имеет дело с Корпусами, Серверами, Кабелями и подобными специфическими для области промышленности классами

Экземпляры класса обычно хранятся в базе данных 692 AtomMind Server, но только реляционные базы данных 697 могут использоваться для их хранения. Также возможно извлечения данных из сторонних систем с помощью драйвера базы данных 647.

Администрирование классов

Для администрирования классов используется два контекста: первый - это общий контекст классов 1471, который служит как хранилище. Второй - это контекст класса 1473, которые содержит информацию для одного типа классов (и всех его экземпляров).

Классы не работают со связанным хранилищем NoSQL в момент использованя нативной функциональности движков подсистемы хранения.

Каждый пользователь 478 имеет свой набор классов.

общие настройки

Для настройки плагина Классы используются Общие настройки:

Свойство	Описание
Контекст хранилища по умолчанию	Контекст указывает на классы, лежащие в основе хранилища.

Управление экземплярами класса

Экземпляры класса управляются с помощью инструментальных панелей 912. Стандартные (**абсолютные** и **относительные**) инструментальные панели могут действовать как "ворота в мир классов". Они содержат списки экземпляров класса, которые отформатированы, отфильтрованы и отсортированы согласно заданным правилам. Клик на экземпляр в любом из данных списков классов обычно приводит системного оператора к инструментальной панели **экземпляра класса,** которая устанавливает поля выбранного экземпляра, также как и списки других экземпляров классов, соединенных с ним с помощью различных отношений 888. Обратитесь к разделу типы инструментальных панелей 913 для более детальной информации.

УПРАВЛЕНИЕ СХЕМОЙ БАЗЫ ДАННЫХ

Создание, обновление и удаление таблиц базы данных 692, которые содержат экземпляры класса, являются полностью автоматическими. Как только создан новый класс, для его экземпляров создается новая таблица базы данных. Имя таблицы совпадает с именем класса. Таблица обновляется каждый раз, когда изменяются поля 887 класса жизненных циклов 888. Наконец, таблица удаляется, если сам класс удален.

Также каждое отношение многие ко многим 892 преобразовывается в выделенную таблицу данных. Создание, управление и удаление таблицы происходит атоматически.

Создание операторского интерфейса на основе классов

Классы - это очень мощные объекты, которые помогают в построении полнофункциональных приложений на платформе AtomMind. Вот список основных шагов, требуемых для построения таких приложений:

1) Создайте полную структуру класса для вашего приложения

2) Создайте ваши классы и определите их поля 887. Поля класса могут включать ссылки на другие классы (поля типа Длинное 51, которые использует Редактор экземпляров класса 52), прикрепленные файлы (поля типа Данные 52). Даннные, которые используют редакторы файлов, изображений или звука), и любые другие типичные поля, такие как Целое, Строка, Дата, Таблица данных, и т.д.

3) Настройте отношения[888] между классами, либо создав поля типа Длинное[51], которые используют Редактор экземпляров класса[52] для конфигурации ссылок "один ко многим", либо добавления отношений "многие ко многим"[892]

4) Настройте жизненные циклы[888] экземпляров класса

5) Настройте просмотры[888] для ваших классов для активации повторного использования наборов полей экземпляров, правил фильтрации и сортировки в различных инстументальных панелях

6) На одной из главных инструментальных панелей[912] вашего операторского интерфейса, добавьте несколько элементов типа списка экземпляров класса для активации навигации в мире ваших классов

7) Создайте специализированную инструментальную панель экземпляра класса[913] для каждого из ваших классов. Клик по экземпляру класса в любом списке экземпляров приведет оператора к инструментальной панели экземпляров класса, валидной для выбранного класса.

8) На ваших инструментальных панелях экземпляра класса:

   a) Добавьте элементы полей экземпляров класса для отображения атрибутов экземпляра класса.

   b) Добавьте элементы списка экземпляров класса для отображения списка других экземпляров классов, которые относятся к текущему экземпляру. Настройте поле Отношение в параметрах списка экземпляров класса для отфильтровывания только соответствующих экземпляров.

   c) Добавьте элементы журнала событий для показа событий создания/изменения/удаления экземпляров класса и комментариев экземпляров класса (Class Instance Commented events). Обратитесь к разделу Контекст класса[1473] для более детальной информации о данных событиях.

   d) Добавьте элементы таблицы данных, которые покажут разрешенные переходы состояний для ваших классов. Обратитесь к разделу жизненные циклы[888] для более детальной информации.

   e) Добавьте любые другие элементы инструментальной панели (виджеты, таблицы даннных и т.д.)

Как только сконфигурированы инструментальные панели верхнего уровня и инструментальные панели экземпляров класса, ваши операторы могут начать работу с приложением и управление вашими классами.

Наследование классов

Каждый контекст Класса предоставляет контейнер Подкласса. Можно создать контекст Класса внутри контейнера Подкласса. По умолчанию, такой созданный вами контекст будет иметь те же `Fields`, `Many to Many Relations` и `Lifecycles` переменные, что и его родительский контекст. Таким образом, вы можете использовать родительский Класс в качестве шаблона для создания Классов с аналогичной структурой.

Наследование классов можно использовать во многих случаях, например, при создании объектов CMDB. Это дает вам Конфигурационную единицу со следующими параметрами: **Name**, **Location** и **Status**. Вы можете создать контекст дочернего **IP устройства** и добавить другие поля к его свойствам, например: **IP address**, **Subnet Mask**, **Manufacturer**. Таким образом, вы можете создать контексты **IP устройств** с набором параметров: **Name**, **Location**, **Status**, **IP address**, **Subnet Mask**, **Manufacturer**.

12.15.1 Поля

Каждый класс имеет *поля*, которые определяют параметры (атрибуты) каждого экземпляра класса. Так как экземпляры класса хранятся в специальной таблице базы данных, каждое поле определяет одну колонку в данной таблице.

Свойства поля схожи со свойствами поля таблицы данных[51]. Поля класса управляются с помощью таблицы Поля[891].

Заданные поля

Все классы имеют несколько заданных полей, которые могут, однако, быть изменены или удалены. Вот список полей по умолчанию:

Имя поля	Тип поля	Описание поля
instance_id	Длинное	Это поле содержит автоматически сгенерированный ID экземпляра класса. Как только поле с этим именем преобразовано в базу данных, используется принцип автоматической генерации идентификатора рядов. ID экземпляра никогда не должен изменяться с момента первого создания экземпляра класса.
author	Строка	Имя пользователя[478] AtomMind Server, создавшего экземпляр.
creation_time	Дата	Дата/время создания экземпляра.

update_time	Дата	Дата/время последнего изменения экземпляра. Изменение может быть изменением значения поля, переходом жизненного цикла, и т.д.

После добавления полей пользователей необходимо обновить параметры 52 элементов инструментальной панели 918, использующие отредактированный класс.

12.15.2 Просмотры

Просмотры класса используются для просмотра и управления экземплярами класса. Каждый просмотр определяет:

- Имя и описание просмотра
- Набор колонок (полей класса) показан, когда используется просмотр
- Набор ограничения фильтров показанных экземпляров
- Настройки сортировки экземпляров
- Привязки полей классов

12.15.3 Связи

Отношения используются для соединения экземпляров различных классов друг с другом. Это необходимо для любого вида интеллектуального анализа данных и систем управления документами/объектами.

Отношения помогают наладить навигацию между различными списками экземпляров, инструментальными панелями и другими компонентами пользовательского интерфейса.

Доступны следующие типы отношений:

Один ко многим

В этом случае один экземпляр класса А связан со многими экземплярами класса Б. Технически, в этом случае класс Б определяет поле типа Дннное 52, которое содержит ID экземпляра класса А. Имя поля и описание составляют *имя отношения.* Это поле использует редактор экземпляров внешнего класса 52, который позволяет выбрать экземпляр класса А, если редактируется экземпляр класса Б. Инструментальная панель 912 экземпляра класса А может включать список связанных экземпляров класса Б.

Информация о связи один ко многим от класса А к классу Б хранится в специальном поле таблицы базы данных, которая хранит экземпляры класса Б.

Многие ко многим

В этом случае каждый экземпляр класса А связан со многими экземплярами класса Б, но в то же время, каждый экземпляр класса Б может быть связан со многими экземплярами класса А внутри этого отношения. Имя отношения и параметры определены в таблице отношений многие ко многим 892.

Если определены отношения многие ко многим, инструментальная панель экземпляров для класса А может содержать список связанных экземпляров класса Б и наоборот.

Отдельная таблица базы данных создана, чтобы содержать связи, определенные отношением многие ко многим.

12.15.4 Жизненные циклы

Каждый класс может иметь несколько *жизненных циклов*. Жизненный цикл - это то, что назначает *состояние* каждого экземпляра класса и позволяет экземпляру изменять свое состояние с помощью п*ереходов между состояниями*. Например, класс Инцидент может иметь жизненный цикл Вычисление, который имеет такие состояния как Открытый, В процессе и Закрытый.

Несмотря на то, что большинство классов не имеют жизненных циклов, или имеют только один, возможно настраивать несколько жизненных циклов для каждого класса. Они остаются полностью независимыми друг от друга.

Параметры конфигурации жизненного цикла описаны в разделе свойств жизненного цикла 892. Каждый жизненный цикл определен таблицей состояний и таблицей переходов между состояниями. Минимальная конфигурация жизненного цикла предполагает:

- Добавление по крайней мере двух состояний

- Добавление позволенных переходов между этими состояниями

Как только был сконфигурирован жизненный цикл, возможно расширение инструментальной панели[912] экземпляра класса с помощью специального компонента, который отображает набор кнопок, переключающий экземпляр в новое состояние. Этот компонент покажет специальную кнопку для каждого перехода, который разрешен на данный момент согласно текущему состоянию, таблице переходов, условиям выхода и входа для текущих и новых состояний.

Добавление панели переходов между состояниями к инструментальной панели экземпляров класса

Для добавления набора кнопок переходов между состояниями на панель инструментов экземпляра класса, откройте элементы инструментальной панели и добавьте элемент **таблицы данных**. В свойствах элемента установите выражение на

```
callFunction({env/storageContext}, "currentTransitions",class1 ",cycle1 ", {data},
dc(), {env/storageInstanceId})
```

где

class1 - это имя класса, чей переход между состояниями будет показан

cycle1 - это имя жизненного цикла

12.15.5 Фильтры экземпляров

Фильтры экземпляров класса определяют, какие экземпляры показаны в списках различных экземпляров класса. Каждый фильтр содержит одно или более правил, которые применяются к полям класса для выяснения, какие экземпляры должны фильтроваться и отображаться в списке экземпляров.

Каждый фильтр определяется набором правил, которые включают:

- **Логическая операция**. Доступные опции - это И (AND) и ИЛИ (OR). Если выбрано И, это правило и предыдущие правила должны быть верными. Если выбрано ИЛИ, либо это правило, либо предыдущие правила должны быть верными. Логическая операция не доступна для первого правила в наборе правил.
- **Тип**. Условие или вложенные условия. Если выбрано условие, правило обрабатывается согласно **Колонке**, **Операции** и **Значению**, выбранными в нем. Если выбраны вложенные условия, результат правила равняется совокупному результату набора вложенных правил, определенного таблицей **вложенных условий**.
- **Колонка**. Определяет, какое поле класса или колонка базы данных проверена правилом.
- **Операция**. Определяет, какая операция применяется для проверки значения колонок, т.е. Равен нулю или Больше чем. Доступные операции зависят от типа **колонки**.
- **Значение**. Значение, по которому проверяется колонка. Все значения определены как строки.
- **Вложенные условия**. Определяет правила подфильтра. Вложенный фильтр - это таблица, с теми же полями, что и таблица правил фильтра высшего уровня. Уровень вложенности не ограничен.

12.15.6 Привязки

Привязки[735] экземпляра класса позволяют изменять выбранные поля экземпляра класса при вызове функции класса **Обработка привязок**.

Фильтр экземпляра, выступая аргументом данной функции, определяет набор экземпляров для обработки.

Важно не путать привязки экземпляра класса с привязками поля просмотра[888] класса:

- Привязки экземпляра класса выполняют постоянную модификацию экземпляров класса, хранящихся в БД
- Привязки поля просмотра класса вычисляют значения вычислимых колонок просмотра класса при активации просмотра, т.е. они влияют только на способ представления данных

12.15.7 Конфигурация классов

Этот раздел содержит информацию о свойствах конфигурации класса.

Все свойства, описанные здесь, доступны с помощью действия конфигурировать[105] контекста Класс[1473].

12.15.7.1 Свойства класса

Этот раздел определяет основные опции для класса.

Описание поля	Имя поля
Имя. Имя класса. Оно должно удовлетворять согласшениям о наименовании 42 контекстов. Имя необходимо для ссылки на данный класс и его экземпляры из других частей системы. Изменение имени класса не разрешено.	name
Описание. Удобное для чтения описание 43 класса.	description
Выражение именования. Выражение, которое используется для вычисления удобных для чтения описаний отдельных экземпляров класса. Среда вычисления 114 выражения именования: Контекст по умолчанию 119 — Контекст класса. Таблица данных по умолчанию 120 — Однорядная таблица, содержащая поля обрабатываемых экземпляров класса. Ряд по умолчанию 119 — 0 Переменные среды 123 — Только стандартные 123 переменные.	namingExpression
Контекст хранилища. Контекст для ссылки на внутреннее хранилище, используемое классом.	storageContext
Количество параллельно работающих привязок. Данная настройка определяет размер ядра пула потоков класса, т.е. стандартное количество привязок 889, которые обрабатываются одновременно.	normalConcurrentBindings
Максимум параллельно работающих привязок. Данная настройка определяет максимальный размер пула потоков класса, т.е. количество одновременно обрабатываемых привязок, разрешенных в случае заполнения очереди необработанных привязок.	maximumConcurrentBindings
Максимальная длина очереди необработанных привязок. Данная настройка определяет, сколько необработанных операций привязки можно поставить в очередь, прежде чем размер пула потоков класса превысит ее размер ядра относительно максимального размера.	maximumBindingQueueLength
Количество параллельных потоков. Размер ядра пула потоков, который будет использоваться для обработки привязок одного экземпляра класса.	normalConcurrentInstanceBindings
Максимум параллельных потоков. Данная настройка определяет максимальный размер пула потоков, который будет использоваться для обработки привязок одного экземпляра класса, т.е. количество одновременно обрабатываемых привязок, разрешенное при переполнении очереди привязок.	maximumConcurrentInstanceBindings
Максимальная длина очереди необработанных потоков. Данная настройка определяет, сколько необработанных операций привязки экземпляра класса можно поставить в очередь, прежде чем размер пула потоков экземпляра класса превысит ее размер ядра относительно максимального размера.	maximumInstanceBindingQueueLength

Эти свойства доступны через переменную childInfo 1474.

12.15.7.2 Поля

Таблица полей содержит свойства полей[887] класса. Каждое поле хранится в отдельной колонке таблицы базы данных, которая хранит экземпляры класса.

Описание поля	Имя поля
Имя. Имя поля. Совпадает с именем колонки базы данных, которая хранит значения поля.	name
Тип. Тип поля, один из типов, разрешенных для Полей таблицы данных[52]. Тип колонки таблицы базы данных, которая будет создана для хранения значений поля, является специфичной для Реляционной системы управления базами данных (Relational DataBase Management System, RBDMS).	type
Описание. Получите больше информации в разделе Свойства поля таблицы данных[51].	description
По умолчанию. Получите больше информации в разделе Свойства поля таблицы данных[51].	defaultValue
Только чтение. Получите больше информации в разделе Свойства поля таблицы данных[51].	readonly
Может быть NULL. Получите больше информации в разделе Свойства поля таблицы данных[51].	nullable
Ключ. Получите больше информации в разделе Свойства поля таблицы данных[51].	key
Допустимые значения. Получите больше информации в разделе Свойства поля таблицы данных[51].	selvals
Расширяемые допустимые значения. Получите больше информации в разделе Свойства поля таблицы данных[51].	extselvals
Скрытый. Получите больше информации в разделе Свойства поля таблицы данных[51].	hidden
Встроенное. Получите больше информации в разделе Свойства поля таблицы данных[51].	inline
Зашифрованное. Получите больше информации в разделе Свойства поля таблицы данных[51].	encrypted
Помощь. Получите больше информации в разделе Свойства поля таблицы данных[51].	help
Редактор. Получите больше информации в разделе Свойства поля таблицы данных[51].	editor
Опции редактора. Получите больше информации в разделе Свойства поля таблицы данных[51].	editorOptions
Группа. Получите больше информации в разделе Свойства поля таблицы данных[51].	group
Первичный ключ. Флажок, показывающий, что это поле - первичный ключ выбранного класса. В классе должно быть только одно поле с этим флажком.	primaryKey

Длина. Длина поля. Используется только для полей строк.	length

Эти свойства доступны с помощью переменной fields[1474].

12.15.7.3 Отношения многие ко многим

Эта таблица определяет отношения[888] многие ко многим между экземплярами этого класса и экземплярами других классов.

Описание поля	Имя поля
Имя. Имя отношения.	name
Описание. Удобное для чтения описание отношения.	description
Связанный класс. Класс, который связан отношением многие ко многим.	relatedClass
Каскадное удаление. Определяет, должны ли экземпляры связанного класса автоматически удаляться, если один экземпляр этого класса удален.	cascaseDelete

Эти свойства доступны с помощью переменной manyToManyRelations[1475].

12.15.7.4 Жизненные циклы

Эта таблица определяет жизненные циклы[888] класса.

Описание поля	Имя поля
Имя. Имя жизненного цикла.	name
Описание. Удобное для чтения описание жизненного цикла.	description
Состояния. Таблица, которая определяет состояния экземпляров класса, доступные внутри жизненного цикла. Каждое состояние определено: • **Именем**. Имя состояния. • **Описанием**. Удобное для чтения описание состояния. • **Условием входа**. Выражение[112], которое должно возвращать true, если экземпляр класса в данный момент может переключиться на новое состояние. При любом переходе между состояниями оценивается условие входа нового состояния. • **Условием выхода**. Выражение[112], которое должно возвращать true, если экземпляр класса в данный момент может покинуть текущее состояние. При любом переходе между состояниями оценивается условие выхода текущего состояния.	states

Среда вычисления[114] выражения условий входа и выхода:

Контекст по умолчанию[119]	Контекст класса.
Таблица данных по умолчанию[120]	Однорядная таблица, содержащая поля обрабатываемого экземпляра класса.
Ряд по умолчанию[119]	0
Переменные среды[123]	Только стандартные[123] переменные.

• **Выражениями входа**. Список привязок 735, который изменяет поля экземпляра класса, когда он входит в новое состояние. Каждая привязка определяется полем **цель** и **выражением**, которое должно предоставить новое значение поля. При любом переходе между состояниями обрабатываются все выражения входа нового состояния. • **Выражениями выхода**.Список привязок 735, который изменяет поля экземпляра класса, когда он выходит из текущего состояния. Каждая привязка определяется полем **цель** и **выражением**, которое должно предоставить новое значение поля. При любом переходе между состояниями обрабатываются все выражения выхода текущего состояния. Среда вычисления 114 выражений входа и выхода: Контекст по умолчанию 119: Контекст класса. Таблица данных по умолчанию 120: Однорядная таблица, содержащая поля обрабатываемого экземпляра класса. Ряд по умолчанию 119: 0 Переменные среды 123: Только стандартные 123 переменные. • **Заполнить параметры**. Таблица, которая определяет, какие поля класса должны быть определены оператором во время перехода между состояниями. Для каждого поля класса, эта таблица определяет, должны ли быть заполнены поля состояния **При входе** и **При выходе.**	
Переходы между состояниями. Таблица, которая определяет разрешенные переходы между **состояниями** экземпляров. Таблица переходов имеет специальную яйчейку для перехода между любыми двумя определенными **состояниями**. Значение яйчейки должно определять удобное для чтения имя перехода.	stateTransitions

Эти свойства доступны с помощью переменной lifecycles 1476.

12.15.7.5 Просмотры

Эта таблица определяет просмотры 888 класса.

Описание поля	Имя поля
Имя. Имя просмотра.	name
Описание. Удобное для чтения описания просмотра.	description
Колонки. Список полей 887 класса, показанных, когда просмотр активен. По умолчанию, все поля класса добавляются к столбцам таблицы и разрешены следующие изменения: • Перегруппировка полей (колонок просмотра) • Установка поля **Visibility** на **Visible** (видимое при просмотре), **Hidden** (невидимое при просмотре, но включено в исходные данные просмотра, т.е. для вычисления значений вычислимых полей), либо **Disabled** (не включено в просмотр) • Установка любого поля просмотра на событие **Read-only**, если оригинальное поле класса редактируемо Также можно добавить новые **Вычислимые поля** в просмотр и смешать их с оригинальными полями класса. Формат каждого вычислимого поля можно задать через настройки 50 **Поля класса**. После добавления вычислимого поля необходимо добавить **Привязку** просмотра, которая будет вычислять его значение для каждого экземпляра класса, ссылаясь на другие колонки просмотра.	columns

Фильтр. Фильтр 889 экземпляра класса, применяемый, когда просмотр активен.	filter
Сортировка. Правила сортировки экземпляров, применяемые, когда просмотр активен.	sorting
Привязки. Привязки 735 полей классов, применяемые, когда просмотр активен. Каждая привязка определяет вычислимое поле (колонку), чье значение будет вычисляться, а также используемое для этого выражение 112.	bindings

Среда вычисления 114 выражения привязки просмотра:	
Контекст по умолчаниюе 119	Контекст класса.
Таблица данных по умолчанию 120	Таблица представляет данные об обрабатываемой записи просмотра, включая все поля класса и вычислимые поля.
Ряд по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

Эти свойства доступны с помощью переменной просмотры 1476.

12.15.7.6 Привязки

Эта таблица определяет привязки 889 экземпляра класса.

Описание поля	Имя поля
Цель. Цель привязки - это особый тип ссылки 117, которая указывает на место, куда будет записан результат вычисления выражения привязки после обработки привязки.	target
Выражение. Выражение 112, которое вычисляется каждый раз при обработке привязки. Результат вычисления сохраняется в **цель привязки**.	expression
Активатор. Ссылка на определенное поле класса, которое будет запускать процесс обработки привязки. Параметр **Активатор** доступен только при включенном параметре **При событии**.	activator
Условие. Выражение 112, вычисляемое первым после активации привязки. Если это выражение возвращает false, выполнение привязки пропускается.	condition
При запуске. Если этот параметр включен, привязка обрабатывается каждый раз при создании экземпляра класса.	onstartup
При событии. Если этот параметр включен и указан **Активатор,** привязка обрабатывается всякий раз при изменении свойства, на которое ссылается активатор. Если **Активатор** ссылается на событие, привязка будет обрабатываться при возникновении события. Если включен параметр **При событии**, а **Активатор** не указан, привязка будет обрабатываться автоматически: **Выражение** привязки содержит ссылки на одну или более переменных, и изменение какой-либо из них запустит выполнение привязки.	onevent
Периодически. Если этот параметр включен, обработка привязки будет происходить периодически.	periodically
Период. Интервал между сессиями обработки привязки. Этот параметр можно изменить только при включенном параметре **Периодически**.	period

Очередь. Имя очереди обработки целей привязки. Движок привязок гарантирует, что доступ к целям привязок из одной и той же очереди будет осуществляться последовательно, в порядке выполнения привязок.	queue

12.16 Процессы

Процесс - многопоточный графический язык, который состоит из блоков[897], соединённых между собой коннекторами[1273]. Процесс позволяет реализовать логику со сложным взаимодействием пользователя и системы. Процесс выполняет заданные блоки в определённом порядке. Блок представляет собой операцию. Например: вызов функции на сервере, вычисление выражения, взаимодействие с пользователем и т.д. Блоки[897] также могут содержать входные аргументы.

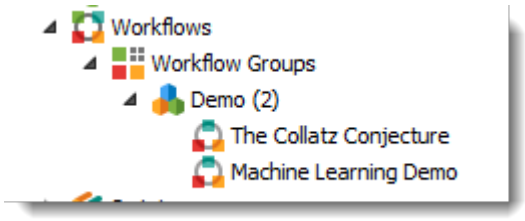
Процесс, написанный на данном графическом языке, транслируется в Java-код. Этот код выполняется в Виртуальной машине Java (JVM), которая запускается AtomMind Server. Каждый процесс выполняется внутри ядра сервера и имеет доступ ко всем объектам и структурам памяти сервера. Таким образом, процессы являются мощным инструментом контроля работы сервера в реальном времени.

Пример процесса

Оператор системы мониторинга аварийных ситуаций получает пожарную тревогу от системы. В случае пожарной тревоги оператор должен следовать заранее определенной последовательности действий. Например, в первую очередь, он должен проанализировать процесс распространения огня на видеопанели, затем, позвонить начальнику смены, после этого, он должен изолировать источник электропитания и так далее. Эта последовательность действий оператора может быть сконфигурирована Процессом и запущена по сигналу пожарной тревоги.

Администрирование процессов

Для администрирования процессов используется два контекста: первый - это общий контекст Процессы[1629], который служит как контейнер. Второй - это контекст Процесс[1630], который содержит информацию для процесса.



12.16.1 Конфигурация процессов

Этот раздел описывает свойства конфигурации процесса.

Все свойства описанные здесь доступны при помощи действия Конфигурировать[105] контекста Процесс[1630].

12.16.1.1 Свойства процессов

Свойства процессов определяют базовые опции для процесса.

Описание поля	Имя поля
Имя. Имя контекста процесса. Оно должно удовлетворять соглашениям по наименованию[42] контекста. Имя должно ссылаться на этот процесс из других частей системы.	name
Описание. Текстовое описание[43] контекста процесса.	description
Шаблон процесса. Текст шаблона[946] виджета в формате XML	template
Размер пула. Определяет размер ядра пула потоков процесса, т.е. стандартное количество задач[896], которые обрабатываются одновременно.	corePoolSize
Максимальный размер пула. Определяет максимальный размер пула потоков процесса, т.е. количество одновременно обрабатываемых задач[896], разрешенных в	maximumPoolSize

случае переполнения очереди задач.	
Максимальная длина необработанной очереди задач. Определяет, сколько необработанных задач[896] можно поставить в очередь, прежде чем размер пула потоков процесса превысит его размер ядра относительно максимального размера	maximumTaskQueueLength
Многопользовательское выполнение. Позволяет выполнять тот же самый процесс несколькими пользователями одновременно.	multiUser
Журналирование. Позволяет журналировать выполнение процесса	logWorkflowExecution

12.16.2 Жизненный цикл процесса

Как только процесс создан и сконфигурирован, он может выполняться как действие[87] или функция[70]. При запуске как интерактивное действие процесс запустится в интерактивном режиме. Если же процесс запущен как функция или действие в неинтерактивном режиме, то процесс запустится в неинтерактивном режиме.

При запуске процесс проходит определенный цикл:

- При повторном запуске, останавливается предыдущий процесс, если он был запущен. При многопользовательском режиме[896] перезапускается только процесс текущего пользователя. Процессы остальных пользователей продолжат работать в старой конфигурации.
- Процесс инициализирует входные блоки[899], чтобы события и изменения вызывали обработку задач процесса.

Задача процесса

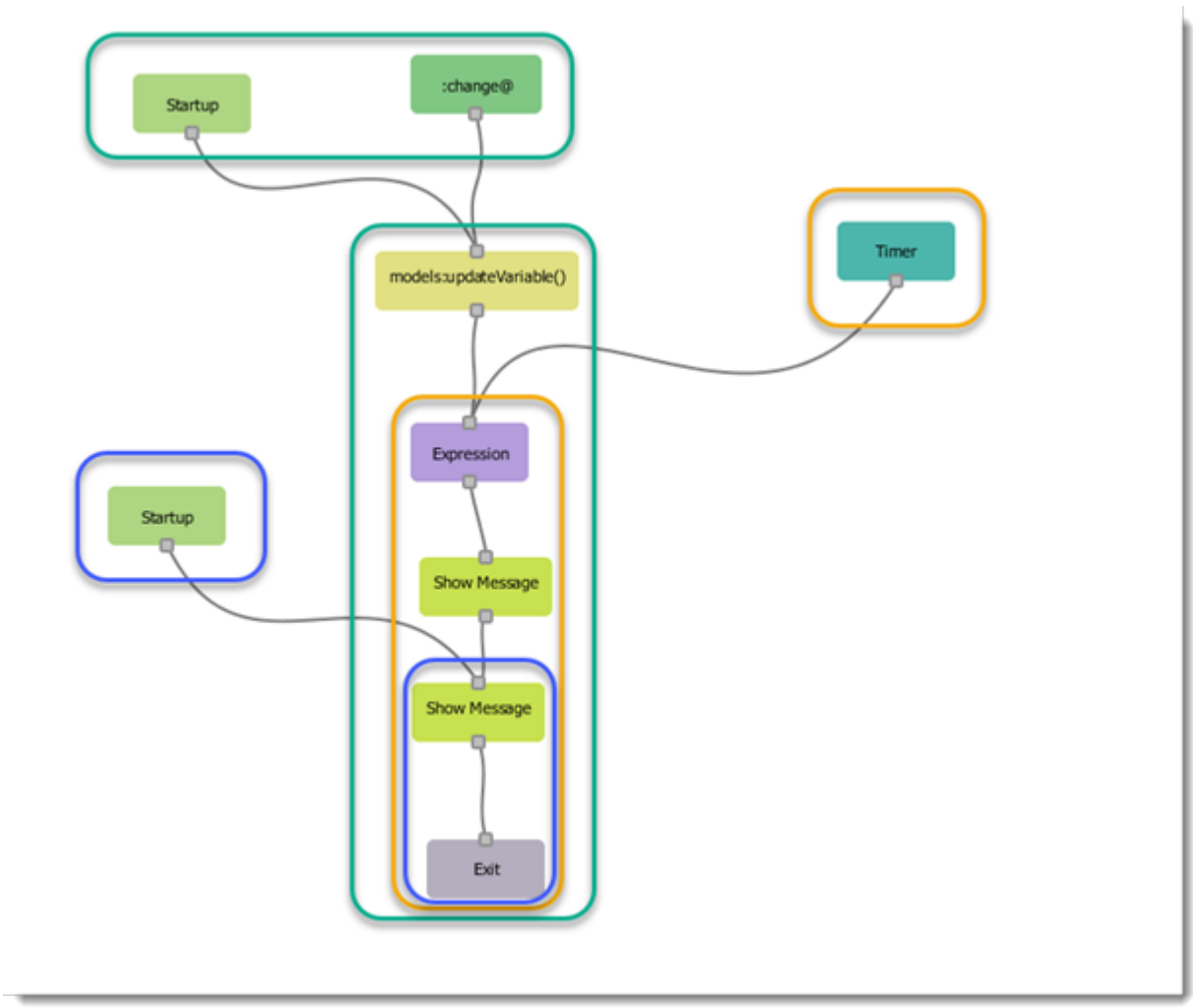
Задача - это множество соединенных между собой блоков[897]. Каждая задача должна начинаться с входного блока. При каждой активации входного блока[899] создается задача на обработку в отдельном потоке, где точкой входа служит сам входной блок, т.е. выполнение задачи начнется со следующего блока, который соединен с входным блоком. Настроить количество одновременных задач можно в свойствах процесса[895].

Пример:

При активации входных блоков, которые выделены зеленым цветом, будет выполнятся задача, выделенная зеленым цветом.

При активации входного блока, который выделены оранжевым цветом, будет выполнятся задача, выделенная оранжевым цветом.

При активации входного блока, который выделены синим цветом, будет выполнятся задача, выделенная синим цветом.



Нормальное функционирование процесса

Как только заканчивается первичный запуск, процесс переключается в ждущий режим. Его последующие действия:

- Входной блок Startup 899 сразу после первичного запуска, добавляет задачу на обработку.
- Входной блок Timer 899 обрабатывается согласно графику, заставляя процесс обрабатывать соответствующие задачи.
- Входной блок Event 899 согласно активатору также вызывают обработку определенных задач процесса.

12.16.3 Блоки

Логика процесса определяется блоками. Блок - это операция, которая имеет тип и параметры. Выполнение процесса всегда начинается с входных блоков. Некоторые типы блоков запрашивают взаимодействие с пользователем, такие блоки содержат все процессы для ответа пользователя при выполнении. Когда выполнение блока закончено, процесс выполняет следующий блок.

Функциональные блоки могут иметь возвращаемое значение. После выполнения блока, результат выполнения операции хранится в переменной среды 123 с именем блока. Поэтому каждый блок может получать значения для каждого блока из ранее выполненных с помощью ссылки на переменные среды 123. Если же значение не было посчитано, то вернется null.Также доступен результат предыдущего вычисления блока и хранится в переменная среды 123 с именем value. Если же предыдущее значение отсутствует, то вернется null.

Каждый блок имеет графический компонент.

Группы блоков

Существует три основные группы блоков:

- Входные блоки 899 с которых начинается выполнение процесса
- Функциональные блоки 901
- Блоки управляющие порядком 905 выполнения процесса

Соединение блоков

Каждый блок имеет точки прикрепления 1284, с помощью которых блоки соединяются друг с другом посредством коннекторов 1273. Входящие точки прикрепления 1284 всегда находятся сверху компонента, выходные - снизу. В одну входящую точку прикрепления 1284 может входить сколько угодно коннекторов 1273. Из выходящей точки прикрепления 1284 может выходить только один коннектор 1273.

Среда вычисления выражений и значений

Поведение каждого типа блоков управляется **параметрами**. Выражения, которые могут появляться в параметрах, так же как и значения, находятся в одной среде. После обработки блока, результат выполнения операции хранится в среде вычисления переменных 114 с именем блока.

Среда вычисления 114 выражений и значений параметров:	
Контекст по умолчанию 119	Сам контекст Процесс 1630.
Таблица данных по умолчанию 120	Таблица, которая задана при запуске процесса. Таблица не разделяется между задачами 896. Для каждой задачи 896 создается отдельный экземпляр таблицы.
Переменные среды 123	Переменные для каждого результата функциональных блоков и предыдущего блока данного процесса. Данные переменные являются локальными и для каждой задачи 896 имеют свои значения.

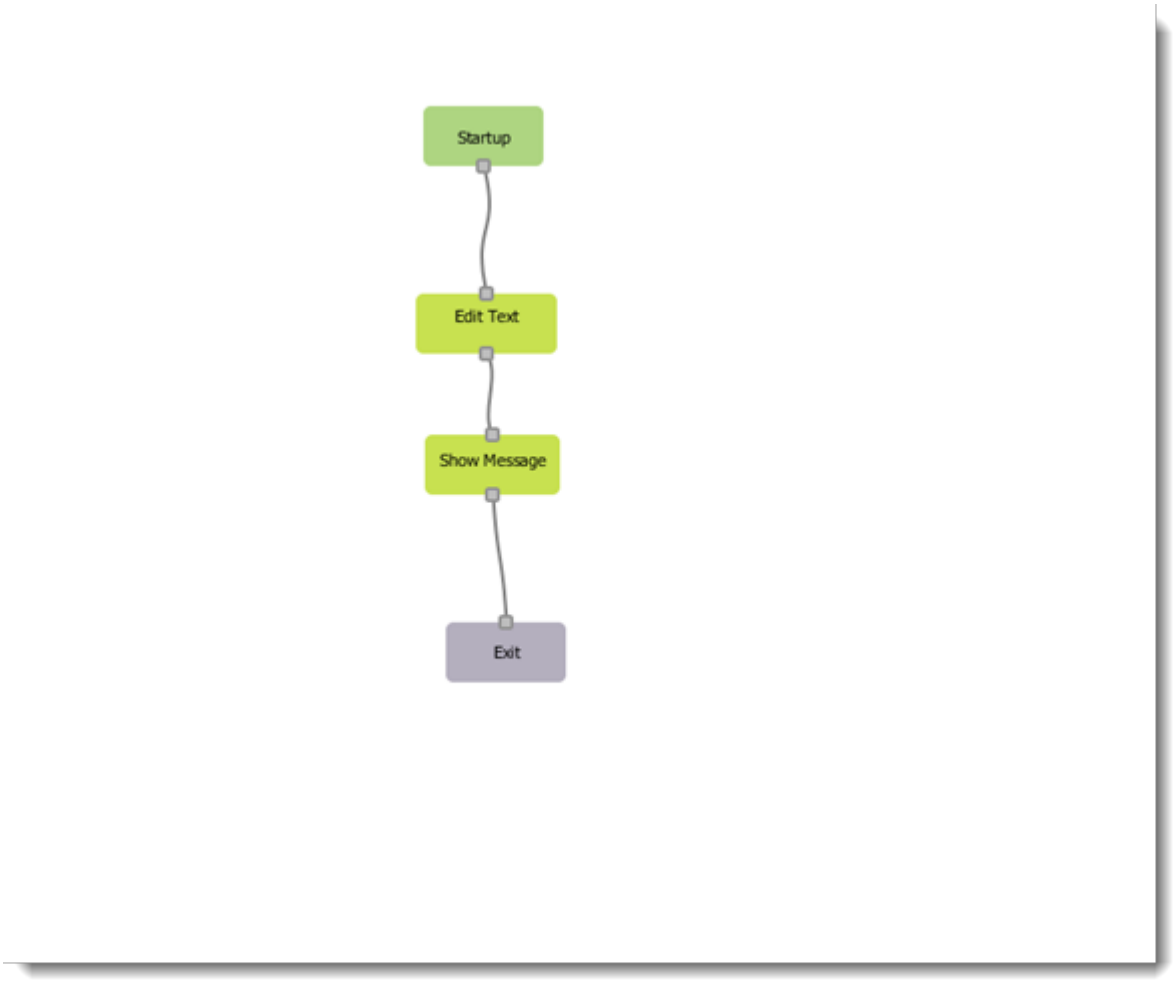
Привязки параметров

Если необходимо получить динамическое поведение блока, должны быть использованы параметры его привязок 741. Эти привязки будут оценены перед выполнением каждого шага, и параметры, определенные как цели привязок, будут перезаписаны с помощью результатов выражения привязок.

Пример: Оператор требует ввести текст объяснения, затем отобразить это объяснение в виде сообщения. При блоке (вызывающем UI процедуру Edit Text, названном `explanation`) мы подсказываем оператору, что нужно ввести текст сообщения, затем нам требуется передать этот текст как параметр UI процедуры отображения сообщений. Поэтому, к блоку(вызывающему UI процедуру Show Message) должна быть добавлена следующая привязка параметров:

- Цель: `message`
- Выражение: `cell({env/explanation},"text")` или `cell({env/value},"text")`

Где `message` - это параметр функции посылки сообщений, и `{env/explanation}` - это результат выполнения `explanation` блока.



12.16.3.1 Входные блоки

Входной блок является точкой входа в процесс. Поэтому каждый процесс должен начинаться с входного блока. При этом входных блоков может быть сколько угодно. Каждый раз, когда один входных блоков активируется, добавляется задача 896 на исполнение процесса. Существуют следующие типы входных блоков:

 Запуск 899

 Событие 900

 Таймер 901

12.16.3.1.1 Запуск

ЗАПУСК

Задача 896 выполняется каждый раз, когда запускается процесс 896.

Графический компонент выглядит так:

Общие свойства

Ширина 1274, Высота 1274, Точки прикрепления 1274

Пользовательские свойства

ВЫРАЖЕНИЕ

См. выражение привязки 738 для получения более подробной информации. Результат выражения сохраняется в переменной среды 123 с именем `value`.

Имя свойства: **expression**

Тип свойства: **String**

УСЛОВИЕ

См. условие привязки 740 для получения более подробной информации.

Имя свойства: **condition**

Тип свойства: **String**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

12.16.3.1.2 Событие

Если обозначен **Активатор** 739, задача 896 обрабатывается, если возникает изменение свойства, на которое ссылается активатор. Если **Активатор** 739 ссылается на событие, задача будет обрабатываться, когда срабатывает событие. Если активен параметр **При событии,** а **Активатор** 739 не определен, задача 896 будет обрабатываться автоматически: **Выражение** 738 блока включает ссылки, указывающие на одну или более переменных. Изменение любой из этих переменных вызывает выполнение задачи 896.

Графический компонент выглядит так:

Общие свойства

Ширина 1274, Высота 1274, Точки прикрепления 1274

Пользовательские свойства

ВЫРАЖЕНИЕ

См. выражение привязки 738 для получения более подробной информации. Результат выражения сохраняется в переменной среды 123 с именем `value`.

Имя свойства: **expression**

Тип свойства: **String**

УСЛОВИЕ

См. условие привязки 740 для получения более подробной информации.

Имя свойства: **condition**

Тип свойства: **String**

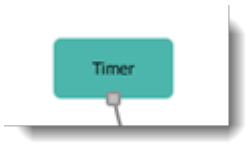
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

12.16.3.1.3 Таймер

Задача 896 будет выполнятся периодически.

Графический компонент выглядит так:



Общие свойства

Ширина 1274, Высота 1274, Точки прикрепления 1274

Пользовательские свойства

ВЫРАЖЕНИЕ

См. выражение привязки 738 для получения более подробной информации. Результат выражения сохраняется в переменной среды 123 с именем `value`.

Имя свойства: **expression**

Тип свойства: **String**

УСЛОВИЕ

См. условие привязки 740 для получения более подробной информации.

Имя свойства: **condition**

Тип свойства: **String**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

12.16.3.2 Функциональные блоки

Функциональные блоки позволяют вычислять выражения, вызывать функции контекстов, вызывать действия и UI процедуры. Существуют следующие типы функциональных блоков:

Выражение 902

Функция 902

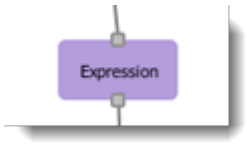
Действие 904

UI процедура 903

12.16.3.2.1 Выражение

Оценивает выражение 1121.

Графический компонент выглядит так:



Общие свойства

Ширина 1274, Высота 1274, Точки прикрепления 1274

Пользовательские свойства

ВЫРАЖЕНИЕ

Выражение для оценки. Среда вычисления переменной описана здесь 898.

Имя свойства: **expression**

Тип свойства: **String**

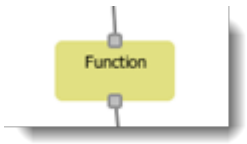
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

12.16.3.2.2 Функция

Вызывает функцию 70 из контекста 41.

Графический компонент выглядит так:



Общие свойства

Ширина 1274, Высота 1274, Точки прикрепления 1274

Пользовательские свойства

ФУНКЦИЯ

Функция выбранного контекста.

Имя свойства: **function**

Тип свойства: **String**

АРГУМЕНТЫ

Таблица данных 49 с параметрами ввода функции.

Имя свойства: **arguments**

Тип свойства: **DataTable**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

12.16.3.2.3 UI процедура

*Процедуры пользовательского интерфейса (*UI процедуры) являются основными структурными элементами действий 87. Данные UI процедуры используются для отображения тревог, подтверждения действий, редактирования свойств и т.д.

Графический компонент выглядит так:

UI процедуры 88 работают только в интерактивном режиме. В неинтерактивном режиме блок UI процедуры 88 бросит исключение с сообщением: *UI procedure cannot execute in headless mode.*

Общие свойства

Ширина 1274, Высота 1274, Точки прикрепления 1274

Пользовательские свойства

ОПИСАНИЕ

Описание, которое будет выводиться в заголовке UI процедуры.

Имя свойства: **description**

Тип свойства: **String**

UI ПРОЦЕДУРА

Тип UI процедуры. См. UI процедура 901 для получения более подробной информации.

Имя свойства: **uiProcedure**

Тип свойства: **String**

АРГУМЕНТЫ

Параметры UI процедуры. См. UI процедура 901 для получения более подробной информации.

Имя свойства: **arguments**

Тип свойства: **DataTable**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

12.16.3.2.4 Действие

Запускает действие 87 в контексте 41.

Графический компонент выглядит так:

Общие свойства

Ширина 1274, Высота 1274, Точки прикрепления 1274

Пользовательские свойства

ДЕЙСТВИЕ

Действие выбранного контекста.

Имя свойства: **action**

Тип свойства: **String**

АРГУМЕНТЫ

Таблица данных 49 с параметрами действия 102.

Имя свойства: **arguments**

Тип свойства: **DataTable**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

12.16.3.2.5 Скрипт

Выполняет **Скрипт**, написанный на Java. Скрипт должен реализовывать `WorkflowScript` интерфейс:

```
public interface WorkflowScript
{
  Object execute(Context con, CallerController caller, RequestController request, EvaluationEnviro
}
```

Графический компонент выглядит следующим образом:

Общие свойства

Ширина 1274, Высота 1274, Точки прикрепления 1274

Пользовательские свойства

СКРИПТ

Исходный код Скрипта на Java. Имя класса скрипты необходимо заменить на формат `%ScriptClassNamePattern%`.

Имя свойства: **script**

Тип свойства: **String**

ОПИСАНИЕ

Описание скрипта.

Имя свойства: **description**

Тип свойства: **String**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

12.16.3.3 Управляющие блоки

Блоки отвечают за порядок выполнение задачи.

Компоненты процессов - графические компоненты блоков 897 процессов 895. Существуют следующие типы управляющих блоков:


Разветвление 905


Обработчик исключений 906


Параллельная задача 907


Выход 907

12.16.3.3.1 Разветвление

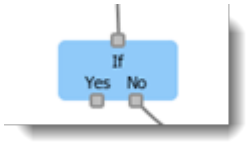
Вычисляет выражение, и в зависимости от результата выбирает следующий блок на выполнение.

Порядок выполнения:

- **Yes.** Следующуй блок, соединенный с **Yes,** будет выполняться, если **выражение** истинно.
- **No.** Следующуй блок, соединенный с **No,** будет выполняться, если **выражение** ложно.

Переменная среды 123 с именем `value` не меняется.

Графический компонент выглядит так:



Общие свойства

Ширина 1274, Высота 1274, Точки прикрепления 1274

Пользовательские свойства

ВЫРАЖЕНИЕ

См. выражение 112 для получения более подробной информации.

Имя свойства: **expression**

Тип свойства: **String**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

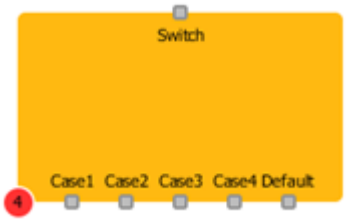
12.16.3.3.2 Переключение

Вычисляет выражение **Условие** и в зависимости от результата выбирает следующий блок на выполнение.

Если результат соответствует одному из значений в таблице Случаев, выполнение процесса перенаправляется на связанный с этим случаем блок. Если соответствующие случаи не найдены, выполнение процесса перенаправляется на блок, связанный со случаем **по умолчанию**.

Переменная среды 123 с именем `value` не меняется.

Графический компонент выглядит так:



Общие свойства

Ширина 1274, Высота 1274, Точки прикрепления 1274

Пользовательские свойства

УСЛОВИЕ

Выражение 112 условия.

Имя свойства: **expression**

Тип свойства: **String**

СЛУЧАИ

Таблица, содержащая описания и значения возможных результатов выражение **Условие**, отличных от результатов по умолчанию. Каждый случай будет использоваться как возможное перенаправление процесса.

Имя свойства: **expression**

Тип свойства: **String**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

12.16.3.3.3 Обработчик исключений

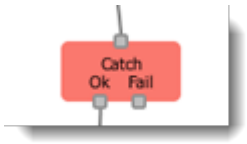
Перехватывает любую ошибку, если она случается при вычислении предыдущего блока.

Порядок выполнения:

- **Ok.** Следующуй блок, соединенный с **Ok,** будет выполняться, если ошибки не произошло.
- **Fail.** Следующуй блок, соединенный с **Fail,** будет выполняться, если ошибка произошла.

Если была ошибка, то сообщение об ошибке сохраняется в переменную среды 123 с именем `value`. В противном случае, переменная среды 123 с именем `value` не меняется.

Графический компонент выглядит так:



Общие свойства

Ширина 1274, Высота 1274, Точки прикрепления 1274

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1288, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

12.16.3.3.4 Параллельная задача

Добавляет задачу на исполнение в отдельном потоке, копируя переменную среды 123 с именем `value`. Текущая задача продолжает работать.

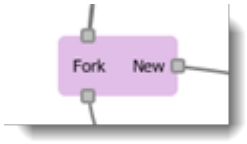
Порядок выполнения:

- **New.** Новая задача будет исполняться со следующего блока, соединенного с **New**.

Для новой задачи будет скопировано значение переменной среды 123 с именем `value` из текущей задачи.

Для текущей задачи переменная среды 123 с именем `value` не меняется.

Графический компонент выглядит так:



Общие свойства

Ширина 1274, Высота 1274, Точки прикрепления 1274

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1288, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

12.16.3.3.5 Выход

Останавливает весь процесс и отменяет все задачи.

Переменная среды 123 с именем `value` не меняется.

Графический компонент выглядит так:

Общие свойства

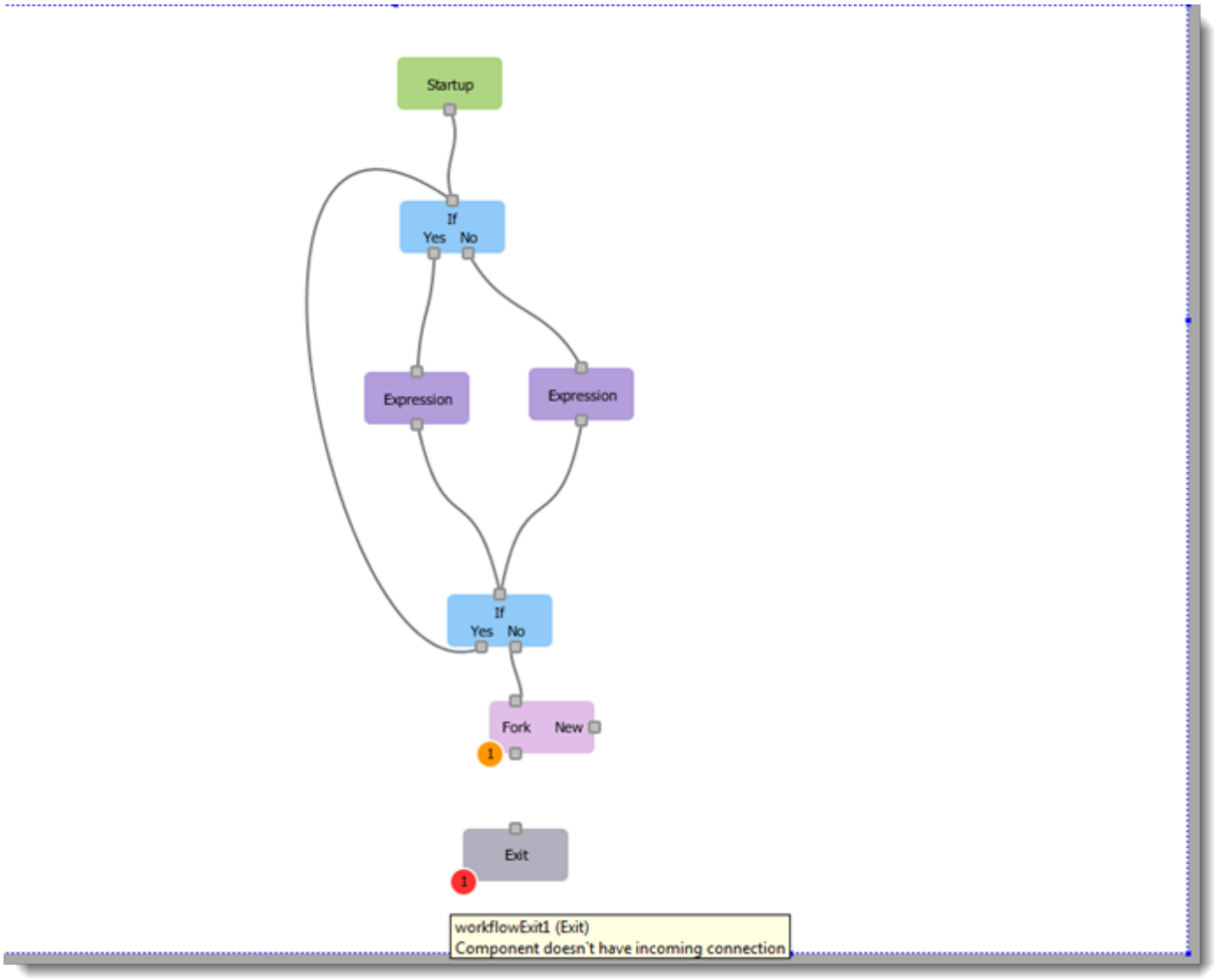
Ширина 1274, Высота 1274, Точки прикрепления 1274

Общие события

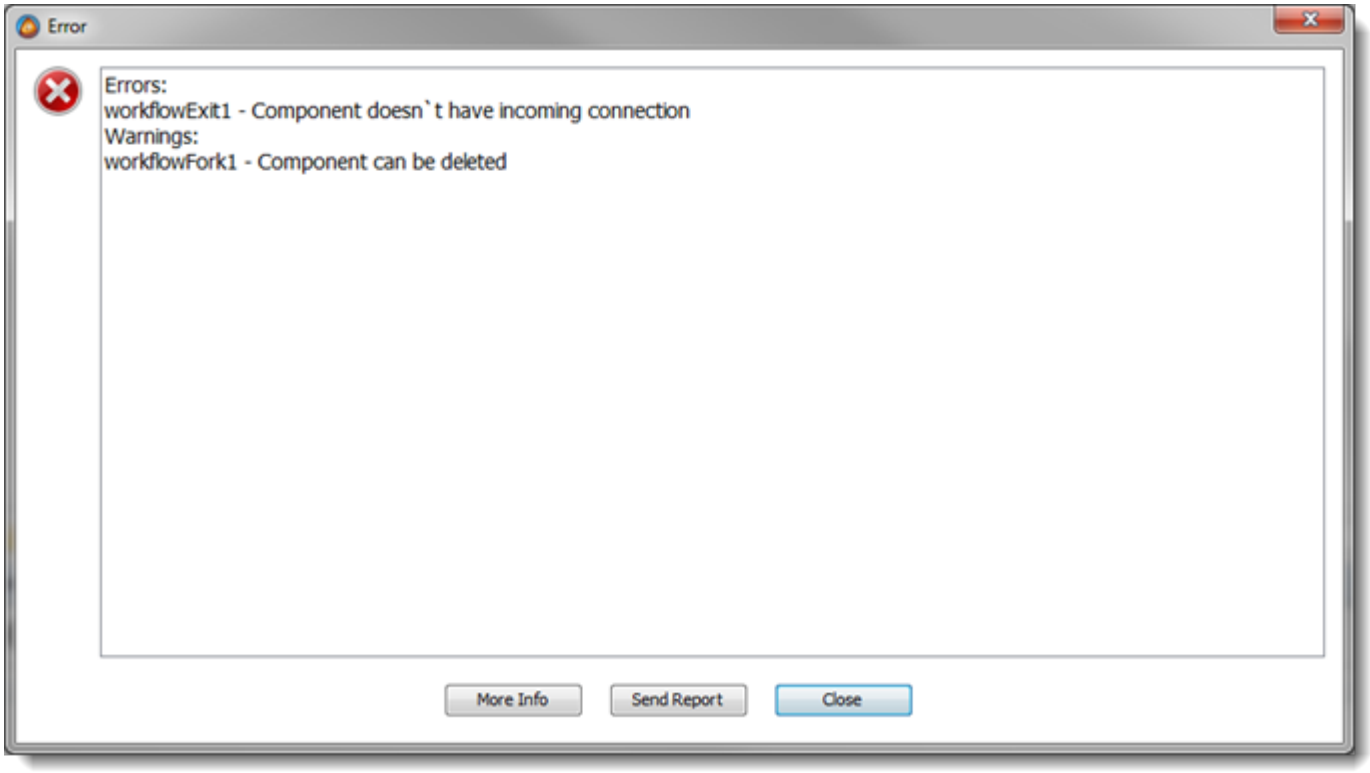
Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

12.16.4 Анализ процессов

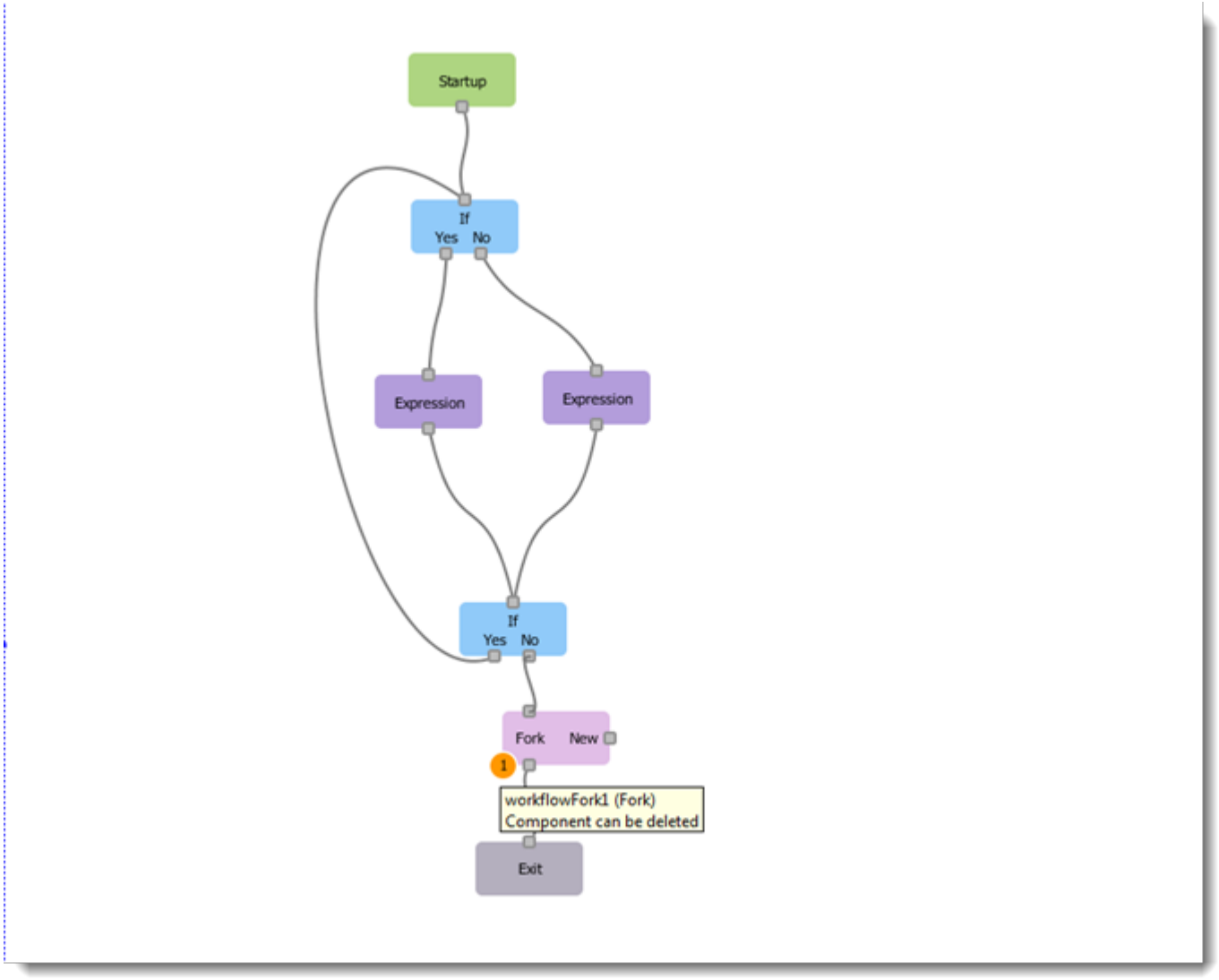
Во время редактирования процесса в редакторе могут отображаться ошибки или предупреждения в левом нижним углу каждого блока 897. Предупреждения рисуются желтым цветом, ошибки - красным. При наведение курсора мыши отобразится всплывающая подсказка с сообщением об ошибке или предупреждении.



Процесс 895 с ошибками запускать нельзя. При попытке запустить процесс с ошибками, то появится диалог со всем ошибками и предупреждениями. В сообщении будет содержаться имя компонента и сообщение об ошибке или предупреждении.



Для того, чтобы запустить процесс, достаточно исправить все ошибки. При этом процесс, который имеет только предупреждения, успешно скомпилируется и запустится, но не будет работать оптимальным образом:



В данном примере компонент Fork может быть удален, так как он ничего не делает.

12.16.5 Производительность процесса

Процессы - это сложные активные ресурсы сервера, которые имеют значительное влияние на производительность.

Влияние процессов на производительность - это совокупность следующих показателей:

- Частота обработки задач[896] процесса. Она может быть явно определена в опциях входных блоков (для периодического блока) или косвенно определена частотой событий или изменениями состояния, вызывающих выполнение задач (для входных блоков при событии).
- Производительность выражений[141].
- Сложность и влияние вызовов функции контекста. См. производительность функций[73] для получения более подробной информации.
- Количество блоков и их порядок выполнения.

Настройки параллельного выполнения задач[895] процесса можно установить, если процесс не способен обрабатывать все свои задачи[896] вовремя.

12.16.6 Безопасность процесса

Все блоки[897] модели обрабатываются при наличии прав доступа[477] владельца процесса. Это по сути означает, что:

- Выражения рассчитываются при наличии прав доступа владельца процесса.

- Вызов функций сервера при наличии прав доступа владельца процесса.
- Вызов действий при наличии прав доступа владельца процесса.

Таким образом, процесс имеет доступ только к тем данным, к которым имеет доступ ее владелец.

Если вы создаете копию определенного процесса в другой учетной записи пользователя 478, возможно, копия не будет нормально функционировать, если новый владелец процесса не имеет прав доступа к данным, передаваемым привязками клонированного процесса.

Модификация процесса

Только пользователи с уровнем прав доступа 486 *Администратора* в контексте процесса могут менять конфигурацию процесса. Это обеспечивает максимальную безопасность для потенциально опасных операций процесса.

12.17 Работа в нескольких временных зонах

AtomMind -- это система с распределенными функциями. В больших установках AtomMind разные компоненты могут быть развернуты в нескольких странах или даже на нескольких континентах. Может получиться, что AtomMind Server, пользователи и Устройства 497 могут находиться в различных временных зонах. При установке системы Вам необходимо правильно настроить временные зоны. После этого временная метка будет автоматически отображаться в нужном часовом поясе для всех системных операций.

Принципы работы AtomMind в рамках нескольких временных зон:

1. AtomMind исходит из того, что сервер находится во временной зоне сервера, указанной в настройках общей конфигурации 179.
2. Временная зона для каждого пользователя 478 определяется в настройках учетной записи пользователя 478. Если временная зона пользователя указана как **Не выбрано**, предполагается, что пользователь находится во временной зоне сервера. Для новых пользователей временная зона по умолчанию - **Не выбрано**.
3. Временная зона для каждого устройства определяется в настройках учетной записи устройства 511. Если временная зона устройства указана как **Не выбрано**, предполагается, что устройство находится во временной зоне сервера. Для новых устройств временная зона по умолчанию - **Не выбрано**.
4. Внутренне AtomMind Server хранит все параметры даты/времени в UTC.
5. Все временные метки, сериализуемые и передаваемые по протоколу AtomMind 2119, кодируются/декодируются в строки с использованием временной зоны UTC.
6. Когда метки времени показываются пользователю в Клиенте AtomMind 359 или Web UI 220, используется местное время временной зоны пользователя. Если временная зона пользователя **Не выбрана**, все метки времени в Web UI показываются согласно временной зоне сервера; все метки времени в AtomMind Client 359 показываются по умолчанию согласно временной зоне ПК, на котором работает AtomMind Client.
7. Временная зона, заданная в операционной системе AtomMind Server используется лишь один раз, чтобы установить временную зону по умолчанию для AtomMind Server. После этого используется только временная зона, определенная в настройках сервера.

13 Визуализация данных

Этот раздел описывает все модули визуализации данных и методы создания пользовательского интерфейса ваших приложений.

13.1 Инструментальные панели

Инструментальные панели - высокоуровневые элементы пользовательских интерфейсов. Каждая инструментальная панель содержит компоненты пользовательского интерфейса, отображенные согласно одной из *компоновок*. Пользовательский интерфейс любого приложения, решения, сервиса или продукта на базе платформы AtomMind - это, по сути, набор инструментальных панелей, объединенных правилами навигации и маршрутизации.

Инструментальные панели обладают достаточной гибкостью, чтобы выступать в качестве любого пользовательского интерфейса, включая формы ввода и отображения данных, человеко-машинные интерфейсы, планы здания/этажа, географические карты, отчеты в стиле бизнес-аналитики с множеством графиков/диаграмм, экраны видеомониторинга в режиме реального времени, и многое другое. Инструментальные панели могут реагировать на вводимую оператором информацию и динамически обновляться, отражая изменения в модели данных на стороне сервера.

По своему **Назначению**, инструментальные панели делятся на:

- Веб 918 панели, созданные для работы в веб интерфейсе 220
- Десктопные 918, работающие в AtomMind Client 359

Инструментальные панели можно также разделить по **Типу**:

- **Абсолютные** - дают общий обзор инфраструктуры, Извлекая данные из множества различных источников
- **Относительные -** работают с одним источником данных (например, устройством 497 или моделью 810)

Добавление виджетов 943 на **Десктопные** инструментальные панели дает дополнительный уровень гибкости их компоновкам благодаря встроенной системе контейнеров.

Хотя продукты для вертикальных рынков, созданные на базе AtomMind, содержат множество готовых инструментальных панелей, всех их можно полностью настроить под себя, а также создать новую инструментальную панель с нуля.

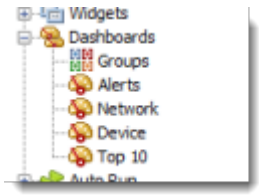
элементы инструментальных панелей

Каждая инструментальная панель имеет несколько элементов, которые составляют ее содержание:

- Элементы **Веб** инструментальной панели - компоненты 231 ее пользовательского интерфейса, описанные в главе Web UI 220.
- Элементы **Десктопной** инструментальной панели описаны в главе Элементы инструментальной панели 918.

Администрирование инструментальных панелей

Для администрирования инструментальных панелей используются два контекста: общий контекст Инструментальные панели 1489, который служит контейнером, и контекст Инструментальная панель 1490, который содержит информацию об одной инструментальной панели.



Каждый пользователь 478 имеет свой набор Инструментальных панелей.

13.1.1 Типы инструментальных панелей

Инструментальные панели делятся на три типа:

- Абсолютные
- Относительные
- Экземпляра класса

Абсолютные инструментальные панели

Абсолютные панели в основном используются для отображения данных, например, нескольких устройств.

В некоторых случаях абсолютная инструментальная панель может отображать или контролировать ранее определенное устройство или источник, чье имя жестко закодировано в параметрах элементов инструментальной панели 918.

Абсолютные инструментальные панели открываются щелчком правой кнопкой мыши по инструментальной панели 1490 ( ) и выбора действия "Открыть инструментальную панель" 913.

Относительные инструментальные панели

Относительные инструментальные панели отображают статус одного устройства/источника, которое не жестко закодировано в конфигурации инструментальной панели. Относительная инструментальная панель открывается для определенного устройства/источника при помощи правого щелчка по нему и выбора действия "Запустить инструментальную панель", имя которой совпадает с описанием инструментальной панели (иконка). Следует отметить, что относительная инструментальная панель, помещенная на другую относительную инструментальную панель, наследует контекст по умолчанию своей родительской инструментальной панели.

инструментальные панели экземпляра класса

Инструментальные панели экземпляра класса визуализируют состояние и поля экземпляра определенного класса 885 и списки экземпляров классов, связанных с ним. Панель экземпляра класса невозможно открыть напрямую, только запустить путем клика на определенный экземпляр класса в списках классов, отображенных на других инструментальных панелях.

13.1.2 Открытие инструментальных панелей

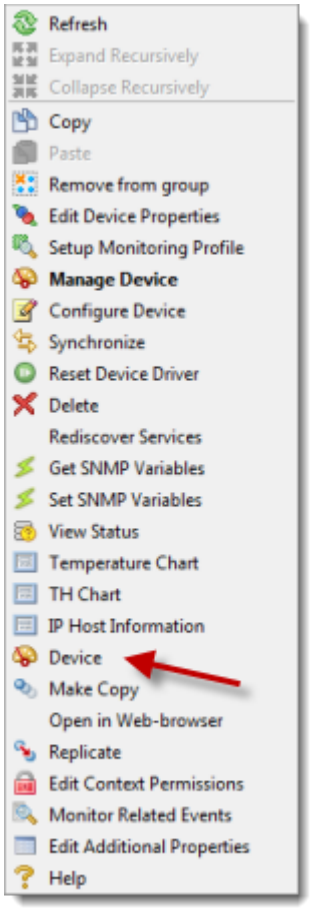
Существуют несколько способов открытия инструментальных панелей:

- При помощи действия Открыть панель 1490 контекста Инструментальная панель. Данный метод следует использовать для абсолютных 913 инструментальных панелей. Для относительных 913 панелей данный метод откроет панель для корневого контекста AtomMind Server, вследствие этого в большинстве случаев она не сможет работать.
- При помощи действия 87 "Открыть панель" (см. далее) с целью запуска относительной 946 инструментальной панели для определенного устройства или системного ресурса.

Действие "Открыть панель"

Данное действие находится в любом контексте, для которого выражение пригодности 913 относительной инструментальной панели является **true.** При запуске действия "Открыть панель" в каком-либо контексте, данный контекст становится контекстом по умолчанию 946 для открываемой панели. Например, если инструментальная панель **Обзор пользователя** создается для контекста "Пользователь" 1608 (например, имеет выражение пригодности `{.#type} == 'user'`), действие **Обзор пользователя** появится в контексте каждого пользователя AtomMind Server. Ее описание совпадает с описанием самой инструментальной панели. Действия, используемые для открытия инструментальных панелей, распознаются по иконке **.**

Действие "Открыть панель" в контекстном меню в AtomMind Client выглядит следующим образом:



Относительные панели инструментов будут устанавливать действие Открыть панель только в те контексты, к которым есть доступ у владельца панели в соответствии с его правами доступа 477.

Чтобы запустить это действие, пользователь должен иметь эффективный уровень прав доступа 489 **Наблюдатель** в двух контекстах:

- Контекст панели инструментов 1490 фактически запущенной панели
- Контекст, на котором работает инструментальная панель, т.е. тот, где определена опция Запустить действие.

13.1.3 Настройка инструментальной панели

Данный раздел посвящен свойствам конфигурации инструментальной панели.

Все свойства, описанные здесь, доступны через действие Настроить 105 контекста Инструментальная панель 1490.

13.1.3.1 Свойства инструментальной панели

Таблица **Свойств** инструментальной панели определяет базовые опции инструментальной панели. Эти опции значительно отличаются для веб и десктопных инструментальных панелей.

К этим свойствам можно получить доступ через переменную childInfo 1491.

свойства веб инструментальной панели

Ниже объясняются настройки веб 918 инструментальных панелей.

Описание поля	Field Name
Имя. Имя контекста инструментальной панели, требуемое для ссылки на данную инструментальную панель из других частей системы. Должно соответствовать соглашениям о наименованиях 42 контекста.	name

Описание. Текстовое описание инструментальной панели, которое также является описанием 43 контекста инструментальной панели.	description
Назначение. В данном случае установите на **Веб**.	destination
Тип. Тип 913 инструментальной панели: **Абсолютная** или **Относительная**.	title
Выражение пригодности. Определяет, для какого контекста (контекстов) может быть использована инструментальная панель. Более подробно об этом см. Пригодность ресурсов 749.	validityExpression
Правила обновления пригодности. Перечень контекстных масок и имен событий. Если событие, определенное полем **Событие** этой таблицы, возникает в любом контексте, который соответствует маске, определенной полем **Маска** в той же записи, **Выражение пригодности** для этого контекста пересчитывается. Это позволяет сделать инструментальную панель пригодной и непригодной для определенного контекст, если происходят какие-то изменения.	validityListeners
Определять пригодность для удаленных контекстов. Если флаг включен, инструментальную панель можно прикрепить не только к локальным контекстам, но и к удаленным контекстам, подключенным по распределенной архитектуре 1334.	allowValidityForRemoteContexts
Выражение для обработки шаблона. Это выражение вычисляется при запуске инструментальной панели. Оно принимает таблицу элементов инструментальной панели за таблицу по умолчанию. Выражение должно оперативно менять таблицу элементов (которая, по сути, составляет шаблон инструментальной панели) и возвращать обновленную версию.	templateProcessingExpression

Среда вычисления 114 Выражения для обработки шаблонов:

Контекст по умолчанию 119	Контекст, для которого открывается относительная инструментальная панель
Таблица данных по умолчанию 120	Таблица элементов 917 инструментальной панели
Ряд по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

Включить кэширование. Включает кэширование 224 для данной инструментальной панели. Эта опция будет действовать только при включенном кэшировании в общих настройках плагина Web UI 353.	cacheEnabled

Свойства инструментальной панели для десктопа

Ниже объясняются настройки инструментальной панели для десктопа 918.

Описание поля	Field Name
Имя. Имя контекста инструментальной панели, требуемое для ссылки на данную инструментальную панель из других частей системы. Должно соответствовать соглашениям о наименованиях 42 контекста.	name
Описание. Текстовое описание инструментальной панели, которое также является описанием 43 контекста инструментальной панели.	description
Выражение заголовка. Заголовок инструментальной панели, который появляется в верхней части окна панели. Если заголовок не указан, вместо него используется **Описание**. Заголовок может быть обычной строкой, либо выражением 112. Если заголовок представлен выражением, оно вычисляется в строку. Это позволяет заголовкам относительных 913 инструментальных панелей отображать значения из контекстов, чьи данные показываются на инструментальной панели.	title

 **Пример:** Допустим, мы создаем инструментальную панель, которая показывает состояние питания устройства. Мы может установить **Выражение заголовка** на `'Power Status: ' + {.#description}`. Тогда заголовок будет включать описание контекста устройства, например, "Power Status: Device213". Среда вычисления 114 Выражения заголовка: Контекст по умолчанию 119 — Контекст, для которого открывается относительная инструментальная панель Таблица данных по умолчанию 120 — Нет Ряд по умолчанию 119 — 0 Переменные среды 123 — Только стандартные 123 переменные.	
Компоновка. Компоновка 924 инструментальной панели, плавающие окна или с прокруткой.	layout
Количество столбцов. Количество столбцов, только для панелей с прокруткой.	columns
Тип. Тип 913 инструментальной панели: **Абсолютная** или **Относительная**.	type
Выражение пригодности. Определяет, для какого контекста (контекстов) может быть использована инструментальная панель. Более подробно об этом см. раздел Пригодность ресурсов 749.	validityExpression
Правила обновления пригодности. Перечень масок контекста и имен событий. Если событие, определенное полем **Событие** этой таблицы, возникает в любом контексте, который соответствует маске, определенной полем **Маска** в той же записи, **Выражение пригодности** для этого контекста пересчитывается. Это позволяет сделать инструментальную панель пригодной и непригодной для определенного контекст, если происходят какие-то изменения.	validityListeners
Определять пригодность для удаленных контекстов. Если флаг включен, инструментальную панель можно прикрепить не только к локальным контекстам, но и к удаленным контекстам, подключенным по распределенной архитектуре 1334.	allowValidityForRemoteContexts
Можно закрыть. Определяет, можно ли закрыть инструментальную панель. Незакрываемые инструментальные панели часто играют роль основной панели в пользовательском интерфейсе оператора.	closable
Закрывать панель при повторном открытии. Определяет, будет ли инструментальная панель закрываться перед повторным открытием.  Использование действия Открыть панель 1490 при активированной опции **Закрывать панель при повторном открытии** на вложенной панели (или панелях), в большинстве случаев приведет к ошибкам в работе.	closeDashboardOnReopen
Запускать через действие "Управление". Определяет, будет ли инструментальная панель открываться посредством действия "Управление узлом", например, действием Управление устройством 1494.	launchViaManage
Панель управления компоновкой. Этот флаг определяет видимость нижней панели (Сохранить/Загрузить макет и другие кнопки).	layoutControlPanel

13.1.3.2 Элементы инструментальной панели

Формат таблицы элементов 918 инструментальной панели зависит от **Назначения** инструментальной панели.

Элементы 918 инструментальной панели определены следующими свойствами:

Описание поля	Имя поля
Заголовок. Заголовок окна элемента.	title
Тип. Тип элемента, более подробно об этом см. раздел Элементы 918.	type
Положение окна. Положение окна элемента внутри окна панели. Более подробную информацию см. здесь.	location
Параметры. Настройки элемента. Настройки определяют, какие данные будут отображаться и как их представлять. Список настроек зависит от **типа** элемента.	parameters
Выражение пригодности. Выражение 112, определяющее, стоит ли отображать элемент на инструментальной панели. Если оно оценивается как FALSE, элемент будет пропущен. Среда вычисления 112 выражения пригодности: Контекст по умолчанию 119 — Контекст, для которого открывается инструментальная панель. Таблица данных по умолчанию 120 — Нет. Ряд по умолчанию 119 — 0 Переменные среды 123 — Только стандартные 123 переменные.	validityExpression

Элементы **веб-версии** инструментальной панели определены следующими свойствами:

Описание поля	Имя поля
Имя. Имя компонента 231 инструментальной панели.	name
Тип. Тип компонента.	type
Параметры. Параметры компонента.	parameters

Доступ к этим свойствам можно получить через переменную элементы 1492.

13.1.4 Безопасность инструментальных панелей

Как только инструментальная панель открывается пользователем 478 системы, открываются и все ее элементы при наличии прав доступа пользователя. Права доступа пользователя-владельца инструментальной панели и пользователей, владеющих отдельными элементами инструментальной панели (например, виджетами 943), игнорируются.

Например, если пользователь Джо открывает инструментальную панель, принадлежащую администратору по умолчанию 479, элементы инструментальной панели будут иметь ограниченные права доступа Джо. Однако если администратор открывает любую из панелей Джо, инструментальная панель и ее элементы будут иметь неограниченные права доступа, допуская потенциальный риск безопасности, если Джо не является пользователем с полномочиями высшего уровня.

13.1.5 Инструментальные панели для веб

Инструментальные панели для веб открываются и работают в веб браузере. Каждая инструментальная панель становится отдельной веб-страницей.

Смотрите раздел Инструментальные панели в Web UI 220 для более подробной информации.

13.1.6 Инструментальные панели для десктопа

Этот раздел описывают настройки и параметры, относящиеся к Инструментальным панелям для десктопа.

Инструментальные панели для десктопа могут открываются и работают только в Десктопном клиентском приложении 359. Открытие таких инструментальных панелей в браузере не поддерживается, для работы в браузере существуют специальные Инструментальные панели для веб 918.

13.1.6.1 Элементы инструментальной панели

Каждая инструментальная панель состоит из *элементов*. Каждый элемент открывается в панели в отдельном подокне окна инструментальной панели.

Свойства 917 элемента включают:

- Заголовок и местоположение окна элемента
- Тип и свойства самого элемента
- Выражение пригодности, которое позволяет исключать определенные элементы, если целевой контекст (т.е. контекст, для которого открывается относительная 913 инструментальная панель) не подходит под особые критерии.

Типы элементов

Поддерживаются следующие типы элементов инструментальной панели:

- Виджет
- Таблица данных
- Свойства
- Журнал событий
- Системное дерево
- Отчет
- Инструментальная панель
- Поля экземпляра класса
- Список экземпляров класса
- Сниппет HTML

ВИДЖЕТ

Данный элемент открывает виджет 943. Если виджет является относительным 946, он запускается для того же контекста, что и инструментальная панель.

Свойства элемента:

- **Виджет**. Путь контексту виджета 1627.

ТАБЛИЦА ДАННЫХ

Данный элемент вычисляет выражение, которое должно возвращать таблицу данных 49. Данные Таблицы данных показаны в Редакторе таблицы данных 382 как часть инструментальной панели.

Элемент Таблица данных идеально подходит для показа результатов запроса 829 в панели инструментов.

Свойства элемента:

- **Выражение получения данных**. Выражение, которое необходимо вычислить для получения данных, которые будут отображены компонентом. Оно должно вернуть Таблицу данных.

Среда вычисления 114 выражения получения данных:	
Контекст по умолчанию 119	• Для абсолютных панелей: сам контекст инструментальной панели • Для относительных панелей: контекст, для которого запущена инструментальная панель • Для инструментальных панелей экземпляров класса: контекст класса
Таблица данных по умолчанию 120	Отсутствует.
Ряд по умолчанию 119	0
Переменые среды 123	Только стандартные 123 переменные.

- **Период обновления**. Период повторного вычисления выражения, т.е. период обновления данных.

Первое вычисление **Выражения** осуществляется сервером, и предваительная таблица посылается в AtomMind Client или другой пользовательский интерфейс AtomMind Server. Последующие вычисления (т.е. через каждый **Период обновления**) выполняются на стороне AtomMind Client или других пользовательских интерфейсов.

- **Только чтение.** Определяет, можно ли редактировать данные.
- **Включить контекстное меню.** Опрелеяет, будет ли показываться контекстное меню.
- **ID иконки**. ID иконки для отображения в заголовке окна элемента.
- **ID справочного раздела**. ID справочного раздела, на который должен ссылаться элемент.
- **Справка**. Текст многострочного информационного сообщения, которое появится над Редактором таблиц данных 382.
- **Показать панель инструментов**. Определяет видимость панели инструментов Редактора таблиц данных.
- **Показать заголовок**. Определяет видимость заголовка Редактора таблиц данных. Неопределенное значение активирует оптимальное поведение по умолчанию.
- **Показать номера строк**. Определяет видимость омеров строк Редактора таблиц данных. Неопределенное значение активирует оптимальное поведение по умолчанию.
- **Горизонтальная прокрутка**. Определяет, будет ли включена горизонтальная прокрутка по умолчанию. Неопределенное значение активирует оптимальное поведение по умолчанию.

СВОЙСТВА

Отображает редактор свойств 377 внутри инструментальной панели.

Свойства элемента:

- **Контекст**. Контекст 41, свойства которого будут отображаться.
- **Группа**. Группа отображаемых свойств или значение NULL для отображения свойств, заданных вручную.
- **Свойства**. Список отображаемых свойств, если значение **Группы** является NULL.
- **Простой режим**. Использует простой режим 378 редактора свойств.
- **Открывать в режиме чтения.** Блокирует редактор при запуске. Необходимо нажать кнопку **Разблокировать** на панели инструментов для редактирования.

ЖУРНАЛ СОБЫТИЙ

Отображает журнал событий 398 на инструментальной панели.

Свойства элемента:

- **Фильтр событий**. Путь контекста используемого фильтра событий 762.
- **События**. Список событий, отображаемых в журнале, определяемых парами "Путь контекста -Имя события". Данная опция включена, если не задан фильтр.
- **Текущие события**. Флажок, указывающий, будет ли отображаться раздел журнала "Текущие события".
- **История событий**. Флажок, указывающий, будет ли отображаться раздел журнала "История событий".

- **Автоматически загружать историю событий.** Включает/Отключает загрузку истории событий при запуске журнала событий.
- **Показывать имена контекстов**. Контролирует видимость столбца "Контекст".
- **Показывать имена событий**. Контролирует видимость столбца "Событие".
- **Показывать уровни событий**. Контролирует видимость столбца "Уровень".
- **Показывать данные событий**. Контролирует видимость столбца "Данные".
- **Показывать подтверждения событий.** Контролирует видимость столбца "Подтверждения".

СИСТЕМНОЕ ДЕРЕВО

Отображает компонент Системное дерево 370 на инструментальной панели. В качестве корня дерева отображается пользователь.

Свойства элемента:

- **Корень**. Путь корневого контекста дерева.
- **Действия**. Флажок, контролирующий видимость подокна "Действия".
- **Контекстное меню**. Флажок, контролирующий видимость Контекстного меню.
- **Показать панель инструментов**. Определяет видимость панели инструментов Системного дерева.
- **Выражение нажатия на узел**. Это выражение вычисляется при нажатии на любой узел Системного дерева.

Среда вычисления 114 Выражения нажатия на узел:

Контекст по умолчанию 119	Контекст инструментальной панели
Таблица данных по умолчанию 120	**Поле** / **Имя** / **Тип** / **Описание** Is Added / isAdded / Boolean / True, если элемент первого пути добавлен к выбору, false - если элемент первого пути удален из выбора. Local Path / localPath / String / Локальный путь в Системном дереве узла, который был добавлен или удален из выбора. Remote Path / remotePath / String / Путь к контексту удаленного сервера узла, который был добавлен или удален из выбора. Selection / selection / Data Table / Список отобранных на настоящий момент узлов. Имеет два поля: • localSelectionPath (String) - Локальный путь выбранного узла в Системном дереве. • remoteSelectionPath (String) - Путь к контексту удаленного сервера выбранного узла
Ряд по умолчанию 119	0
Переменые среды 123	Только стандартные 123 переменные.

- **Выражение фильтрации узла**. Выражение устанавливает а пользовательский фильтр для узлов, отображенных в Системном дереве. Если это выражение вычисляется в true, узел отображается в Системном дереве. В противном случае, узел не отображается.

Среда вычисления 114 Выражения фильтрации узла:

Контекст по умолчанию 119	Контекст узла
Таблица данных по умолчанию 120	Нет.

Ряд по умолчанию 119	0
Переменые среды 123	Только стандартные 123 переменные.

ОТЧЕТ

Показывает обозреватель отчетов 415 внутри инструментальной панели.

Свойства элемента:

- **Отчет**. Путь контекста отчета.

Не все отчеты поддерживаются в Web UI версии. Отчет откроется в новой вкладке браузера или будет загружен, в зависимости от выбранного формата отчета.

ИНСТРУМЕНТАЛЬНАЯ ПАНЕЛЬ

Открывает новую панель инструментов в окне инструментальной панели.

Свойства элемента:

- **Инструментальная панель**. Путь контекста панели инструментов.

СНИППЕТ HTML

Отображает содержание веб-страницы внутри инструментальной панели.

Типы элемента:

- **Фрейм**. Отображает URL-адрес веб-страницы.
- **HTML**. Отображает страницу HTML. Поддерживает элементы CSS.

Вы можете включить или отключить проверку html, используя опцию Проверить HTML.

- **Выражение**. Вычисляет выражение типа string в страницу HTML и отображает ее внутри инструментальной панели.

Вы можете выполнить пользовательское выражение из элемента Сниппет HTML, используя ссылку:

```
<a onclick="<e>{users.admin.dashboards.db2:open!}</e>"
href="javascript:;">Click Me!</a>
```

БИБЛИОТЕКИ JAVASCRIPT В СНИППЕТЕ HTML

Пример ReactJS:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Add React in One Minute</title>
  </head>

  <body>
    <h2>Add React in One Minute</h2>
    <p>This page demonstrates using React with no build tooling.</p>
    <p>React is loaded as a script tag.</p>
    <p>
      This is the first comment.
      <div class="like_button_container" data-commentid="1"></div>
    </p>
  </body>

<script type="text/javascript" src="https://unpkg.com/react@16/umd/react.development.js"></script>
<script type="text/javascript" src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"><
<script type="text/javascript">
const e = React.createElement;
```

```
class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked: false };
  }
  render() {
    let isLiked = localStorage.getItem("likedState") || this.state.liked;
    if (isLiked) {
      return 'You liked comment number ' + this.props.commentID;
    }
    return e(
      'button',
      { onClick: () => { localStorage.setItem("likedState", true); this.setState({ liked: true })}
      'Like'
    );
  }
}
document.querySelectorAll('.like_button_container')
  .forEach(domContainer => {
    const commentID = parseInt(domContainer.dataset.commentid, 10);
    ReactDOM.render(
      e(LikeButton, { commentID: commentID }),
      domContainer
    );
  });
</script>

</html>
```

Пример Vue.js:

```
<!DOCTYPE html>
<html>
<head>
<title>Vue.js</title>
<meta charset="utf-8" />
</head>

<body>
<div id="app">
    <input type="text" v-on:input="setMessage" />
    <p>{{message}}</p>
</div>

<script src="https://unpkg.com/vue"></script>
<script>
'use strict';
var app = new Vue({
    el: '#app',
    data: {
        message: localStorage.getItem("vueMessage") || 'Hello Vue!'
    },
    methods: {
        setMessage: function setMessage(event) {
            this.message = event.target.value;
            localStorage.setItem("vueMessage", this.message);
        }
    }
});
</script>

</body>
</html>
```

Пример AngularJS:

```
<!DOCTYPE html>
<html lang="en">
```

```
<title>AngularJS First Application</title>

<head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.26/angular.min.js"></script>
</head>

<body>
<div ng-app>

   <p>Enter Some text : <input type="text" ng-model="someText"></p>
   <p>Hello {{ someText }}!</p>

</div>
</body>
</html>
```

ПОЛЯ ЭКЗЕМПЛЯРА КЛАССА

данный элемент отображает набор или группу полей[887] экземпляра класса[885] в Редакторе таблиц данных[382]. Он доступен только для панелей инструментов экземпляра класса[913].

Свойства элемента:

- **Только чтение.** Определяет, можно ли редактировать данные.
- **Контекст хранилища**. Определяет контекст хранения[1447] для получения данных. Для классов, должен указывать на контекст класса[1473].
- **Просмотр**. Определяет используемый просмотр[888].
- **Специальный запрос**. Определяет запрос SQL для выполнения. Должен использоваться, когда контекст хранилища указывает на внешнюю базу данных. Доступно, только если **Просмотр** является нестандартным.
- **Класс/Таблица**.Класс или таблица для получения данных. Доступны только если **Просмотр** нестандартный.
- **Тип набора полей**. Определяет, будет ли показана преднастроенная группа полей или пользовательский набор полей.
- **Группа**. Отпределяет, какую группу полей отображать. Доступна, только если **Тип набора групп** является группой.
- **Поля**. Определяет отображаемые поля. Доступна, только если **Тип набора групп** является выбранными полями.
- **ID иконки**. ID иконки для отображения заголовка окна элемента.
- **ID справочного раздела**. ID справочного огтдела, на который должен ссылаться элемент.
- **Справка**. Текст многострочного информационного сообщения, которое появится над Редактором таблиц данных[382].
- **Показать панель инструментов**. Определяет видимость панели инструментов Редактора таблиц данных.
- **Показать заголовок**. Определяет видимость заголовка Редактора таблиц данных. Неопределенное значение активирует оптимальное поведение по умолчанию.
- **Показать номера строк**. Определяет видимость омеров строк Редактора таблиц данных. Неопределенное значение активирует оптимальное поведение по умолчанию..
- **Горизонтальная прокрутка**. Определяет, будет ли включена горизонтальная прокрутка по умолчанию. Неопределенное значение активирует оптимальное поведение по умолчанию.

СПИСОК ЭКЗЕМПЛЯРОВ КЛАССА

Данный элемент отображает список экземпляров класса[885] в Редакторе таблиц данных[382].

Свойства элемента:

- **Только чтение.** Определяет, можно ли редактировать данные.
- **Контекст хранилища**. Определяет контекст хранения[1447] для получения данных. Для классов, должен указывать на контекст класса[1473].
- **Просмотр**. Определяет используемый просмотр[888].
- **Специальный запрос**. Определяет запрос SQL для выполнения. Должен использоваться, когда контекст хранилища указывает на внешнюю базу данных. Доступно, только если **Просмотр** является нестандартным.

- **Класс/Таблица**. Класс или таблица для получения данных. Доступны только если **Просмотр** нестандартный.
- **Фильтр**. Настраивает правила фильтрации 889, применимые к экземплярам. Доступны только если **Просмотр** нестандартный, и **Специальный запрос** не определен.
- **Сортировка**. Настраивает правила сортировки, применимые к экземплярам. Доступны только если **Просмотр** нестандартный, и **Специальный запрос** не определен.
- **Связь**. Если список экземпляров класса показывается на панели инструментов другого класса, возможно отображать только те экземпляры, которые связаны с экземпляром, отображенным на текущей панели. В этом случае необходимо выбрать связь 888 для правильной фильтрации экземпляров.
- **Тип набора полей**. Определяет, будет ли показана преднастроенная группа полей или пользовательский набор полей.
- **Группа**. Отпределяет, какую группу полей отображать. Доступна, только если **Тип набора групп** является группой.
- **Поля**. Определяет отображаемые поля. Доступна, только если **Тип набора групп** является выбранными полями.
- **ID иконки**. ID иконки для отображения заголовка окна элемента.
- **ID справочного раздела**. ID справочного огтдела, на который должен ссылаться элемент.
- **Справка**. Текст многострочного информационного сообщения, которое появится над Редактором таблиц данных 382.
- **Показать панель инструментов**. Определяет видимость панели инструментов Редактора таблиц данных.
- **Показать заголовок**. Определяет видимость заголовка Редактора таблиц данных. Неопределенное значение активирует оптимальное поведение по умолчанию.
- **Показать номера строк**. Определяет видимость омеров строк Редактора таблиц данных. Неопределенное значение активирует оптимальное поведение по умолчанию..
- **Горизонтальная прокрутка**. Определяет, будет ли включена горизонтальная прокрутка по умолчанию. Неопределенное значение активирует оптимальное поведение по умолчанию.
- **Добавить запись**. Позволяет переписать "Добавить запись" в инструментальной панели списка экземпляров класса. Добавит новый экземпляр класса по умолчанию. Выбранное действие будет выполнено после переписывания.
- **Показывать результат добавления записи**. Включите данный параметр, чтобы показывать динамику действий "Добавить запись". Отключение параметра скроет подобные действия. Подробнее см. Интерактивные действия 100.
- **Параметры действия добавления записи**. Определяет параметры **Действия добавления записи**.
- **Действие удаления записи.** Позволяет переписать "Удалить ряд" в инструментальной панели списка экземпляров класса. Удалит выбранный экземпляр класса по умолчанию. Выбранное действие будет выполнено после переписывания.
- **Параметры действия удаления записи**. Определяет параметры **Действия удаления записи**.
- **Редактирование в новом окне**. Позволяет изменить ряд таблицы в отдельном окне.
- **Действие обновления записи**. Позволяет переписать действие "Изменить в новом окне". Выбранное действие будет выполнено после переписывания.
- **Параметры действия обновления записи**. Определяет параметры **Действия обновления записи**.

13.1.6.2 Форматы инструментальных панелей

Доступны два формата инструментальных панелей:

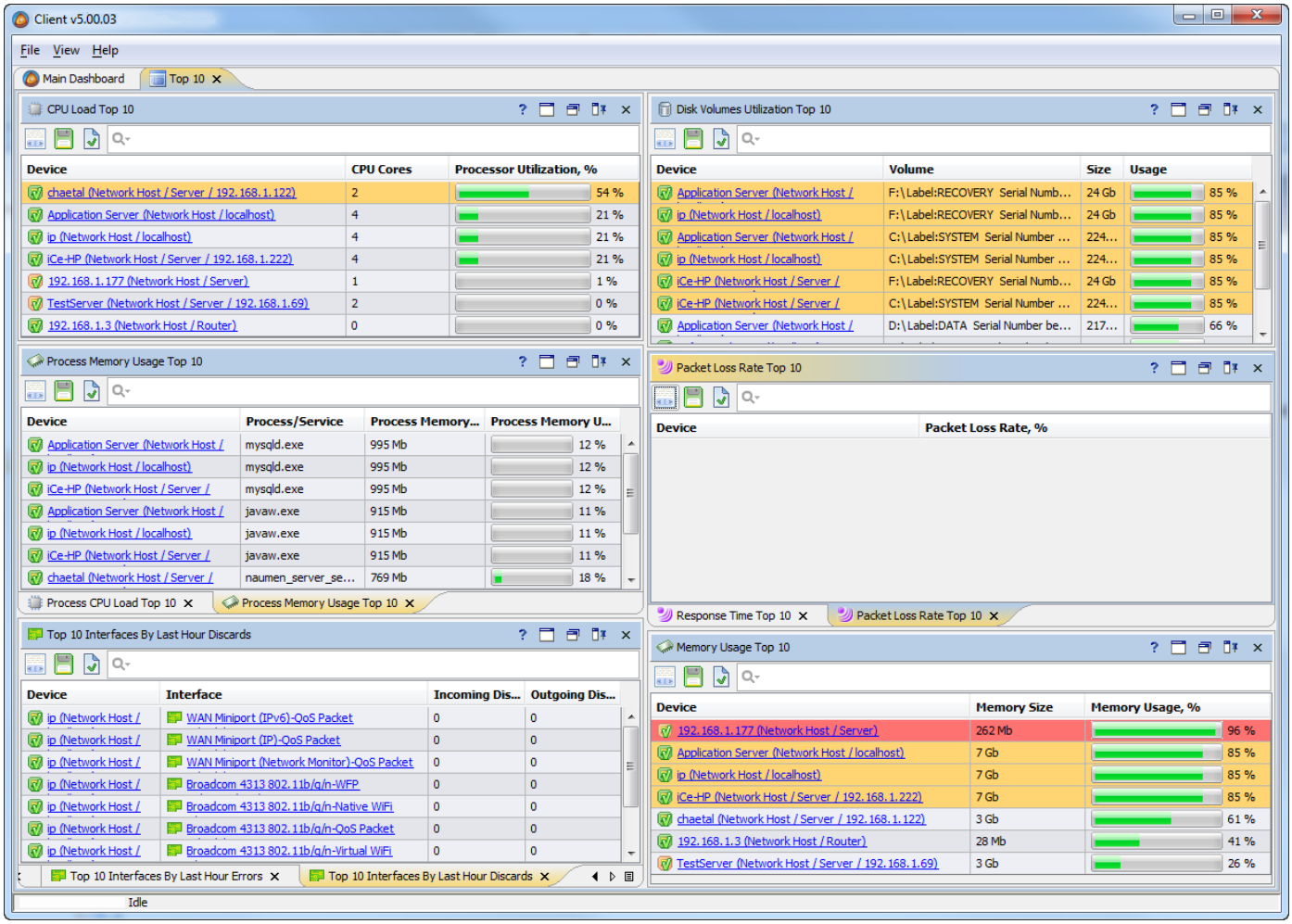
- Плавающие окна
- С прокруткой

Элемент положение окна 926 инструментальной панели по-разному интерпретируется в зависимости от формата панели.

Плавающие окна

В плавающих окнах все окна инструментальной панели помещаются на одном экране и, возможно, группируются в виде вкладок. Этими плавающими окнами можно управлять (закрывать, перемещать и др.), как описано в статье Настройка плавающих окон 366.

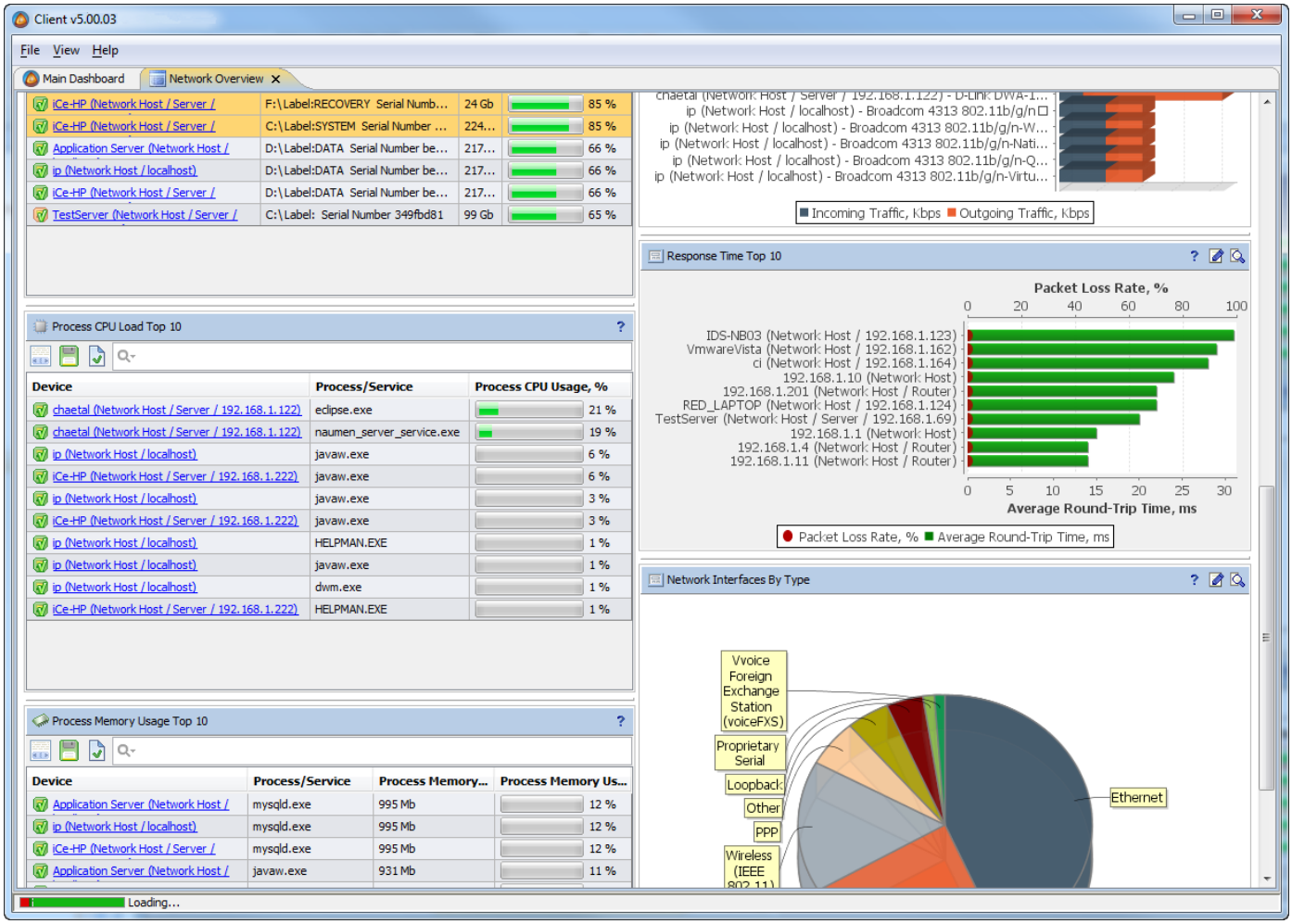
При данном формате все свойства для каждого положения элемента интерпретируются ровно так, как описано в разделе положение окна 926.



С прокруткой

Инструментальные панели с прокруткой помещают окна одно под другим в несколько колонок. Если все окна не помещаются на экране, появляется полоса вертикальной прокрутки.

При данном формате параметры **Состояние** и **Сторона**, определяемые в положении окна 926, игнорируются. Параметр **Индекс** определяет отсчитываемый с нуля индекс колонки, к которой добавляется окно. В каждой колонке окна появляются в том же порядке, что и в таблице Элементы инструментальной панели 918.



13.1.6.3 Таблица свойств инструментальной панели

Таблица свойств инструментальной панели позволяет задавать свойства инструментальной панели 912, которая разместит определенное окно на пользовательском интерфейсе. Эта таблица может обычно задаваться при конфигурировании местоположения 926 различных окон в пределах пользовательского интерфейса.

Свойства инструментальной панели включают следующую информацию:

- Имя и описание инструментальной панели
- Тип разметки 924 инструментальной панели
- Количество колонок (в инструментальной панели с прокруткой)
- Флажок, определяющий, можно ли закрыть инструментальную панель

Если параметры **Имя инструментальной панели** и **Описание инструментальной панели** определены, окно будет располагаться в инструментальной панели 912, не являющейся панелью по умолчанию. Инструментальные панели используются для группировки множества окон. Если инструментальная панель с данным именем не существует, она будет создана.

13.1.6.4 Положение окна

Настройки Положение Окна позволяют определить, где расположено окно на пользовательском интерфейсе.

Положение Окна включает следующую информацию:

- Имя и описание инструментальной панели
- Состояние панели (пристыкованное, плавающее, иконка на боковой панели, боковая панель)
- Строна панели (вверху, слева, внизу, справа)
- Индекс панели

- Ширина и высота
- Флажки контроля положения

Системные операторы могут двигать/пристыковывать/менять размер окон, а их состояние/расположение сохраняются в рабочем пространстве AtomMind Client. Таким образом, если некое окно открывается во второй раз, оно в большинстве случаев сохраняет предыдущие параметры расположения.

Инструментальная панель

Если определены параметры **Имя инструментальной панели** и **Описание инструментальной панели**, окно будет расположено в инструментальной панели 912 не по умолчанию. Инструментальные панели используются для группирования множества окон вместе. Если инструментальной панели с данным именем не существует, она будет создана.

Состояние, сторона и индекс

Комбинация параметров **Состояние**, **Сторона** и **Индекс** определяют расположение окна на инструментальной панели.

Ниже приведенная таблица показывает, как эти параметры работают вместе для плавающей инструментальной панели 924:

Состояние	Сторона	Индекс	Комментарии
Пристыкованное	сверху, слева, снизу, справа	Любое целое число больше 0	Фреймы с тем же режимом, стороной и индексом формируют единую панель с вкладками.
Боковая панель	сверху, слева, снизу, справа	Любое целое число больше 0	Фреймы с той же стороной и индексом формируют группу на боковой панели.
Плавающее	не доступно	Любое целое число больше 0	Фреймы с тем же режимом и индексом формируют панель с вкладками и находятся в том же плавающем окне.

Для инструментальной панели с прокруткой 924 параметры **Состояние** и **Сторона** игнорируются, в то время как параметр **Индекс** определяет индекс на основе нуля того ряда, к которому нужно добавить окно.

Ширина и высота

Если определены **Ширина** и/или **Высота**, они обрабатываются следующим образом:

- Если окно пристыковано, менеджер стыкуемых окон наилучшим образом подстраивает размеры всех окон. Окно не будет иметь абсолютно тот же размер, который определен параметрами **Ширина**/**Высота**.
- Если окно находится на боковой панели, оно будет придерживаться **Ширины**, если расположить его в левой/правой панели, и будет придерживаться **Высоты**, если расположить его в верхней/нижней панели.
- Если окно плавающее, у него будет размер, определенный параметрами **Ширина**/**Высота**.

Флажки контроля положения

Положение окна включает следующие дополнительные флажки контроля положения/размера:

- Разрешить изменение размера
- Разрешить закрытие
- Разрешить пристыковывание
- Разрешить плавающую функцию
- Разрешить разворачивание
- Разрешить пристыковывание к боковой панели

Показать заголовок

Этот флажок управляет видимостью заголовка окна. Если он не показан, все кнопки управления положением окна будут недоступны.

Window Key

Параметр **Ключ** используется для управления уникальным именем окна (window key). Имя окна может быть доступно из других частей пользовательского интерфейса.

13.2 Отчеты

Отчеты формируются для показа данных в графическом формате, пригодном для печати. Без отчетов не могут обойтись приложения и системы с высокоразвитой обработкой данных и возможностями аналитики. В AtomMind любая таблица данных, извлеченная из единой модели данных, может использоваться для создания отчета:

- Данные, поступающие с аппаратных устройств 497 (например, статистика интерфейса маршрутизатора сети)
- Свойства ресурсов системы
- Результаты запросов 829
- Исторические значения свойства или события, выбранные по критериям пользователя
- Данные, создаваемые скриптом 879

Создавать отчеты очень просто. Не нужно ничего программировать. Сначала вы используете Редактор выражений 404, чтобы создать выражение исходных данных 929 (для извлечения данных из системы). Затем AtomMind Server генерирует шаблон отчета для представления этих данных согласно вашим предпочтениям по размеру бумаги, шрифту, цветам, ширине столбцов, группировке и т.д. Наконец, можно использовать Редактор отчетов 416, чтобы окончательно настроить ваш отчет, добавить логотипы, подобрать цвета и шрифты, изменить формат данных и т.д.

Просмотр и печать отчетов осуществляется при помощи Просмотра отчетов 415. Любой отчет можно экспортировать в:

- PDF (Adobe Acrobat)
- RTF (Rich Text Format)
- ODT (Open Office)
- HTML (Hypertext Markup Language)
- XLS (Microsoft Excel)
- CSV (Character Separated Values)
- XML (Extensible Markup Language)

В большинстве случаев отчеты создаются пользователями подобно тревогам 779, запросам 829 и другим системным объектам. Однако некоторые отчеты предопределены в системном дистрибутиве и не могут быть изменены пользователем. Эти отчеты встроены в пакет дистрибутива AtomMind и плагины 207.

Каждый пользователь 478 имеет свой набор отчетов.

Каждый пользовательский отчет характеризуется двумя важными свойствами:

- **Шаблон отчета**, который определяет внешний вид и формат отчета
- **Выражение входных данных,** которое используется для извлечения данных и внесения их в шаблон отчета

Документация по теме:

- Построение отчета о статусе устройства 1678
- Запланированная отправка e-mail отчетов 1682

Администрирование отчетов

Для администрирования отчетов используются два контекста: общий контекст Отчеты 1551, который является контейнером, и контекст Отчет 1553, который содержит информацию об одном отчете.

Предопределенные отчеты могут не иметь всех настроек и действий, которые доступны для пользовательских отчетов.

13.2.1 Выражение исходных данных

Выражение исходных данных является выражением AtomMind 112, которое вычисляется каждый раз, когда запускается отчет и показывается пользователю. Данное выражение **должно** вычисляться в таблицу данных 49. Состояние ошибки возникнет, если значение выражения будет простого типа, например, логическое, целое число или строка. Данные из таблицы данных, полученные в качестве результата выражения, используются для заполнения шаблона отчета или подготовки самого отчета.

Для относительных 932 отчетов данное выражение может содержать ссылки 117 с соответствующими путями контекста 41, любые соотносимые ссылки будут разрешаться относительно контекста, в котором было выбрано действие Запустить отчет 933. Если отчет был запущен с использованием действия Показать отчет 1555 или из Редактора отчетов 416, связанные ссылки будут разрешаться относительно контекста, обозначенного свойством отчета Контекст по умолчанию 935.

Среда вычисления 114 выражения исходных данных:	
Контекст по умолчанию 119	Для абсолютного 932 отчета - сам контекст отчета. Для относительного 932 отчета - контекст, для которого запускается отчет.
Таблица данных по умолчанию 120	Таблица параметров отчета (если отчет параметризованный 938).
Ряд по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

Пример: Если формируется отчет о данных, полученных в результате запроса 829, выражение данных источника должно содержать ссылку на переменную **данных** в контексте запроса 1548. Итак, если запрос называется **ds_traffic_stats** и принадлежит пользователю **admin** (это предопределенный запрос 832), мы можем получить следующее выражение данных источника:

```
{users.admin.queries.ds_traffic_stats:data}
```

Пример: Перед вами пример выражения данных источника относительного `Attendance Report`:

```
{attendance:attendanceData('{.:}')}
```

Данное выражение источника данных ссылается на функцию 70 **attendanceData** из контекста **attendance.** В процессе оценки вызов данной функции осуществляется одним параметром, путь контекста из которого было выбрано действие запуска 933 **Attendance Report** (или путь контекста по умолчанию 929, если отчет был запущен в тестовом режиме из редактора отчетов 416). Объяснение, почему `{.:}` разрешается в путь контекста по умолчанию, см. в разделе Стандартные ссылки 118.

Пример: Представленное ниже выражение относительного отчета **История Тревоги** загружает исторические 75 события тревоги 790 из базы данных и возвращает их в таблицу. Лишь несколько полей тревоги, выбранных функцией `subtable()`, включаются в отчет.

```
subtable({events:get('{.:}', "alert", null, null, false, null, false)},
"eCreationtime", "eLevel", "cause", "message", "trigger", "eAcknowledgements")
```

См. руководство Отбор и Обработка Событий 1699 для получения более подробной информации.

Контекст по умолчанию для отчета

Говоря простым языком, **контекст по умолчанию** представляет собой свойство отчета, используемое для разрешения ссылок в выражение источника данных 929, когда при запуске относительного 932 отчета не указана "цель". Он используется в двух случаях:

- При запуске относительного 932 отчета напрямую, без указания "целевого" контекста.
- При построении отчета в тестовом режиме из редактора отчетов 416.

Когда запуск связанного отчета осуществляется через действие Запустить отчет 933 ("классический" способ), пути относительного контекста, содержащиеся в ссылках внутри выражения данных источника 929, разрешаются соответственно в путь контекста, из которого было инициировано действие "Запустить отчет". Свойство Контекст по Умолчанию в этом случае не используется.

13.2.2 Шаблон отчета

Шаблон отчета определяет формат страницы и различные преобразования, касающиеся данных источника, при выполнении отчета. Шаблон задан в формате XML и сохраняется как свойство 935 отчета. Шаблон создается 930 автоматически в процессе формирования отчета. Данный автоматически создаваемый шаблон основан на данных, выбранных из выражения данных источника. Существует два способа изменения шаблона существующего отчета:

- Сразу в формате XML, используя действие Конфигурировать отчет 1553 в контексте отчета.
- Используя редактор отчетов 416, выполнив действие Редактировать шаблон отчета 1553 в контекста отчета.

Отчеты формируются, заполняются и обрабатываются редактором JasperReports. Для более подробной информации о структуре шаблона, доступных элементах и их свойствах обратитесь к руководству по Jasper Reports.

13.2.3 Генерация шаблонов отчета

AtomMind Server способен автоматически генерировать шаблоны отчета 928 из данных, содержащихся в любой таблице данных 49. Перед генерацией шаблона, есть возможность указать его свойства:

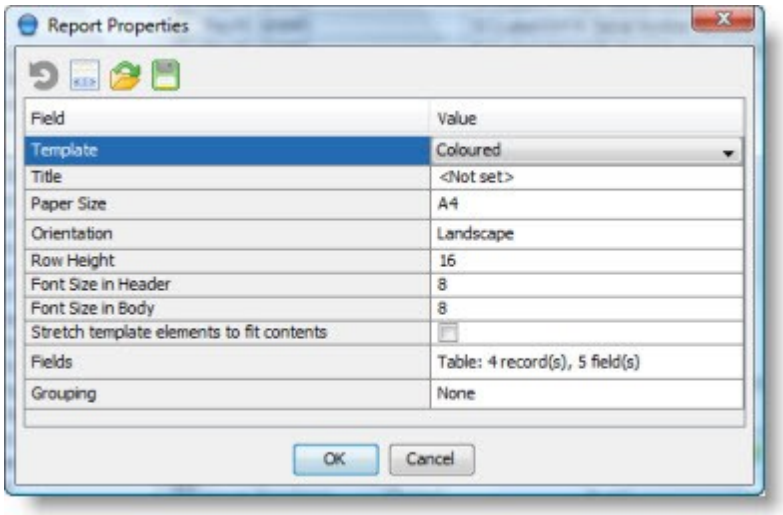
Свойство	Имя	Тип	Описание
Шаблон	template	Integer	Тип шаблона отчета: **простой** (удобен для черно-белой печати) или **цветной**.
Заголовок	title	String	Заголовок отчета. Появляется в начале первой страницы.
Размер бумаги	paper	String	Указывает размер бумаги для печати.
Ориентация	orientation	Integer	Книжная/альбомная ориентация для печати.
Высота ряда	rowHeight	Integer	Высота каждого ряда в основной части отчета.
Размер шрифта заголовка	headerFontSize	Integer	Размер шрифта для заголовков колонок.
Размер шрифта основной части	bodyFontSize	Integer	Размер шрифта для содержимого отчета.
Растягивать элементы шаблона под размер содержимого	stretch	Boolean	Показывает, что высота ряда может быть увеличена для внесения элементов, которые не вмещаются в стандартный размер ряда. Если это свойство отключено, элементы, которые не вмещаются в свои ряды, будут скрыты.
Поля	fields	Data Table	Конфигурация полей отчета: • **Видимость**. Флажок, показывающий, что поле таблицы данных будет показано в отчете. • **Относительная ширина**. Плавающее значение точек дает подсказку для вычисления ширины колонки. Если относительная ширина одной колонки в два раза больше относительной ширины другой, то первая колонка будет в два раза шире в шаблоне отчета. • **Заголовок колонки**. Заголовок колонок отчета.
Группировка	grouping	String	Если включено, строки отчета будут объединены выбранным полем.
Формат отчета	reportFormat	String	Формат файла создаваемого отчета. Возможные значения: pdf, rtf, odt, html, xls, csv, xml.

Пример

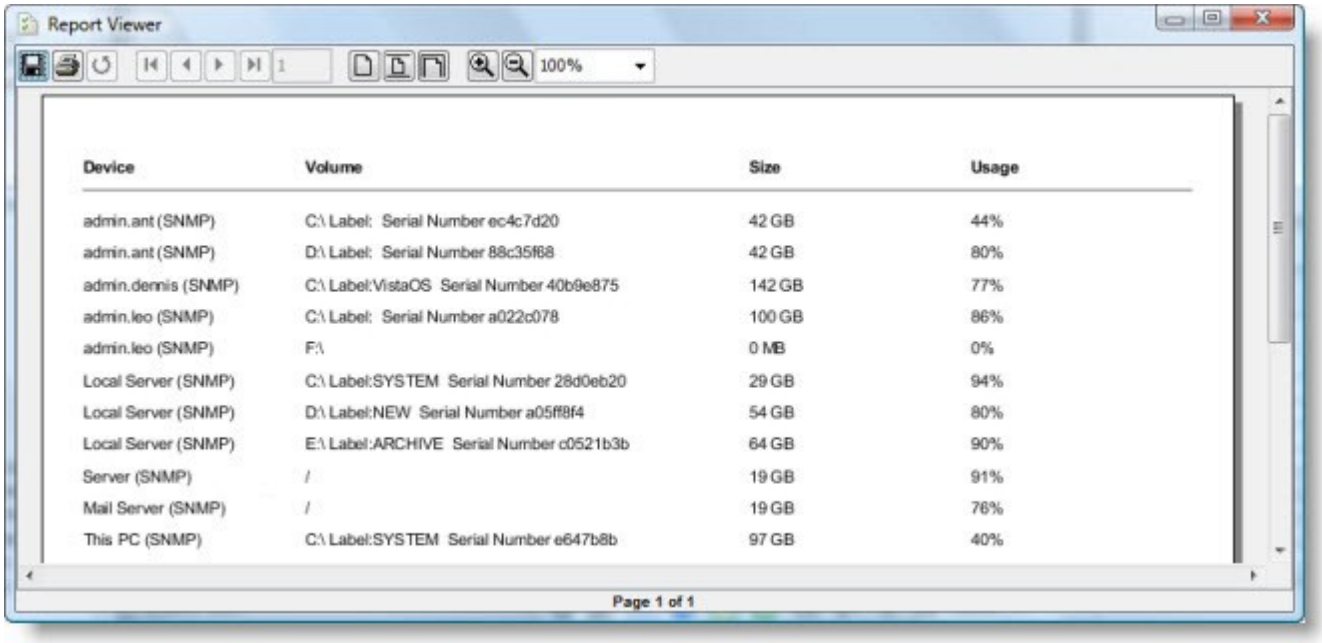
Исходные данные для отчета:

Device	Volume	Size	Usage
admin.ant (SNMP)	C:\ Label: Serial Number ec4c7d20	42 GB	44%
admin.ant (SNMP)	D:\ Label: Serial Number 88c35f68	42 GB	80%
admin.dennis (SNMP)	C:\ Label:VistaOS Serial Number 40b9e875	142 GB	77%
admin.leo (SNMP)	C:\ Label: Serial Number a022c078	100 GB	86%
admin.leo (SNMP)	F:\	0 MB	0%
Local Server (SNMP)	C:\ Label:SYSTEM Serial Number 28d0eb20	29 GB	94%
Local Server (SNMP)	D:\ Label:NEW Serial Number a05ff8f4	54 GB	80%
Local Server (SNMP)	E:\ Label:ARCHIVE Serial Number c0521b3b	64 GB	90%
Server (SNMP)	/	19 GB	91%
Mail Server (SNMP)	/	19 GB	76%
This PC (SNMP)	C:\ Label:SYSTEM Serial Number e647b8b	97 GB	40%
This PC (SNMP)	D:\ Label:DATA Serial Number 3e25e3c3	182 GB	44%
This PC (SNMP)	E:\ Label:ARCHIVE Serial Number da5863fa	185 GB	24%
Web Server (SNMP)	/	19 GB	91%

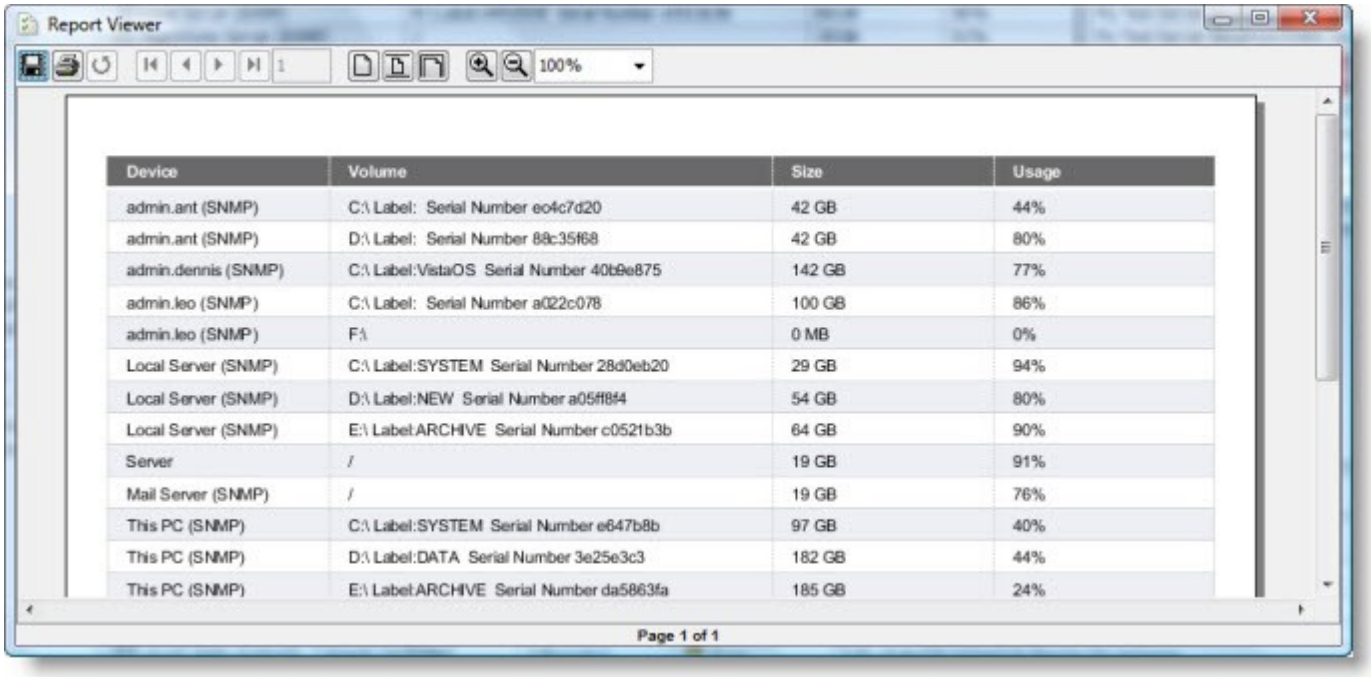
Параметры отчета:



Итоговый отчет (простой шаблон):



Итоговый отчет (цветной шаблон):



13.2.4 Типы отчетов

Отчеты подразделяются на две категории:

- **Абсолютные** отчеты
- **Относительные** отчеты

Абсолютные отчеты

Абсолютные отчеты обычно обрабатывают данные одного какого-либо источника (обычно одного контекста). Например, отчет об учетных записях пользователей AtomMind Server показывает список учетных записей пользователей 478 внутри всей системы. Данный отчет должен быть абсолютным.

Выражение данных источника 929 абсолютного отчета не может иметь относительных ссылок 117.

Относительные отчеты

Относительные отчеты предназначены для обработки данных определенного контекста, выбираемого оператором при запуске 932 отчета. Это оказывается полезным при создании отчета для данных, взятых из одного источника системы или Device. Относительные отчеты обычно активируются действием Запустить отчет 933 в том контексте, чьи данные подлежат обработке. Например, в системе сетевого управления может быть относительный отчет **Статистика использования интерфейса роутера**. Чтобы посмотреть этот отчет, нажмите правой кнопкой мыши на Device **Роутер** в AtomMind Client и в контекстном меню выберите **Статистика использования интерфейса роутера**.

Относительные отчеты устанавливают Действие Запуска только в контексты, доступные для владельца отчета в соответствии с его правами доступа 477.

13.2.5 Запуск отчетов

При запуске отчета AtomMind Server выполняет выражение данных источника 929, использует полученные данные для заполнения шаблона 930 отчета и показывает подготовленный отчет при помощи Пользовательской процедуры Показать отчет 97.

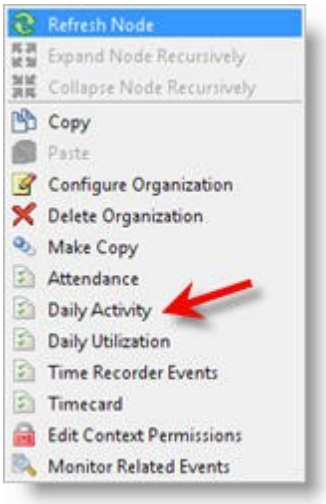
Запуск отчета может осуществляться одним из приведенных ниже способов:

- Напрямую, используя действие Показать отчет 1555 в контексте отчета. Для относительных 932 отчетов данный метод оценивает выражение данных источника 929 относительно контекста, заданного свойством отчета "Контекст по умолчанию" 935.
- При помощи действия 87 **Запустить отчет** (см. далее). Для относительных 932 отчетов данный метод оценивает выражение данных источника 929 относительно контекста, из которого запускается данное действие.

ДЕЙСТВИЕ ЗАПУСК ОТЧЕТА

Данное действие можно найти в любом контексте, для которого выражение пригодности 932 является **true**. Например, если создается **Отчет посещаемости** для контекста **Владелец карточки** (т.е. выражение пригодности является `{.#type} == 'cardholder'`), в контексте каждого владельца карточки появляется действие `Attendance Report`. Его описание совпадает с описанием отчета. Действия, используемые для запуска отчета, можно легко узнать по иконке  . Пожалуйста, обратитесь к разделу Ссылки 117 для подробного описания принципа работы ссылки `{.#type}`.

Действие Запуск отчета в контекстном меню AtomMind Client выглядит таким образом:

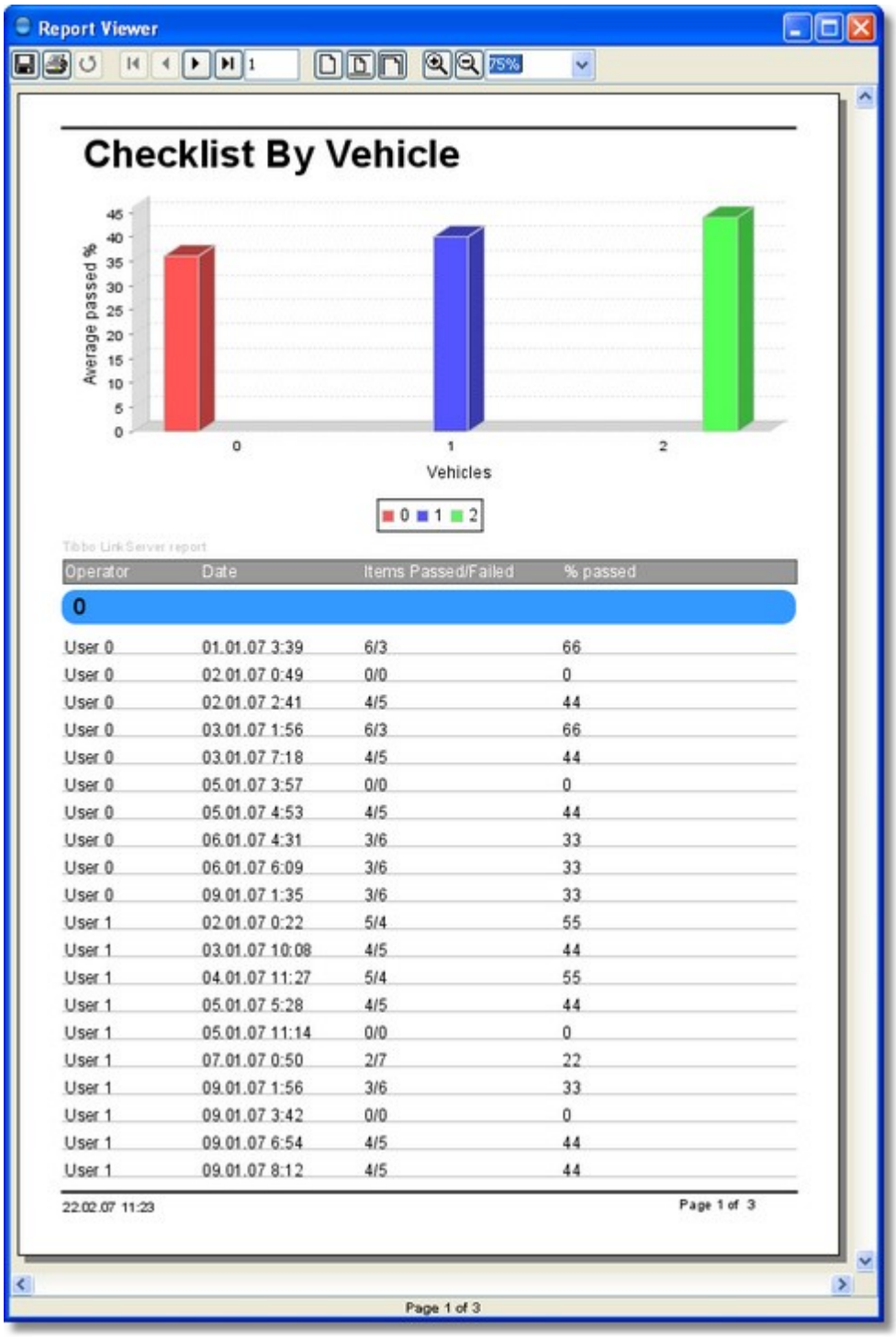


Чтобы сделать данное действие доступным, пользователь должен обладать действующими правами доступа 489 в поле **Пользователь** в двух контекстах:

- Контекст запускаемого отчета 1553
- Контекст, на основе которого работает отчет, т.е. в котором определено действие Запустить отчет.

13.2.6 Просмотр отчетов

Посмотреть отчет можно, выполнив действие Показать отчет[1553] в его контексте. Данное действие запрашивает параметры, заполняет шаблон отчета данными, полученными из различных контекстов[41] и представляет пользователю подготовленный отчет:



Отчет может быть распечатан или извлечен в файл. Более подробно об этом можно узнать в описании Пользовательской процедуры "Показать отчет[97]".

Просмотр созданных ранее отчетов

Если включена опция Сохранить историю[935], возможно просматривать, эскпортировать, печатать и управлять ранее созданными экземплярами отчета.

Чтобы открыть список ранее созданных отчетов, запустите действие **Просмотреть историю** Контекста отчета[1553].

13.2.7 Конфигурация отчета

Данный раздел посвящен свойствам конфигурации отчета.

Доступ к данным свойствам осуществляется путем выполнения действия "Настроить 1051" в контексте Отчет 1553.

13.2.7.1 Свойства отчета

Каждый отчет имеет следующие основные свойства:

Описание поля	Имя поля
Имя. Имя контекста отчета, используемое для ссылки на него из других частей системы. Должно соответствовать соглашению об именах 42 контекстов.	name
Описание. Текстовое описание 43 контекста отчета.	description
Параметризован. Флажок, указывающий, что отчет параметризованн 938.	parameterized
Выражение для получения исходных данных. Выражение 112, используемое для выборки данных при заполнении шаблона отчета. Среда вычисления 114 Выражения для получения исходных данных: Контекст по умолчанию 119 — Контекст, к которому прикрепляется отчет согласно **Выражению пригодности**. **Контекст по молчанию,** если отчет проходит тестирование. Таблица данных по умолчанию 120 — Таблица параметров параметризованного отчета 938. Ряд по умолчанию 119 — 0 Переменные среды 123 — Только стандартные 123 переменные.	expression
Исходные данные параметризатора. Данное поле используется только при включенном флажке **параметризован**. Исходные данные параметризатора используются процессором параметризации для построения выражения данных источника во время запуска отчета, основанного на определяемых пользователем параметрах. Более подробную информацию смотри в разделе параметризованные отчеты 938.	parameterizer
Шаблон отчета. Шаблон отчета в формате XML.	template
Сохранить историю. Если включена эта опция, полная копия отчета сохраняется в базе данных 692 сервера каждый раз, когда запускается отчет. Чтобы посмотреть все сохраненные копии отчета, запустите действие **Просмотреть историю** из контекста отчета 1553.	saveHistory
Тип отчета. Тип 932 отчета, **относительный** или **абсолютный**.	type
Выражение пригодности. Определяет, какой контекст (или контексты) отчет сможет "понять". Болле подробно об этом см. в разделе Выражение пригодности 932.	validityExpression
Правила обновления пригодности. Список масок контекста и имен событий. Если событие, определенное полем **Событие** данной таблицы, происходит в контексте, соответствующем полю **Маска** той же записи, **Выражение Пригодности** будет заново рассчитано для данного контекста. Это позволяет сделать отчет пригодным/непригодным для определенного контекста в случае внесения в него некоторых изменений.	validityListeners
Контекст по умолчанию. Контекст по умолчанию, используемый для оценки выражения данных источника. См. раздел Контекст по умолчанию для отчета 929 для более подробной информации.	defaultContext

Расположение по умолчанию. Положение окна 926 отчета используется, когда отчет показывается вручную, т.е. без использования Автозапуска 941, Избранного 2186 или других средств.  Когда действие Показать отчет 1555 добавлено в Автозапуск избранного, его параметры выполнения 102 позволяют перезаписать **Расположение по умолчанию** и открыть окно отчета на пользовательской панели инструментов/позиции.	defaultLocation
Инструментальная панель. Данная опция позволяет настраивать запуск отчета как инструментальной панели.	defaultDashboard

Данные свойства доступны через переменную childInfo 1556.

13.2.7.2 Дополнительные параметры отчета

Данная таблица определяет дополнительные параметры, используемые для заполнения шаблона отчета. Имена данных параметров **должны совпадать с именами параметров отчета, используемых при его выполнении.**

Значение каждого параметра рассчитывается выражением 112, вводимым в поле "Значение".

Описание поля	Имя поля
Параметр. Имя параметра.	parameter
Значение. Выражение, используемое для расчета значения параметра.	value

Дополнительные параметры отчета доступны через переменную parameters 1557.

Среда вычисления

Среда вычисления 112 используется для расчета значений дополнительных параметров отчета:	
Контекст по умолчанию 119	Контекст запускаемого отчета.
Таблица данных по умолчанию 120	Результат параметризации 938 отчета. Это означает, что ссылки в выражении **Значения** могут относиться к полям, определяемым форматом параметров 1441 исходных данных параметризатора.
Ряд по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

Встроенные дополнительные параметры

Ката параметров каждого отчета включает несколько параметров, которые позволяют получить доступ к структурам данных на стороне сервера. Эти параметры помогают решать сложные задачи, такие как создание подотчета, и т.д.

Имя параметра	Тип параметра	Описание параметра
contextManager	ContextManager	Менеджер контекстов сервера. Более подробно см. в статье Работа с контекстами 1378 раздела Расширение и интеграция платформы 1339.
reportContext	String	Путь контекста отчета.
defaultContext	String	Путь контекста отчета по умолчанию.
callerController	CallerController	Объект вызывающего контроллера, заключающий права доступа пользователя, запустившего отчет. Более

		подробно см. в статье Управление правами доступа 1392 раздела Расширение и интеграция 1339.

13.2.7.3 Подотчеты

Подотчеты могут быть использованы для встраивания нескольких отчетов в один главный отчет.

Каждый подотчет должен быть прикреплен к главному с помощью создания записи в свойстве **подотчеты** отчета 928. Это свойство имеет следующие поля:

Описание поля	Имя поля
Имя. Уникальное имя подотчета. По этому имени к подотчету будут ссылаться из исходного шаблона отчета.	name
Тип шаблона. Тип шаблона 930 подотчета. **Static** означает, что шаблон получен от контекста отчета, выбранного в поле **Шаблон**. **Динамический** означает, что шаблон вычисляется из поля **Выражение шаблона**.	templateType
Шаблон. Контекст отчета 928 будет использован как шаблон подотчета для **статического** подотчета.	template
Выражение шаблона. Выражение 112, которое должно вернуть шаблон 930 для подотчета в случае **динамического** типа шаблона. Он должен вычисляться для XML-строки.	templateExpression
Тип данных. Тип исходных данных 929 . **Статический** означает, что данные получены от контекста отчета 928, выбранного в поле **Шаблон**. **Динамический** означает, что данные вычисляются из поля **Выражение данных**.	dataType
Выражение данных. Выражение 112, которое должно вернуть таблицу данных 49, которая будет использована как исходные данные для подотчета.	dataExpression

Среда вычисления

Среда вычисления 114 **выражения шаблона** и **выражения данных** :	
Контекст по умолчанию 119	Для абсолютного 932 отчета, сам контекст отчета. Для относительного 932 отчета, контекст, для которого запущен отчет.
Таблица данных по умолчанию 120	Таблица параметров отчета (если отчет параметрезован 938). Это означает, что ссылки в выражении **значения** могут относиться к полям, определенным форматом параметров 1444 исходных данных параметризатора.
Ряд по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

Использование подотчетов в редакторе отчетов

В то время как AtomMind хранит шаблоны отчетов внутри своей логической структуры, существуют особенности работы с подотчетами в редакторе отчетов 416. Особый параметр отчета **ХРАНИЛИЩЕ** используется для доступа к шаблонам подотчетов, источникам данных и ресурсам. Для качественной настройки компонента подотчетов после его добавления к шаблону отчета, вам следует установить следующие свойства этого компонента:

- **Выражение подотчета**. Это выражение определяет, каким образом получен шаблон подотчета. Должно быть использовано выражение `$P{STORAGE}.subReport("<subreportName>")`, где `<subreportName>` - это имя подпорта из свойства таблицы **подпорты**.
- **Тип подключения**. Должен быть установлен на `Use a datasource expression`. Это позволяет определять выражение источника данных.
- **Выражение источника данных.** Это выражение возвращает источник данных подотчета. Должно быть использовано выражение `$P{STORAGE}.dataSource("<subreportName>")`, где `<subreportName>` - это имя подотчета из свойства таблицы **подотчеты**.

13.2.7.4 Ресурсы

К отчетам помут прикрепляться различные ресурсы, такие как логотип компании. **Ресурсы** добавляются к соответствующему свойству отчета[928]. Это свойство имеет следующие поля:

Описание поля	Имя поля
Тип. Тип ресурса. **Статический** означает, что ресурс определяется полем **данные ресурса**. **Динамический** означает, что ресурс определяется полями **имя** и **выражение**.	type
Данные ресурса. Данные ресурса, в случае если он имеет **статический** тип. Данные загружаются из файла, включая имя файла. После того как файл загружен, имя файла становится идентификатором ресурса, который может быть использован в шаблоне отчета ресурсом **динамического** типа.	resource
Имя. Уникальное имя ресурса. Используется только для **статического** типа ресурса. По этому имени на ресурс ссылаются в шаблоне отчета (для **динамического** типа ресурса).	name
Выражение. Выражение[112] должно вернуть массив байтов. Он будет использован для **динамического** типа ресурса.	expression

Среда вычисления

Среда вычисления[114] **выражения**:	
Контекст по умолчанию[119]	Для абсолютного[932] отчета, сам контекст отчета. Для относительного[932] отчета, контекст, для которого запущен отчет.
Таблица данных по умолчанию[120]	Таблица параметров отчета (если отчет параметрезован[938]). Это означает, что ссылки в выражении **значения** могут относиться к полям, определенным форматом параметров[1444] исходных данных параметризатора.
Ряд по умолчанию[119]	0
Переменные среды[123]	Только стандартные[123] переменные.

Использование ресурсов в редакторе отчетов

Для доступа к ресурсам в редакторе отчетов[416] используется особый параметр отчета **ХРАНИЛИЩЕ**. Для этого может использоваться компонент изображений. Для качественной настройки компонента изображений после его добавления к шаблону отчета, вам следует установить следующие свойства этого компонента:

- **Выражение изображения**. Это выражение определяет, каким способом данный компонент получает свои данные. Должно быть использовано выражение `$P{STORAGE}.resource("<resourceId>")`, где `<resourceId>` - это **имя** ресурса или имя файла **данных ресурса**.
- **Класс выражения**. Ожидаемый тип возврата выражения ресурса. Должен быть установлен на `java.io.InputStream`.

13.2.8 Параметризованные отчеты

Параметризация отчета может быть полезной, когда вы хотите позволить оператору конфигурировать отчет во время его выполнения. Если установлен флажок **Параметризован** в свойствах[935] отчета, пользователю может быть предложено ввести один или более параметров при запуске[932] отчета. Запрос параметров отчета осуществляется при выполнении пользовательской процедуры Изменить данные[90].

Параметризация является сложной темой, которая выходит далеко за рамки данной статьи. Однако, чтобы понять этот раздел, вам нужно иметь четкое представление о процессе и механизме параметризации. Поэтому, если вы хотите использовать параметризованные фильтры и вы не до конца усвоили данный материал, рекомендуем вам перейти к теме Механизм параметризации[1441], прочитать всю статью до полного ее понимания, а затем вернуться в данный раздел.

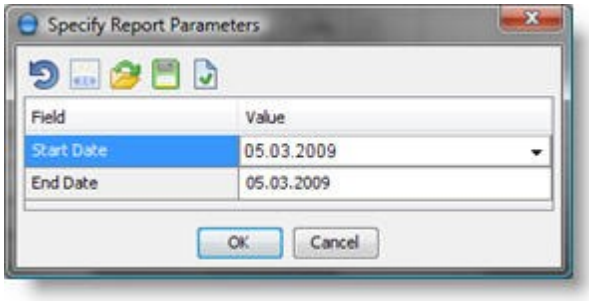
Когда в свойствах отчета включена опция **Параметризован**, процесс обработки использует **данные источника параметризатора**, чтобы сформировать выражение данных источника[929] в реальном времени, основанное на

заданных оператором параметрах. Для более подробной информации обратитесь к разделу Механизм параметризации 1444.

Далее приведен пример Данных источника параметризатора с двумя полями даты в **Формате** (startDate и endDate) и **Праметризованном выражении**:

```
{attendance:timeRecorderEvData('{.:}',"<e>{startDate}</e>", "<e>{endDate}</e>")}
```

Отчет, в состав которого входит данное **параметризованное выражение**, будет запрашивать при активации **дату запуска** и **дату окончания**:



Конечное выражение данных источника, используемое для формирования отчета, будет выглядеть примерно таким образом:

```
{attendance:timeRecorderEvData('{.:}',"2009-05-01 00:00:00.000", "2009-05-02
00:00:00.000")}
```

13.2.9 Безопасность отчетов

Отчет Выражение данных источника 929 разрешается путем использования прав доступа 477 пользователя 478, запускающего отчет. Таким образом, только данные, доступные этому пользователю, используются для заполнения отчета.

Относительные 932 отчеты добавляются только к контекстам, доступным при наличии прав доступа владельца отчета.

13.2.10 Производительность отчетов

Тревоги - это "пассивные" компоненты сервера, не вызывающие постоянной нагрузки процессора. Влияние на производительность, вызванное процессом формирования отчета, отпределяется следующим:

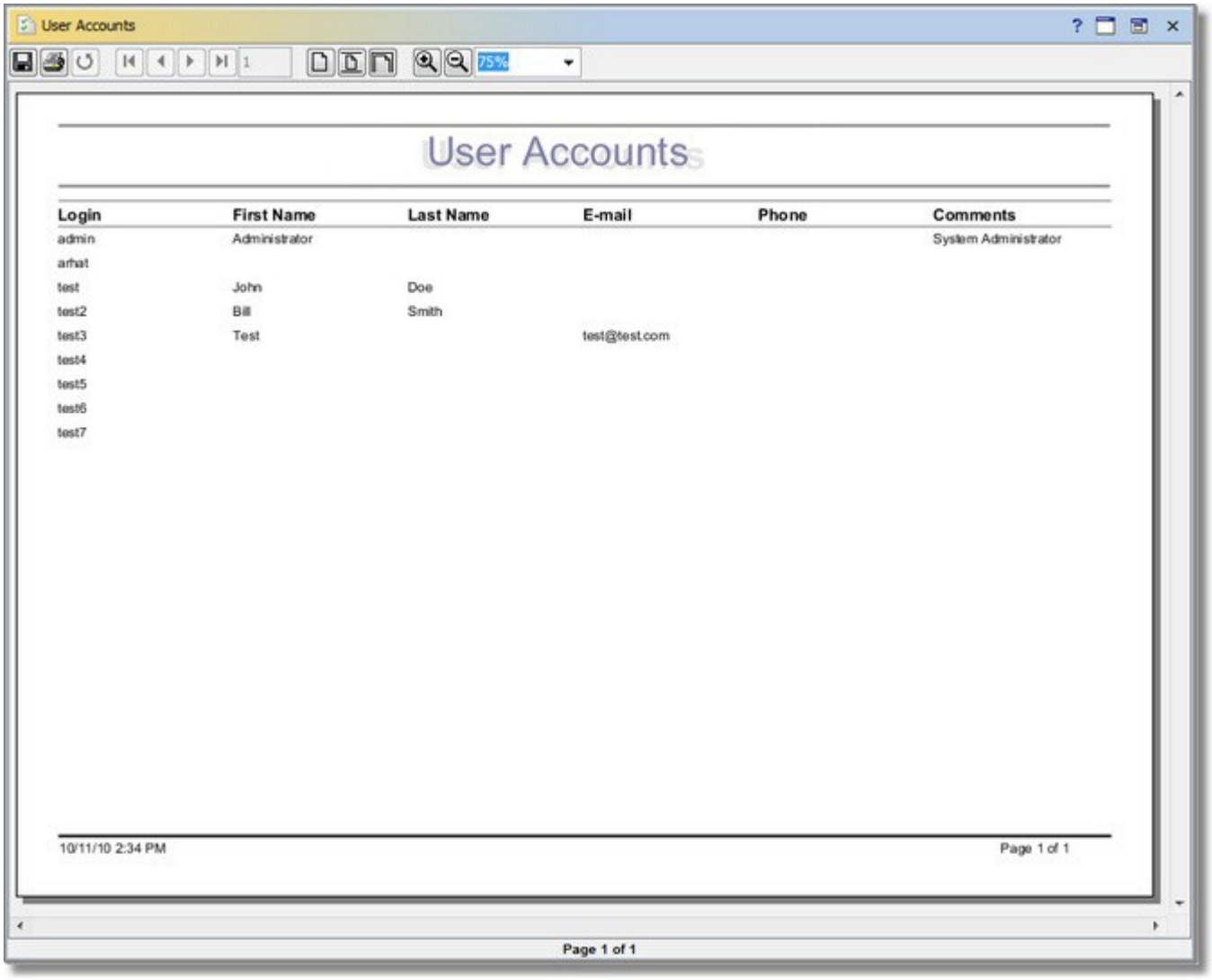
- Влияние на производительность 141 оценки Выражения Данных Источника отчета. Отчеты часто формируются на основе большого пакета данных, требуя значительного объёма памяти для функционирования.
- Дополнительное время процессора и память, требуемые для заполнения шаблона отчета. Это лишь малая часть использования ресурсов, требуемая для подготовки пакета данных, и в редких случаях действительно сложные шаблоны могут увеличить влияние процесса заполнения шаблона.

13.2.11 Встроенные отчеты

Базовый дистрибутив AtomMind Server содержит несколько встроенных отчетов:

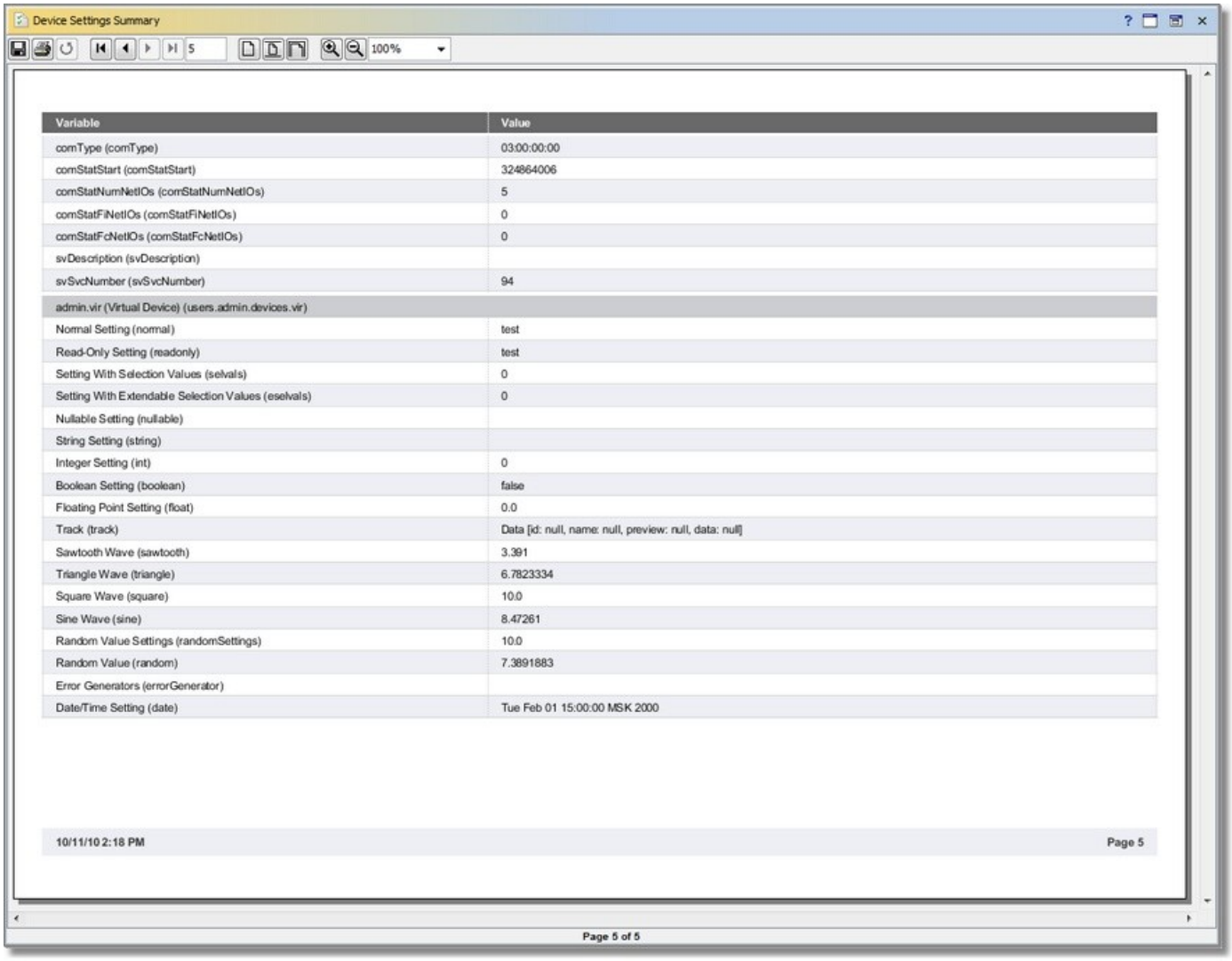
Учетные записи пользователей

Данный отчет содержит информацию о всех учетных записях 478 AtomMind Server, которая предоставляется пользователю, просматривающему отчет:



Обзор настроек устройства

Данный отчет представляет обзор настроек всех Device, состоящих из одной ячейки (т.е. не в виде таблицы или массива):



История тревог

Данный отчет показывает историю определенной тревоги, а именно:

- Дату/время тревоги
- Уровень тревоги
- Причину тревоги
- Сообщение тревоги
- Сообщение триггера
- Подтверждения, если есть

Данный отчет является относительным 932, он пригоден для любого контекста тревоги 1454.

13.3 Автозапуск

Функция "Автозапуск" обеспечивает автоматическое выполнение действий 87 при входе в интерфейс пользователя AtomMind Server, например, в AtomMind Client. Действия автозапуска являются полезными при создании "инструментальных панелей", которые содержат виджеты 943, запросы 829 и отчеты 928, отображаемые автоматически на определенных заранее позициях, как только пользователь запускает AtomMind Client. Другое применение автозапуска - автоматическое отображение статуса критически важного устройства 497.

Действия запускаются в том же порядке, в каком они появляются в контексте "Автозапуск". Последовательность потомков данного контекста может быть изменена. Пользователи AtomMind Client для этого могут использовать функцию переупорядочивания 374 системного дерева, в то время как другие интефейсы пользователя AtomMind Server обеспечивают подобные функции.

В отличие от действий, которые запускаются "напрямую" (например, из системного дерева 370 AtomMind Client), действия, запускаемые при помощи средства "Автозапуск", могут использовать параметры выполнения 102.

Каждый пользователь 478 имеет свои действия автозапуска.

Сопутствующая документация: Создание инструментальной панели для мониторинга устройств в реальном времени 1660

Переупорядочивание действий автозапуска

Возможно изменить порядок выполнения действий автозапуска, используя Системное дерево 370 AtomMind Client:

- Откройте узел **Автозапуск** в **Пользователях => *Your_Username***
- Перетащите одно из действий и поместите его между двумя другими действиями (либо в начале/конце списка)

См. раздел Переупорядочивание узлов 374 для получения более подробной информации.

Невозможно переупорядочить действия, используя контейнер Автозапуск, расположенный прямо под узлом **Сервер** системного дерева, потому что этот контейнер смешивает действия автозапуска, принадлежащие различным пользователям 478 системы.

Администрирование автозапуска

Для администрирования автоматически запускаемых действий используются два контекста 41: общий контекст Автозапуск 1468, который выступает в роли контейнера, и контекст Действие автозапуска 1470, который содержит информацию о каждом автоматически запускаемом действии.

Автозапуск виджетов, отчетов и инструментальных панелей

Виджеты 943, отчеты 928, инструментальные панели 912 и некоторые другие системные объекты могут быть *абсолютными* и *относительными*. Абсолютный объект запускается сам по себе, относительные объекты запускаются в **определенном контексте** 41, который служит источником данных.

Таким образом, добавление действия запуска относительного объекта напрямую в автозапуск не позволит системе распознать, для какого контекста оно должно быть запущено. Вместо этого, действие запуска объекта, **расположенное в целевом контексте**, должно быть добавлено в автозапуск.

Пример: Предположим, что мы имеем относительный виджет **График трафика**, который является пригодным для всех сетевых устройств 497. Вы можете добавить график трафика **определенного устройства** в автозапуск. Для этого нужно сделать следующее:

- Перетащите выбранное устройство в узел "Автозапуск".
- Выберите **График трафика** в диалоговом окне предлагаемых действий.

13.3.1 Конфигурация автозапуска

Каждое действие автозапуска обладает несколькими свойствами:

Описание поля	Имя поля
Описание. Текстовое описание автоматически запускаемого действия. Также является описанием 43 контекста "Действие автозапуска".	description
Маска 44 **контекстов**. Определяет область применения действия "Автозапуск". Действие запускается из всех контекстов, соответствующих маске.	mask
Действие. Действие, запускаемое при активации автозапуска.	action

Активный. Действие "Автозапуск" может быть включено или отключено (без удаления самого действия).	enabled
Условие. Выражение 112, которое должно вычисляться в true для выполнения действия автозапуска.	condition
Параметры. Параметры выполнения 102 действия.	executionParameters

Данные свойства доступны через переменную childInfo 147.

13.3.2 Автозапуск в распределенной архитектуре

Этот раздел описывает особенности поведения действий автозапуска в распределенной инсталляции 1332 AtomMind.

Чтобы действие автозапуска, подключенное от сервера-поставщика, запускалось клиентами, подключенными к серверу-потребителю, убедитесь, что путь контекста подсоединенного контекста действия автозапуска на сервере-потребителе соответствует маске контекста `users.*.autorun.*`, т.е. действие удаленного автозапуска появляется среди локальных действий.

13.4 Виджеты

Виджет - это "субприложение" с *графическим пользовательским интерфейсом* (GUI), определенным рядом компонентов виджета 955 и их макетом 950. Виджеты применяются для создания пользовательских форм и мнемосхем (HMI). Они могут использоваться, например, для управления, настройки и мониторинга различных аппаратных средств или системных компонентов.

Виджеты можно группировать в инструментальную панель 912, которая будет работать как пользовательский интерфейс.

Виджеты создавались для работы в десктопном приложении, AtomMind Client 359. Есть несколько способов для работы с виджетами в вебе (например, веб проигрыватель виджетов 1314), но в последних версиях AtomMind веб-инструментальные панели 912 полностью заменяют и даже превосходят виджеты по своему функционалу.

Пожалуйста, начинайте создавать ваш пользовательский интерфейс с помощью виджетов только если вы целенаправленно создаете десктопный интерфейс, например, для работы на сенорных панелях, в интерактивных терминалах и подобных устройствах.

Компоненты виджетов связаны с данными сервера и устройств при помощи привязок данных 1295. Привязки могут также связывать компоненты и определять различные шаблоны преобразования данных на основе выражений. Привязки могут быть активированы:

- Во время запуска виджета
- При событии на стороне сервера или изменении состояния, например, при поступлении новых данных с устройства
- При событии компонента виджета или изменении состояния, например, клик по кнопке
- Периодически

В виджет можно вставлять скрипты 1308, написанные на Java, если задачу по обработке данных нельзя выполнить, используя выражения.

Виджеты могут включать большое количество разных компонентов 955:

- Текстовые поля и поля для ввода паролей
- Текстовые и HTML области
- Триггерные и обычные кнопки
- Поля со списком
- Списки
- Растровые изображения
- Динамические векторные рисунки
- Графики
- Динамические карты (дорожные/рельефные/спутниковые)

- Независимые и зависимые кнопки
- Регуляторы и счётчики
- Компоненты выбора даты и времени
- Индикаторы выполнения
- Таблицы
- Журналы событий
- Системные деревья
- Индикаторы, измерители и диодные дисплеи
- Видео плееры
- Графы топологии

Эти компоненты можно сгруппировать в различные контейнеры 1041:

- Панели
- Панели с закладками
- Панели с разделителями
- Многослойные панели
- Вложенные виджеты
- Всплывающие меню

Контейнеры поддерживают два вида компоновки 950:

- Сетка, которая автоматически выравнивает компоненты согласно их размеру. Это особенно удобно при создании форм.
- Абсолютное позиционирование, строго определяющее позиционирование и размеры компонентов. Данная компоновка полезна при создании пользовательских экранов, карт, планов и пр. в стилистике пользовательского интерфейса.

Для создания сложных интерфейсов возможно объединение внутри одного виджета контейнеров с разными компоновками.

Компоненты виджетов интерфейса имеют тысячи редактируемых свойств 1274. Некоторые из них общие для нескольких типов компонентов, другие - специфичны для компонентов. Свойства компонентов включают видимость, размеры, границы, шрифты, цветы, штриховку, курсоры, всплывающие подсказки, свойства фокуса и другие.

Перед тем, как начать: Виджеты позволяют вам управлять и просматривать данные AtomMind в различных формах. Чтобы их создание стало эффективным, вам необходимо ознакомиться с ссылками 117, выражениями 112 и, наконец, привязкой данных 735. Перед продолжением, пожалуйста, прочитайте эти три главы.

Как только вы ознакомились с данной информацией, вы можете приступить к изучению способов применения привязки к виджетам в статье Привязки виджета 1295.

Виджеты создаются в Редакторе виджетов 423, который является частью AtomMind Client. Ядро виджета представляет собой шаблон 946 - текст в формате XML, содержащий формат и свойства внутренних компонентов вместе со связями между этими компонентами и *моделью данных*, т.е. данных, полученных из контекстов 41 AtomMind Server. Данные отношения существуют в форме привязок виджета 1295 и скриптов виджета 1308.

Можно осуществлять *запуск* каждого виджета в различных пользовательских интерфейсах AtomMind Server, например, в AtomMind Client 359. Его оформление зависит от пользовательского интерфейса, в котором он был запущен, но его общая структура и поведение будут такими же.

Виджет в AtomMind Client выглядит таким образом:

Сопутствующая документация:

- Создание виджета для управления устройством 1643
- Управление параметрами устройства при помощи графика 1666
- Построение сложных графиков 1671
- Хранение истории изменений объекта 1695
- Управление SVG изображениями 1650

Администрирование виджетов

Для администрирования виджетов используются два контекста: контекст Виджеты 1624, который служит в качестве контейнера, и контекст Виджет 1627, который содержит информацию об одном виджете.

глобальные настройки виджетов

Доступны следующие глобальные настройки плагина 207 Виджеты:

- **Режим выполнения виджетов**. Определяет, как виджеты выполняются в веб интерфейсе 220: на сервере (подходит для браузеров без поддержки Java, но создаёт дополнительную нагрузку на сервер) или в браузере (подходит для браузеров с поддержкой Java).
- **Запретить выполнение скриптов на стороне сервера**. Если настройка включена, работающие на сервере виджеты не могут использовать скрипты. Это важно в случае, если ваши дизайнеры виджетов недостаточно надежны, так как скрипты виджетов на стороне сервера имеют полный доступ к данным сервера и полному контролю над поведением сервера.

Встроенные виджеты

Несколько виджетов встроены в AtomMind Server и появляются в учетной записи пользователя администратор по умолчанию 479:

- **Исполнитель запросов**. Этот виджет позволяет вам протестировать и отладить запросы 829.
- **Суточные тревоги**. Содержит столбчатую диаграмму, отображающую количество ежедневных тревог и таблицу со списком экземпляров тревог, как только вы кликаете на определенном столбике диаграммы.

13.4.1 Создание виджетов

Создавать виджеты возможно только в AtomMind Client 359. Другие пользовательские интерфейсы AtomMind Server не имеют аналогов с AtomMind GUI Builder 423.

Новые виджеты создаются при помощи действия Создать новый виджет 1624 в контексте Виджеты. Это действие перетаскивания 101, которое допустимо для всех контекстов. Данное действие предполагает, что один контекст ("контекст по умолчанию") выбирается до запуска действия. Этот контекст по умолчанию имеет одно простое назначение во время создания виджета: его тип 43 будет использован в качестве типа контекста, для которого создается новый виджет. Действие Запустить виджет 948 будет установлено во все контексты данного типа, как только виджет будет создан.

Пример: Если контекст Пользователь 1608 перетаскивается в контекст Виджеты 1624 (т.е. выбирается в качестве "контекста по умолчанию" для дейтсвия "Создать новый виджет"), новый виджет может использоваться в каждом контексте пользователя в AtomMind Server. Каждый контекст пользователя будет иметь действие "Запустить виджет" с описанием, совпадающим с описанием виджета. См. действие Запустить виджет" 948 для получения более подробной информации.

Как только тип контекста выбран, запускается AtomMind GUI Builder 423 для построения виджета, который редактирует его шаблон 946 в визуальном редакторе. Теперь вы создаете формат 950 нового виджета, добавляя компоненты 955 и настраивая их свойства. На данном этапе необходимо определить привязки 1295, т.е. отношения между компонентами виджета и различными переменными 61 или функциями 70 контекстов AtomMind Server.

Вы можете прекратить изменение шаблона виджета в AtomMind GUI Builder, нажав кнопку **Готово** или **Отмена** на панели инструментов AtomMind GUI Builder. При нажатии **Отмена** действие отменяется и виджет не создается. При нажатии **Готово** пользователю предлагается 90 назначить свойства 947 нового виджета. Большинство свойств уже определены, но вам всё равно необходимо определить **имя** и **описание** для нового виджета.

После этого AtomMind Server создает контекст Виджет 1627, представляющий новый виджет. Затем он находит все контексты того же типа, что и "принятый" контекст для этого действия, и устанавливает для них действие Запустить виджет 948.

Теперь создание виджета завершено. Вы можете запустить виджет из контекста, для которого он был установлен.

13.4.2 Шаблон виджета

Шаблон - ядро виджета. Он записан в формате XML и хранится в качестве свойства 947 контекста Виджет 1627. Он определяет структуру виджета и состоит из двух частей:

- **Макет** 950, определяющий внутренние компоненты 955 виджета, их свойства и соответствующие позиции
- **Привязки** 1295 **данных**, которые определяют связь между компонентами виджета и данными, полученными от AtomMind Server

В большинстве случаев шаблон виджета редактируется в AtomMind GUI Builder 423 и его изменение инициируется действием Редактировать шаблон 1627 контекста "Виджет". Но в некоторых редких случаях шаблон можно изменить вручную (или заменить), редактируя его как текст. AtomMind GUI Builder 423 также позволяет экспортировать и импортировать шаблоны из внешних файлов XML.

Резервные копии шаблона

Можно сделать так, чтобы сервер хранил историю изменений шаблона виджета. Для получения более подробной информации обратитесь к руководству Хранение истории изменений объекта 1695.

13.4.3 Типы виджетов

Существует два типа виджетов:

- **Абсолютные** виджеты
- **Относительные** виджеты

Абсолютные виджеты

Абсолютные виджеты обычно обрабатывают и интегрируют данные, полученные из разных исходных контекстов.

Например, сложный виджет мнемосхемы в приложении контролирования процесса отображает данные от разных датчиков и имеет кнопки или другие активные компоненты для отправки команд исполнительным устройствам. Такой виджет должен быть абсолютным.

Привязки 1295 абсолютного виджета не могут использовать относительные ссылки 117.

Относительные виджеты

Относительные виджеты предназначены для обработки данных из определенного контекста 41 и его дочерних. Они служат для создания формы или пользовательского интерфейса, представляя данные Device или системного ресурса. Относительные виджеты обычно активируются при помощи действия Запустить виджет 948 контекста, данные которого будут использоваться в виджете. Например, в системе сетевого управления может быть относительный виджет, который называется **Таблица загрузки ЦП**. Нажмите правой кнопкой мыши на Device **Почтовый сервер** в AtomMind Client и выберите **Таблица загрузки ЦП** в контекстном меню.

13.4.4 Контекст по умолчанию виджета

Термин **контекст по умолчанию** применяется к виджету в двух различных случаях:

- Во время создания 945 или изменения виджета
- Во время работы 949 виджета

Во время создания виджета: при редактировании шаблона виджета в AtomMind GUI Builder создаются новые привязки 1295 его компонентов 955 и данных из контекстов. Когда создается новый или же редактируется виджет, пути контекста в новых привязках, которые ссылаются на контексты, расположенные ниже контекста по умолчанию, формируются согласно контексту по умолчанию.

Во время работы виджета: Контекст по умолчанию определяется также во время работы виджета. Это контекст, из которого выполняется действие Запустить виджет 948. Данный контекст по умолчанию используется для разрешения относительных ссылок во время работы виджета. `{.deviceservers:children#records}` - разрешение ссылки в количество устройств пользователя, если виджет создается для контекста пользователя. Он использует контекстный путь, относящийся к контексту по умолчанию.

Виджет обычно получает данные из контекста по умолчанию, но может обмениваться данными с другими контекстами AtomMind Server.

Для получения более подробной информации см. привязки виджетов 1295.

Примечание: Контекстом по умолчанию для каждого абсолютного 946 виджета является корневой контекст дерева контекстов AtomMind Server.

13.4.5 Конфигурация виджета

Свойства виджета могут быть просмотрены и изменены, используя действие Настроить 1627 в контексте виджета.

Описание поля	Имя поля
Имя виджета. Имя контекста виджета. Должно удовлетворять соглашениям о наименовании 42 контекста. Имя необходимо для ссылки на данный виджет из других частей системы.	name
Описание виджета. Текстовое описание 43 контекста виджета.	description
Шаблон виджета. Текст шаблона 946 виджета в формате XML.	template
Тип виджета. Тип 946 виджета, **абсолютный** или **относительный**.	type
Выражение пригодности. Определяет, какой контекст (контексты) виджет может "понять". Для получения более подробной информации обратитесь к разделу Пригодность ресурса 946.	validityExpression
Правила обновления пригодности. Список масок контекста и имен событий. Если событие, заданное в поле **Event** данной таблицы, происходит в любом контексте, соответствующем маске, заданной в поле **Mask** той же записи, **выражение пригодности** будет заново рассчитано для данного контекста. Это позволяет сделать виджет подходящим/неподходящим для определенного контекста в случае возникновения в нем изменений.	validityListeners
Определять пригодность для удаленных контекстов. При включенном флажке разрешено прикрепление виджета не только к локальным контекстам, но и к удаленным контекстам, подключеным по распределенной архитектуре 1334.	allowFalidityForRemoteContexts
Контекст по умолчанию. Контекст по умолчанию, используемый при редактировании шаблона 946 относительного 946 виджета в AtomMind GUI Builder или его запуске без использования действия "Запустить виджет" в Device или системном ресурсе. Его значение автоматически устанавливается в пути к контексту, для которого был создан 945 виджет. Обычно нет необходимости его редактировать, за исключением удаления контекста. Для получения более подробной информации см. Контекст по умолчанию для виджета 946.	defaultContext
Положение по умолчанию. Положение окна виджета используется, когда виджет запускается вручную, т.е. без использования Автозапуска 941, Избранного 2186 и других средств.  Когда действие Запустить виджет 1627 добавляется в Автозапуск "Избранного", его параметры работы 102 позволяют переопределить **Положение по умолчанию** и открыть виджет на пользовательской инструментальной панели.	defaultLocation

Инструментальная панель. Данная настройка позволяет сконфигурировать запуск виджета как инструментальной панели.	defaultDashboard

Данные свойства доступны через переменную childInfo 1628.

13.4.6 Запуск виджетов

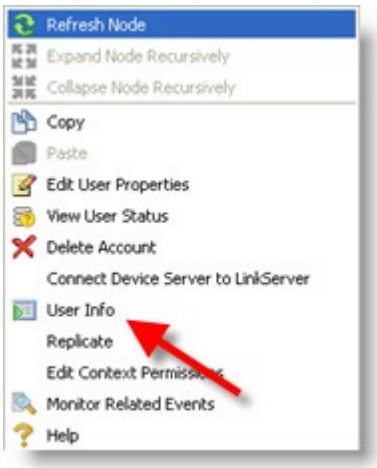
Существует несколько способов запуска виджета:

- Из графического пользовательского интерфейса (GUI) 423. Данный метод обычно применяется при тестировании нового виджета.
- Использование действия Запустить виджет 1627 контекста "Виджет". Для относительных 946 виджетов данный метод запускает виджет, который будет обрабатывать данные из контекста, указанного в свойстве "Контекст по умолчанию" 947.
- Использование действия 87 **Запустить виджет** (см. далее) с целью запуска относительного 946 виджета для определенного Device или системного ресурса.

Действие "Запустить виджет"

Данное действие находится в каждом контексте, для которого выражение пригодности 946 является **true**. Когда действие запускается в некотором контексте, он становится контекстом по умолчанию 946 для работающего виджета. Например, если виджет **Форма информации пользователя** создан для контекста Пользователь 1608 (например, имеет выражение пригодности `{.#type} == 'user'`), в контексте каждого пользователя AtomMind Server появится действие **Форма информации пользователя**. Его описание совпадает с описанием самого виджета. Действия, используемые для запуска виджетов, легко узнать по иконке .

Действие "Запустить виджет" в контекстном меню в AtomMind Client выглядит следующим образом:



Относительные виджеты устанавливают их действие Запуск Виджета только в те контексты, которые доступны для владельца виджета в соответствии с его правами доступа 477.

Чтобы запустить это действие, пользователь должен иметь необходимый уровень прав доступа 489 **Наблюдатель** в двух контекстах:

- Контекст виджета 1627 запускаемого виджета
- Контекст, с которым работает виджет, т.е. тот, где определено действие "Запустить виджет"

Окно виджета

В дополнение к стандартному набору кнопок на панели инструментов 366 окно виджета имеет несколько дополнительных кнопок:

Редактировать. Останавливает работающий виджет для его редактирования в графическом пользовательском интерфейсе (GUI) 423.

Журнал событий. Открывает журнал событий[1311] виджета.

Автономный виджет-проигрыватель

Часто необходимо, чтобы виджет работал как отдельное приложение, без запуска AtomMind Client или другого пользовательского интерфейса AtomMind Server. Например, виджет может отображаться на сенсорной панели встроенного ПК, контролирующего какое-либо оборудование.

Чтобы запустить виджет в автономном режиме, используйте приложение Виджет-проигрыватель, которое является частью AtomMind Client.

13.4.6.1 Входные параметры виджета

Когда виджет запускается, на входе он получает специальную Таблицу данных[49]. Эта таблица данных называется Таблица Входных Параметров Виджета.

Передача параметров виджету

Если виджет запускается напрямую из AtomMind Client (вручную выполняя Действие Запуска[948]), нет способа передать ему какие-либо параметры.

Однако виджеты можно запускать изнутри других виджетов. Когда виджет хочет запустить другой виджет, он запускает привязку[1295], чья цель[1295] - начать действие[737] Запустить виджет. Это выражение[1296] привязки должно выдавать результат в виде Таблицы Данных, и эта таблица передается новому открытому виджету в виде Таблицы Входных Параметров Виджета.

Итак, вот необходимая последовательность для передачи параметров между виджетами:

- В инициирующем виджете создайте привязку, чья цель указывает на действие запуска другого виджета, например `users.admin.widgets.voltageChart:launch!`
- Убедитесь, что выражение вышеуказанной привязки создает/извлекает и возвращает объект Таблица данных.

Обработка входных параметров в виджете

Получить доступ к входным параметрам виджета легко. Таблица Входных Параметров Виджета - это всегда таблица по умолчанию[120] в период оценки выражения привязки виджета. Таким образом, если эта таблица имеет поле с именем `mode`, на этот параметр можно сослаться из любого другого выражения ппривязки, используя ссылку `{mode}`.

13.4.6.2 Работа виджета

Когда виджет запускается в пользовательском интерфейсе AtomMind Server, для него создается новое окно, в котором виджет воспроизводит содержание корневой панели (root panel). Размер окна определяется параметрами панели **Ширина** и **Высота** (в пикселях) или рассчитывается автоматически на основе оптимальных размеров внутренних компонентов виджета, если **Ширина** или **Высота** корневой панели не заданы (т.е. установлены на 0). *Оптимальным размером* является тот размер компонента, когда он может отображать всю информацию целиком и без прокрутки экрана. Оптимальный размер должен быть таким, чтобы можно было отобразить текст полностью. Размер ползунка для прокрутки зависит от необходимой длины для отображения компонента полностью, и т.д. Если определены только ширина или высота, то оптимально устанавливается другая величина.

Если размер окна виджета изменяется, его внутренние компоненты переносятся и приобретают новый размер согласно их индивидуальным ограничениям[950] и размерам. Формат виджета может также изменяться во время работы, когда его компоненты модифицируются привязками[1295] из-за действий пользователя или изменений данных контекста.

Обработка привязок

При запуске виджета его процессор обрабатывает все привязки, которые должны обрабатываться при запуске виджета. Данный процесс называется Сеанс обработки привязок. Обычно привязки, выполненные при запуске, используются для чтения определенных значений контекста и инициализации компонентов виджета.

Во время работы виджет контролирует изменения в контекстах и свойствах компонентов и снова обрабатывает привязки при их обнаружении. Также возможно настроить привязки для периодической проверки этих значений.

Каждая обрабатываемая привязка может изменять данные сервера, свойства компонентов виджета или выполнять его скрипт[1308].

Для получения более подробной информации см. статью привязки[1295].

13.4.7 Модель данных виджетов

Каждый виджет представлен контекстом 41 на сервере, который содержит настройки расположения 950 виджетов, компоненты 955, привязки 1295 и многое другое.

В то же время, после запуска 948 виджета создается отдельное дерево контекстов 41, которое представляет модель данных текущего виджета. Каждый контекст дерева контекстов виджета - это один UI-компонент 955 запущенного виджета. Все контексты компонентов добавляются в корень контекстного дерева виджета, поэтому иерархия контекста компонентов не соответствует иерархии вложенных в виджет контейнеров 1041 и компонентов.

Переменные 61, функции 70 и события 73 контекста компонента виджета соответствуют свойствам, операциям и событиям этого же UI-компонента. Таким образом, обращение к сущностям контекста компонента из привязок 226 виджета - это процесс, по сути работающий с UI-компонентами, например, настройка UI-компонентов, реагирование на их события и пр.

13.4.8 Компоновка виджета

Каждый виджет состоит из нескольких графических компонентов 955, которые можно разделить на две большие группы:

- Обычные компоненты
- Контейнеры

Основное различие между контейнерами и обычными компонентами заключается в том, что контейнеры могут включать в себя другие компоненты и контейнеры.

Каждый контейнер имеет одну или более панелей или вкладок, в которых могут располагаться другие компоненты. Например, контейнер Панель 1045 имеет одно всегда видимое окно без дополнительного оформления. Контейнер Панель вкладок 1045 может иметь любое количество подокон (вкладок) или контейнеров. У каждого подокна свой макет и количество дочерних компонентов или контейнеров.

Основным компонентом виджета является (root panel). Данный компонент - это контейнер для всех других компонентов виджета. Во время работы виджета она занимает все доступное место окна виджета.

Типы компоновки панелей

Каждая панель в контейнере имеет собственную компоновку расположенных на нем компонентов. Существует два типа компоновки:

- Сетка 950
- Абсолютное позиционирование 954

Тип компоновки можно менять при помощи свойства Компоновка 1041.

Если вы хотите создать виджет с изменяемыми размерами, используйте Сетку.

Ограничения компонентов

Ограничения компонента виджета определяют его положение и размер в контейнере-родителе. Они также определяют, как компонент меняет свой размер и положение при расширении/сужении окна виджета.

Для каждого компонента виджета предусмотрены собственные ограничители, за исключением корневого подокна.

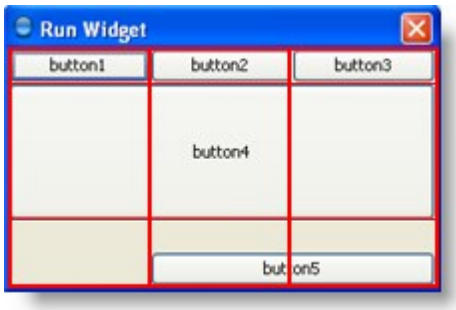
13.4.8.1 Компоновка "Сетка"

Данная компоновка представляет собой сетку из строк и столбцов, позволяя определенным компонентам занимать несколько строк и столбцов. Высота строк и ширина столбцов может быть разной. Сетку можно скрыть/показать во время редактирования шаблона виджета в AtomMind GUI Builder 423, однако, ее не видно во время работы виджета. Видимы только компоненты.

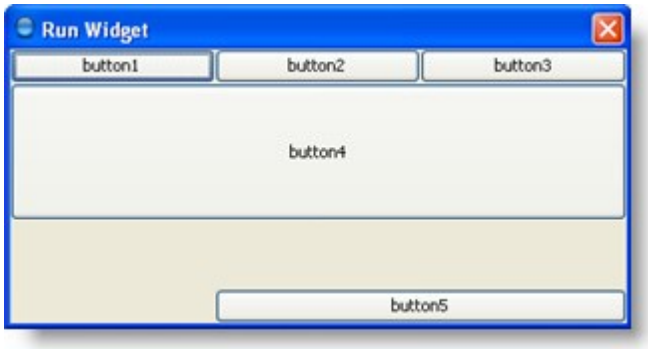
Не путайте компоновку "Сетка" с абсолютной привязкой к сетке 438 компоновки.

Далее приведен пример виджета с пятью компонентами-кнопками 968 в его **корневой панели**:

Компоновка "Сетка" для данного виджета состоит из трех строк и трех столбцов. Кнопка во второй строке занимает все столбцы, а кнопка в третьей строке занимает два правых столбца:



При увеличении окна виджета, как показано на следующем рисунке, вы увидите, что кнопка строки, которая содержит Кнопку 5, увеличивается вертикально. Горизонтальное расстояние распределяется поровну между всеми столбцами. Изменение размера основано на **весе**, заданном для индивидуальных компонентов в макете панели. Также вы заметите, что каждый компонент занимает все доступное место по горизонтали, но не все (как, например, кнопка 5) доступное место по вертикали. Данное поведение также определяется весом компонента.



Ограничения компоновки "Сетка"

Выделяют следующие ограничения, характеризующие положение компонентов при компоновке "Сетка":

КОЛОНКА, РЯД

Определяют колонку и ряд верхней левой части компонента. Крайняя левая колонка - **column 0**, а верхний ряд - **row 0**.

Имена переменных: gridx, gridy

КОЛИЧЕСТВО ЗАНИМАЕМЫХ КОЛОНОК, КОЛИЧЕСТВО ЗАНИМАЕМЫХ РЯДОВ

Определяют количество столбцов и строк, занимаемых компонентом в компоновке "Сетка" контейнера-родителя. Данные ограничители устанавливают число ячеек, которые занимает компонент. Значение по умолчанию равно 1.

Имена переменных: **gridWidth, gridHeight**

ТОЧКА ПРИВЯЗКИ

Используется для определения места компонента в случае, когда он меньшего размера, чем область, в которой он размещен. Корректные значения:

- Центр
- Север

- Восток
- Юг
- Запад
- Северо-Запад
- Северо-Восток
- Юго-Восток
- Юго-Запад

Итак, если у вас много свободного пространства и маленький компонент, точка привязки позволит вам определить необходимое место положение компонента.

Имя переменной: **anchor**

ЗАЛИВКА

Используется, когда свободное пространство больше, чем требуемый размер компонента, для определения, каким образом следует изменить размер компонента. Корректные значения включают **Отсутствует** (по умолчанию), **По горизонтали** (расширяет компонент по горизонтали соразмерно с областью изображения, но не изменяет высоту), **По вертикали** (увеличивает высоту компонента по вертикали соразмерно с областью изображения) и **По горизонтали и вертикали** (размер компонента подстраивается для полного заполнения области изображения).

Имя переменной: **fill**

ВЕРХНЯЯ ГРАНИЦА, НИЖНЯЯ ГРАНИЦА, ЛЕВАЯ ГРАНИЦА, ПРАВАЯ ГРАНИЦА

Определяет внешнее выравнивание компонента - минимальное количество места между компонентом и краями области отображения. *Область отображения* представляет собой пространство, предоставляемое текущей компоновкой для компонента. По умолчанию каждый компонент не имеет внешних отступов (т.е. поля установлены на ноль).

Имена свойств: **insetsTop, insetsBottom, insetsLeft, insetsRight**

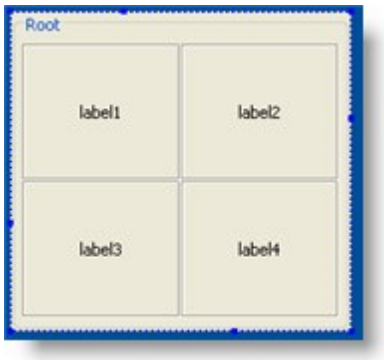
ВЕС X, ВЕС Y

От влияния по горизонтали и вертикали зависит, в каком отношении распределится ширина столбцов и высота строк в свободной области. Для каждого столбца или строки вычисляется его отношение к компоненту с максимальным весом в этом же столбце или строке.

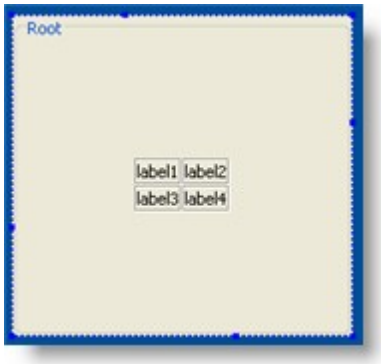
До тех пор, пока вы не зададите по крайней мере одно ненулевое значение для Weight X или Weight Y, все компоненты будут группироваться в центре контейнера. Это происходит потому, что вес равен 0.0 (по умолчанию), и любое свободное место отводится для пространства между ячейками и краями контейнера.

Обычно весовые значения установлены на 0.0 и 1.0. Большие числа указывают на то, что строка или столбец компонента нуждаются в большем пространстве. Свободное место образуется преимущественно около крайнего правого столбца и нижней строки.

Компоненты, весовые значения которых равны 1.0, выглядят следующим образом:



Установка весовых значений того же компонента на 0.0 приведет к данному результату:



Имена свойств: **weightx, weighty**

ПРИМЕР ОГРАНИЧЕНИЙ

Вернемся к нашему примеру виджета с пятью кнопками:

Здесь приведены ограничения для данных кнопок (отображены значения не по умолчанию):

Кнопка 1:

Ряд: 0

Колонка: 0

Вес X: 1.0

Заливка: горизонтальная

Кнопка 2:

Ряд: 0

Колонка: 1

Вес X: 1.0

Заливка: горизонтальная

Кнопка 3:

Ряд: 0

Колонка: 2

Вес X: 1.0

Заливка: горизонтальная

Кнопка 4:

Ряд: 1

Колонка: 0

Количество колонок: 3

Вес X: 1.0

Заливка: горизонтальная

Кнопка 5:

Ряд: 2

Колонка: 1

Количество колонок: 2

Точка привязки: центр

Вехнее поле: 20

Вес X: 1.0

Вес Y: 1.0

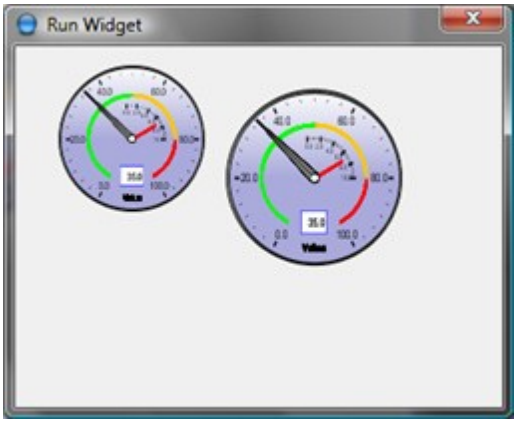
Заливка: горизонтальная

Чтобы определить, как построить компоновку, см. раздел Редактирование компоновки виджета 432 в описании AtomMind GUI Builder.

13.4.8.2 Компоновка "Абсолютное позиционирование"

Данный макет позволяет задать абсолютное позиционирование для каждого компонента. Это удобно для размещения различных видов измерителей, счетчиков и подобных компонентов поверх изображения или анимированных графиков. Компоненты в абсолютном макете могут перекрывать друг друга.

Далее приведен пример того, как может выглядеть абсолютный макет (два компонента расположены на определенных позициях в корневой панели 1042):



Ограничения абсолютного позиционирования

Ограничителями, характеризующими позицию компонента при абсолютном позиционировании, являются:

КООРДИНАТА X, КООРДИНАТА Y

Абсолютные горизонтальные и вертикальные координаты компонента. Увеличение горизонтальных (X) координат происходит слева направо, увеличение вертикальных (Y) координат осуществляется сверху вниз, таким образом, крайний левый угол рамки имеет координаты X=0, Y=0.

ВЫСОТА (Z-ORDER)

Высота определяет, как "высоко" расположен компонент в контейнере-родителе и, следовательно, какие компоненты он перекроет. Увеличение значения высоты расположит компонент выше других и он перекроет их.

МАСШТАБИРОВАНИЕ

Масштабирование позволяет определить, как контейнер и его компоненты будут масштабироваться.

Допустимые значения:

- Нет - стандартное поведение, без масштабирования.
- Вертикально - только вертикальное масштабирование.
- Горизонтально - только горизонтальное масштабирование.
- В обе стороны - масштабирование и вертикально, и горизонтально.

13.4.9 Компоненты

Компоненты виджета подразделяются на несколько категорий:

- Простые компоненты 955
- Контейнеры 1041
- Графики 1051
- Индикаторы 1238
- Графические примитивы 1268

Простые компоненты отображают данные или взаимодействуют с пользователем. Контейнеры содержат информацию о простых копонентах или других контейнерах. Однако некоторые контейнеры могут также взаимодействовать с пользователем. Например, контейнер Панель с вкладками 1045 управляется кликами мыши по вкладкам (для отображения выбранной вкладки).

Графики и индикаторы не имеют существенных отличий от обычных компонентовю Они были помещены в отдельную группу только из-за их многочисленности.

Более подробную информацию о компонентах виджета см. в разделе Компоновка виджета 950.

Компоненты имеют немного отличающийся друг от друга вид при различных пользовательских интерфейсах AtomMind Server и в различных операционных системых. Вид компонента и его поведение в определенном пользовательском интерфейсе AtomMind Server определяется **отрисовщиком компонента.**

Обычные компоненты

Обычные компоненты не имеют компонентов-потомков (в противоположность контейнерам 1041). Далее приведен список простых компонентов в редакторе:

Метка 956

Текстовое поле 958


Поле ввода пароля 959

Поле с форматированием 960


Список 964

Текстовая область 965

HTML область 967


Кнопка 968


Триггерная кнопка 971


Зависимая кнопка 973


Независимая кнопка 974


Поле со списком 975


Дата / время 976

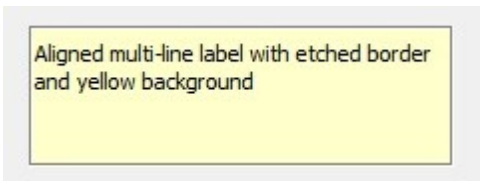

Редактор таблицы данных 979


Лог событий 983

13.4.9.1 Метка

Метка - это отображаемая область для короткой текстовой надписи, которая не реагирует на вводимые события и поэтому может не получать фокуса ввода с клавитатуры.

Метка выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Основной цвет 1275, Фон 1275, Непрозрачный 1275, Рамка 1275, Шрифт 1275, Курсор 1276, Всплывающая подсказка 1276, Фокусируемый 1276, Всплывающее меню 1276

Пользовательские свойства

ТЕКСТ (Свойство по умолчанию 1274)

Текст сообщения метки.

Имя свойства: **text**

Тип свойства: **Строка**

Помимо компонента Область HTML 967, метка может также отображать простой текст HTML, который должен иметь в начале тег `<html>`, например: `<html><font color="red">Red</font> and <b>bold</b> text.</html>`

ВЫРАВНИВАНИЕ

Горизонтальное выравнивание текста в области изображения метки.

Возможные значения:

Описание	Значение
Слева	2
По центру	0
Справа	4

Имя свойства: **horizontalAlignment**

Тип свойства: **Целое**

ВЫРАВНИВАНИЕ ПО ВЕРТИКАЛИ

Выравнивание текста по вертикали в области метки. **Сверху**, **По центру** и **Снизу**.

Возможные значения:

Описание	Значение
Сверху	1
По центру	0
Снизу	3

Имя свойства: **verticalAlignment**

Тип свойства: **Целое**

ПИКТОГРАММА

Изображение на метке. Если значение является NULL, изображение не показывается.

Имя свойства: **icon**

Тип свойства: **Блок данных**

ПРОМЕЖУТОК МЕЖДУ ПИКТОГРАММОЙ И ТЕКСТОМ

Определяет расстояние между текстом и пиктограммой.

Имя свойства: **iconTextGap**

Тип свойства: **Целое**

ПИКТОГРАММА ПРИ ДЕАКТИВАЦИИ

Изображение, появляющееся на метке, когда она отключена. Если значение NULL, пиктограмма сформируется автоматически из обычной пиктограммы.

Имя свойства: **disabledIcon**

Тип свойства: **Блок данных**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1288, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.9.2 Текстовое поле

Текстовое поле является компонентом, который позволяет редактировать одну строку текста.

Текстовое поле выглядит следующим образом:

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Основной цвет 1275, Фон 1275, Непрозрачный 1275, Рамка 1275, Шрифт 1275, Всплывающая подсказка 1276, Фокусируемый 1276, Всплывающее меню 1276

Пользовательские свойства

ТЕКСТ (Свойство по умолчанию 1274)

Текст по умолчанию в текстовом поле.

Имя свойства: **text**

Тип свойства: **String**

РЕДАКТИРУЕМЫЙ

Флажок указывает на то, что текст в поле может быть изменен. Основное различие между нередактируемыми и невключенными 1275 компонентами представляет тот факт, что текст в нередактируемом компоненте остается доступным и может быть выбран при помощи мыши.

ЭКРАННАЯ КЛАВИАТУРА

Определяет, должен ли этот компонент поддерживать экранную клавиатуру для ввода текста. Для получения более подробной информации см. поддержка сенсорного экрана 1316.

Возможные варианты:

- Отсутствует - не использует экранную клавиатуру
- Один щелчок - отображает клавиатуру при щелчке по любой области компонента
- Двойной щелчок - отображает клавиатуру, если по компоненту щелкнуть два раза

Имя свойства: **onscreenKeyboard**

Тип свойства: **Integer**

ВЫРАВНИВАНИЕ

Горизонтальное выравнивание текста в области изображения метки.

Возможные значения:

Описание	Значение
Слева	2
По центру	0
Справа	4

Имя свойства: **horizontalAlignment**

Тип свойства: **Integer**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

Пользовательские события

ОБНОВЛЕНИЕ ВСТАВКИ

Событие формируется каждый раз, когда передвигается вставка или вводится текст в компоненте.

Имя события: **caretUpdate**

Поля события:

Поле	Имя	Тип	Описание
идентификатор	id	целое	Идентификатор типа события.
точка	dot	целое	Положение вставки.
пометка	mark	целое	Положение другого конца логического отбора. Если отбора нет, оно будет таким же, как точка.

13.4.9.3 Поле ввода пароля

Поле ввода пароля является компонентом, позволяющим изменять одну строку маскируемого (скрытого) текста.

Поле ввода пароля выглядит следующим образом:

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Основной цвет 1275, Фон 1275, Непрозрачный 1275, Рамка 1275, Шрифт 1275, Всплывающая подсказка 1276, Фокусируемый 1276, Всплывающее меню 1276

Пользовательские свойства

ЭКРАННАЯ КЛАВИАТУРА

Определяет, должен ли компонент поддерживать экранную клавиатуру для ввода текста. Для получения более подробной информации см. поддержка сенсорного экрана 1316.

Возможные варианты:

- Отсутствует - не использует экранную клавиатуру
- Один щелчок - отображает клавиатуру при щелчке на любой области компонента
- Двойной щелчок - отображает клавиатуру, если на компоненте щелкнуть два раза

Имя свойства: **onscreenKeyboard**

Тип свойства: **Integer**

ВЫРАВНИВАНИЕ

Горизонтальное выравнивание текста в пределах отображаемой области.

Возможные значения:

Описание	Значение
Слева	2
По центру	0
Справа	4

Имя свойства: **horizontalAlignment**

Тип свойства: **Integer**

ПАРОЛЬ (Свойство по умолчанию 1274)

Пароль по умолчанию в поле ввода пароля. Должно оставаться пустым.

Имя свойства: **password**

Тип свойства: **String**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

Пользовательские события

ОБНОВЛЕНИЕ ВСТАВКИ

Это событие формируется каждый раз, когда переносится вставка или вводится любой текст в компоненте.

Имя события: **caretUpdate**

Поля события:

Поле	Имя	Тип	Описание
идентификатор	id	целое	Идентификатор типа события.
точка	dot	целое	Положение вставки.
пометка	mark	целое	Положение другого конца логического отбора. Если отбора нет, оно будет таким же, как точка.

13.4.9.4 Поле с форматированием

Поле с форматированием является компонентом, позволяющим изменить одну строку текста и поддерживающим произвольное форматирование.

Поле с форматированием выглядит следующим образом:

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Основной цвет 1275, Фон 1275, Непрозрачный 1275, Рамка 1275, Шрифт 1275, Всплывающая подсказка 1276, Фокусируемый 1276, Всплывающее меню 1276

Пользовательские свойства

ТЕКСТ (Свойство по умолчанию 1274)

Текст по умолчанию в текстовом поле.

Имя свойства: **text**

Тип свойства: **String**

РЕДАКТИРУЕМЫЙ

Флажок указывает на то, что текст в поле может быть изменен. Основное различие между нередактируемыми и невключенными 1275 компонентами представляет тот факт, что текст в нередактируемом компоненте остается доступным и может быть выбран при помощи мыши.

ЭКРАННАЯ КЛАВИАТУРА

Определяет, должен ли компонент поддерживать экранную клавиатуру для ввода текста. Для получения более подробной информации см. поддержку сенсорного экрана 1316.

Возможные варианты:

- Отсутствует - не использует экранную клавиатуру
- Один щелчок - отображает клавиатуру при щелчке на любой области компонента
- Двойной щелчок - отображает клавиатуру, если на компоненте щелкнуть два раза

Имя свойства: **onscreenKeyboard**

Тип свойства: **Integer**

ВЫРАВНИВАНИЕ

Горизонтальное выравнивание текста в компоненте.

Возможные значения:

Описание	Значение
Слева	2
По центру	0
Справа	4

Имя свойства: **horizontalAlignment**

Тип свойства: **Integer**

РЕЖИМ ВАЛИДАЦИИ

Выбирает регулярное выражение или валидацию поля маски.

Возможные значения:

Описание	Значение
Маска	0
Регулярное выражение	1

Имя свойства: **validationMode**

Тип свойства: **Integer**

Маска, Валидные символы, Невалидные символы, Символ-заполнитель и **Заполнитель** включаются только тогда, когда режим форматирования равен *Маске.*

Регулярное выражение включается только тогда, когда **Режим валидации** равен *Регулярному выражению*.

МАСКА

Шаблон валидации маски. Данный параметр определяет набор символов для ввода в каждой позиции поля.

Могут быть определены следующие символы:

Символ	Описание
#	Любое число.
'	Символ используется для выделения любого из специальных символов форматирования
U	Любой символ. Все символы нижнего регистра преобразовываются в символы верхнего регистра.
L	Любой символ. Все символы верхнего регистра преобразовываются в символы нижнего регистра.
A	Любой символ или число.
?	Любой символ.
*	Символ или число.
H	Любой шестнадцатиричный символ (0-9, a-f или A-F)

Свойство **Маска** работает вместе со свойствами **Валидные символы** и **Невалидные символы** (см. далее):

- Только символы, появляющиеся в свойстве **Валидные символы**, будут разрешены для любой позиции строки.
- Любой символ, появляющийся в свойстве **Невалидные символы**, не будет разрешен для любой позиции конечного значения.

Имя свойства: **mask**

Тип свойства: **String**

ВАЛИДНЫЕ СИМВОЛЫ

Список валидных символов в любой позиции строки. Таким образом, если значение свойства **Валидные символы** ненулевое, символ будет считаться валидным, если он есть внутри строки **валидных символов** и соответствует маске.

Имя свойства: **validCharacters**

Тип свойства: **String**

НЕВАЛИДНЫЕ СИМВОЛЫ

Список символов, которые будут невалидными в любой позиции строки, независимо от **Маски**.

Имя свойства: **invalidCharacters**

Тип свойства: **String**

ЗАПОЛНИТЕЛЬ

Используемая строка, если значение не полностью заполняет строку. Значение NULL подразумевает использование **символа-заполнителя**. Приоритет отдается строке заполнителя.

Введеная строка будет использоваться в исходном формате, после будет использоваться только символ-заполнитель.

Имя свойства: **placeholder**

Тип свойства: **String**

СИМВОЛ-ЗАПОЛНИТЕЛЬ

Символ, используемый вместо отсутствующих в значении, т.е. пользователь должен их ввести. Значением по умолчанию является пробел.

Это применимо только тогда, когда строка Заполнитель не была задана или не полностью заполняет маску.

Имя свойства: **placeholderCharacter**

Тип свойства: **String**

РЕГУЛЯРНОЕ ВЫРАЖЕНИЕ

Шаблон регулярного выражения 2142, используемый для валидации значения, если **Режим валидации** установлен на "Регулярное выражение".

Имя свойства: **regEx**

Тип свойства: **String**

ПЕРЕЗАПИСЫВАТЬ ТЕКСТ

Настраивает поведение при вводе символов. Если данный параметр является true, новые символы записываются поверх существующих символов в поле.

Имя свойства: **overwritesText**

Тип свойства: **Boolean**

ПОВЕДЕНИЕ ПРИ ПОТЕРЕ ФОКУСА

Следит за тем, что происходит, когда поле форматируемого текста теряет фокус.

Описание	Значение	Примечание
Сохранить	0	Фиксирует значение. Если отредактированное значение не является допустимым, тогда значение компонента не изменяется, а отредактированное значение не заменяет значение компонента.
Сохранить или Восстановить	1	Похоже на Сохранить, но если значение недопустимо, поведение такое же, как при Восстановить.
Восстановить	2	Возвращает дисплей к заданному по умолчанию, возможно, с потерей текущего редактирования.
Удержать	3	Не предпринимает никаких действий, т.е. не обновляет значение.

Имя свойства: **focusLostBehaviour**

Тип свойства: **Boolean**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

Пользовательские события

ОБНОВЛЕНИЕ ВСТАВКИ

Это событие формируется каждый раз, когда вставка перемещается или в компонент вводится любой текст.

Имя события: **caretUpdate**

Поля события:

Поле	Имя	Тип	Описание
идентификатор	id	целое	Идентификатор типа события.

точка	dot	целое	Положение вставки.
отметка	mark	целое	Положение другого конца логического отбора. Если отбора нет, оно будет таким же, как точка.

13.4.9.5 Список

Компонент, позволяющий пользователю выбирать одно и более значения из списка.

Список выглядит следующим образом:

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Основной цвет 1275, Фон 1275, Непрозрачный 1275, Рамка 1275, Шрифт 1275, Всплывающая подсказка 1276, Фокусируемый 1276, Всплывающее меню 1276

Пользовательские свойства

РЕЖИМ ВЫБОРА

Режим выбор элементов списка: **Выбор одного элемента, Выбор одного интервала элементов** или **Выбор нескольких интервалов элементов**.

Имя свойства: **selectionMode**

Тип свойства: **Integer**

КОМПОНОВКА

Определяет вид ячеек списка. Список из четырех ячеек может быть представлен следующими способами:

По вертикали:

0

1

2

3

По вертикали с переносом:

0 1

2 3

По горизонтали с переносом:

0 2

1 3

Имя свойства: **layoutOrientation**

Тип свойства: **Integer**

ЭЛЕМЕНТЫ СПИСКА

Таблица, определяющая все элементы списка.

Имя свойства: **listItems**

Тип свойства: **Data Table**

Можно заполнить Список значениями выборки 49 из определенного поля таблицы данных, используя привязку 1295. Например, привязка, которая внесет в список страны, доступные в поле Страна переменной Информация о пользователе 1609, определенной в контексте Пользователь пользователя admin, будет выглядеть следующим образом:

Цель: `form/list1:listItems`

Выражение: `{users.admin:childInfo$country#options}`

Данная привязка создается автоматически при перетаскивании поля переменной в список в Редакторе виджетов 423. Более подробная информация доступна здесь 442.

ВЫБРАННЫЕ ЭЛЕМЕНТЫ (Свойство по умолчанию 1274)

Список изначально выбираемых элементов.

Имя свойства: **selectedItems**

Тип свойства: **Data Table**

ВЫБРАННЫЙ ЭЛЕМЕНТ

Список элементов, выбранных по умолчанию (удобный способ установить один элемент в свойстве **Выбранные элементы**).

Имя свойства: **selectedItem**

Тип свойства: *dynamic*

КОЛИЧЕСТВО ВИДИМЫХ РЯДОВ

Устанавливает предпочитаемое количество рядов в списке, отображаемом без необходимости в прокрутке списка при наличии достаточного пространства в области контейнера-родителя.

Данное свойство можно установить на **Авто**. В данном случае список будет занимать всё доступное пространство в области контейнера-родителя.

Имя свойства: **visibleRowCount**

Тип свойства: **Integer**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

Пользовательские события

ВЫБОР В СПИСКЕ

Данное событие появляется, когда пользователь выбирает или отменяет выбор каких-либо элементов из списка.

Имя события: **listSelection**

13.4.9.6 Текстовая область

Текстовая область является компонентом, отображающим множество редактируемых строк текста.

Текстовая область выглядит следующим образом:

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Основной цвет 1275, Фон 1275, Непрозрачный 1275, Рамка 1275, Шрифт 1275, Курсор 1276, Всплывающая подсказка 1276, Фокусируемый 1276, Всплывающее меню 1276

Чтобы включить прокрутку, текстовую область следует поместить на панель с необходимыми настройками прокрутки, а ширина 1274 и высота 1274 текстовой области устанавливаются на ноль.

Пользовательские свойства

ТЕКСТ (Свойство по умолчанию 1274)

Текст по умолчанию в текстовой области.

Имя свойства: **text**

Тип свойства: **String**

РЕДАКТИРУЕМЫЙ

Флажок указывает на то, что текст в поле может быть изменен. Основное различие между нередактируемыми и невключенными 1275 компонентами представляет тот факт, что текст в нередактируемом компоненте остается доступным и может быть выбран при помощи мыши.

Имя свойства: **editable**

Тип свойства: **Boolean**

ПЕРЕНОС СТРОК

Устанавливает политику переноса текстовой области по строкам. Если включено, будет осуществляться перенос строк, если они превышают заданную длину. Если отключено, перенос строк не выполняется никогда.

Имя свойства: **lineWrap**

Тип свойства: **Boolean**

ПЕРЕНОС ПО ГРАНИЦАМ СЛОВ

Устанавливает стиль переноса, используемый при включенном **построчном переносе**. Если данное свойство включено, перенос осуществляется по границам слова (пробелам), когда строки превышают заданную длину. Если отключено, перенос будет выполняться по границам символов (т.е. перенос в середине слова).

Имя свойства: **wrapStyleWord**

Тип свойства: **Boolean**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1288, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

Пользовательские события

ОБНОВЛЕНИЕ ВСТАВКИ

Данное событие возникает каждый раз во время передвижения вставки или при вводе текста в компоненте.

Имя события: **caretUpdate**

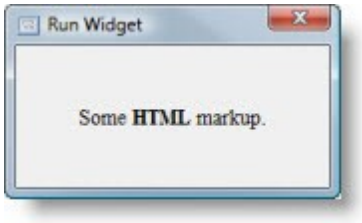
Поля события:

Поле	Имя	Тип	Описание
идентификатор	id	Integer	Идентификатор типа события.
точка	dot	Integer	Положение вставки.
отметка	mark	Integer	Положение другого конца логического отбора. Если отбора нет, оно будет таким же, как точка.

13.4.9.7 HTML область

Область HTML является компонентом, который отображает текст, отформатированный при помощи языка HTML. Включение тегов *<html></html>* не требуется.

HTML область выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Основной цвет 1275, Фон 1275, Непрозрачный 1275, Рамка 1275, Шрифт 1275, Курсор 1276, Всплывающая подсказка 1276, Фокусируемый 1276, Всплывающее меню 1276

Пользовательские свойства

ТЕКСТ (Свойство по умолчанию 1274)

Текст по умолчанию в области HTML.

Имя свойства: **text**

Тип свойства: **String**

РЕДАКТИРУЕМЫЙ

Имя свойства: **editable**

Тип свойства: **Boolean**

Флажок указывает на то, что текст в поле может быть изменен. Основное различие между нередактируемыми и невключенными 1275 компонентами представляет тот факт, что текст в нередактируемом компоненте остается доступным и может быть выбран при помощи мыши.

ПЕРЕХОД ПО ССЫЛКАМ

Включает переход по ссылкам внутри области HTML.

Имя свойства: **browsingLinks**

Тип свойства: **Boolean**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

пользовательские события

ЩЕЛЧОК ПО ССЫЛКЕ

Событие возникает каждый раз при щелчке по ссылке.

Имя события: **mouseLinkClicked**

Поля события:

Поле	Имя	Тип	Описание
ID	id	Integer	Идентификатор типа события.
When	when	Date	Временная метка.
Reference	hRef	String	Назначение ссылки.
Description	hDescription	String	Описание ссылки.

НАВЕДЕНИЕ КУРСОРА НА ССЫЛКУ

Событие возникает каждый раз при наведении курсора на ссылку.

Имя события: **mouseLinkEntered**

Поля события:

Поле	Имя	Тип	Описание
ID	id	Integer	Идентификатор типа события.
When	when	Date	Временная метка.
Reference	hRef	String	Назначение ссылки.
Description	hDescription	String	Описание ссылки.

ВЫВОД КУРСОРА МЫШИ СО ССЫЛКИ

Событие возникает каждый раз при отведении курсора от ссылки.

Имя события: **mouseLinkExited**

Поля события:

Поле	Имя	Тип	Описание
ID	id	Integer	Идентификатор типа события.
When	when	Date	Временная метка.
Reference	hRef	String	Назначение ссылки.
Description	hDescription	String	Описание ссылки.

13.4.9.8 Кнопка

Обычная кнопка 

Кнопка выглядит следующим образом:

Пример: см. пример Открыть новый виджет при нажатии кнопки 1297, чтобы понять, как выполнить действие 87 при нажатии кнопки, например, открыть другой виджет.

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Основной цвет 1275, Фон 1275, Непрозрачный 1275, Рамка 1275, Шрифт 1275, Всплывающая подсказка 1276, Фокусируемый 1276, Всплывающее меню 1276

Пользовательские свойства

ТЕКСТ (Свойство по умолчанию 1274)

Текст метки кнопки.

Кнопки поддерживают многострочные тексты HTML с форматированием. Чтобы запустить HTML, начните значение свойства **Text** тегом <html>: `<html>Line <b>1</b><br>Line <b>2</b>`

Имя свойства: **text**

Тип свойства: **String**

ВЫРАВНИВАНИЕ

Горизонтальное выравнивание текста в отображаемой области кнопки.

Возможные значения:

Описание	Значение
Слева	2
По центру	0
Справа	4

Имя свойства: **horizontalAlignment**

Тип свойства: **Integer**

ВЕРТИКАЛЬНОЕ ВЫРАВНИВАНИЕ

Вертикальное выравнивание текста в отображаемой области кнопки.

Возможные значения:

Описание	Значение
Сверху	1
По центру	0
Снизу	3

Имя свойства: **verticalAlignment**

Тип свойства: **Integer**

ПОЛЯ

Расстояние между границей и меткой с каждого края. Значение полей имеет следующие поля:

Свойство	Имя	Тип
Сверху	top	целое
По левому краю	left	целое
Снизу	bottom	целое

По правому краю	right	целое

Имя свойства: **margins**

Тип свойства: **DataTable**

ОПЕРАЦИИ

Список действий виджета, выполняемых при нажатии на кнопку.

Данный список может содержать ноль или более следующих операций:

- **Подтвердить**. Данное действие вызывает событие подтвердить 1044 корневой панели.
- **Выход**. Данное действие вызывает событие выход 1044 корневой панели.
- **Закрыть**. Останавливает работу виджета и закрывает его окно.

Имя свойства: **operations**

Тип свойства: **Data Table**

ПИКТОГРАММА

Изображение на кнопке. Если NULL, то изображение не показывается.

Имя свойства: **icon**

Тип свойства: **Data Block**

ПРОМЕЖУТОК МЕЖДУ ПИКТОГРАММОЙ И ТЕКСТОМ

Определяет пространство между текстом и иконкой.

Имя свойства: **iconTextGap**

Тип свойства: **Integer**

ПИКТОГРАММА ПРИ НАЖАТИИ

Изображение, отображающееся на кнопке при ее нажатии. Если NULL, отображается изображение по умолчанию.

Имя свойства: **pressedIcon**

Тип свойства: **Data Block**

ПИКТОГРАММА ПРИ ДЕАКТИВАЦИИ

Изображение, появляющееся при отключении кнопки. Если NULL, иконка отключения автоматически сформируется из обычной иконки.

Имя свойства: **disabledIcon**

Тип свойства: **Data Block**

ПИКТОГРАММА ПРИ НАВЕДЕНИИ

Изображение, появляющееся на кнопке при наведении на нее мыши. Если NULL, будет показано изображение по умолчанию.

Имя свойства: **rolloverIcon**

Тип свойства: **Data Block**

ПРОРИСОВАННЫЙ ФОКУС

Флажок, контролирующий пиктограмму внутри кнопки при получении фокуса.

Данную опцию лучше отключить, чтобы улучшить вид кнопки, состоящей только из иконки.

Имя свойства: **focusPainted**

Тип свойства: **Boolean**

ЗАКРАШИВАТЬ ОБЛАСТЬ

Флажок, контролирующий заполнение области кнопки.

Данную опцию следует отключить для улучшения вида кнопки, состоящей только из иконки.

Имя свойства: **contentAreaFilled**

Тип свойства: **Boolean**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1288, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

Пользовательские события

КЛИК

Данное событие возникает, когда пользователь кликает на кнопку.

Событие Клик должно использоваться для реакции на клики мыши вместо событий Клик мыши или Нажатие кнопки мыши. Эти события могут вызвать потерю нескольких кликов.

Имя события: **click**

13.4.9.9 Триггерная кнопка

Кнопка переключения между двумя состояниями. Триггерные кнопки используются внутри Групп кнопок 1040 для создания группы кнопок, в которой только одна кнопка может быть выбрана за один раз.

Триггерная кнопка выглядит следующим образом:

Если триггерная кнопка входит в группу кнопок, она может быть освобождена только нажатием другой кнопки в той же группе.

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Основной цвет 1275, Фон 1275, Непрозрачный 1275, Рамка 1275, Шрифт 1275, Всплывающая подсказка 1276, Фокусируемый 1276, Всплывающее меню 1276

Пользовательские свойства

ТЕКСТ (Свойство по умолчанию 1274)

Текст метки кнопки.

Имя свойства: **text**

Тип свойства: **String**

ВЫБРАН

Флажок, указывающий, что триггерная кнопка выбрана (нажата).

Имя свойства: **selected**

Тип свойства: **Boolean**

ВЫРАВНИВАНИЕ

Выравнивание по горизонтали текста внутри области отображения кнопки.

Возможные значения:

Описание	Значение
Слева	2
По центру	0
Справа	4

Имя свойства: **horizontalAlignment**

Тип свойства: **Integer**

ВЫРАВНИВАНИЕ ПО ВЕРТИКАЛИ

Выравнивание текста внутри области отображения кнопки по вертикали.

Возможные значения:

Описание	Значение
Сверху	1
По центру	0
Снизу	3

Имя свойства: **verticalAlignment**

Тип свойства: **Integer**

ПОЛЯ

Расстояние между границей и меткой с каждого края. Значение полей имеет следующие поля:

Свойство	Имя	Тип
Сверху	top	целое
Слева	left	целое
Снизу	bottom	целое
Справа	right	целое

Имя свойства: **margins**

Тип свойства: **DataTable**

ГРУППА КНОПОК

Группа кнопок 1040, к которой принадлежит кнопка.

Имя свойства: **buttonGroupID**

Тип свойства: **String**

ЗНАЧЕНИЕ

Значение, относящееся к триггерной кнопке.

Имя свойства: **value**

Тип свойства: **String**

ПИКТОГРАММА

Изображение на кнопке. Если значение является NULL, изображение не показывается.

Имя свойства: **icon**

Тип свойства: **Data Block**

ПРОМЕЖУТОК МЕЖДУ ПИКТОГРАММОЙ И ТЕКСТОМ

Определяет пространство между текстом и пиктограммой.

Имя свойства: **iconTextGap**

Тип свойства: **Integer**

ПИКТОГРАММА ПРИ НАЖАТИИ

Изображение, отображающееся на кнопке при ее нажатии. Если NULL, отображается изображение по умолчанию.

Имя свойства: **pressedIcon**

Тип свойства: **Data Block**

ПИКТОГРАММА ПРИ ДЕАКТИВАЦИИ

Изображение, появляющееся при отключении кнопки. Если NULL, иконка отключения автоматически сформируется из обычной иконки.

Имя свойства: **disabledIcon**

ВЫБРАННАЯ ПИКТОГРАММА

Изображение, появляющееся на кнопке, когда выбирается кнопка. Если NULL, отобразится изображение по умолчанию.

Имя свойства: **selectedIcon**

Тип свойства: **Data Block**

ПИКТОГРАММА ПРИ НАВЕДЕНИИ

Изображение, появляющееся на кнопке при наведении на нее мыши. Если NULL, будет показано изображение по умолчанию.

Имя свойства: **rolloverIcon**

Тип свойства: **Data Block**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1288, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.9.10 Зависимая кнопка

Элемент, который может иметь два состояния, выбран и не выбран, и отображать свое состояние пользователю. Зависимая кнопка используются внутри групп кнопок 1040, в которой одновременно может быть выбрана только одна.

Зависимая кнопка выглядит следующим образом:

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Основной цвет 1275, Фон 1275, Непрозрачный 1275, Рамка 1275, Шрифт 1275, Всплывающая подсказка 1276, Фокусируемый 1276, Всплывающее меню 1276

Пользовательские свойства

ТЕКСТ (Свойство по умолчанию 1274)

Текст метки зависимой кнопки

Имя свойства: **text**

Тип свойства: **String**

ГРУППА КНОПОК

Группа кнопок 1040, к которой принадлежит кнопка.

Имя свойства: **buttonGroupID**

Тип свойства: **String**

ЗНАЧЕНИЕ

Значение, относящееся к зависимой кнопке.

Имя свойства: **value**

Тип свойства: **String**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.9.11 Независимая кнопка

Элемент, который может быть выбран или выбор которого может быть отменен, и чье состояние отображается для пользователя.

Независимая кнопка выглядит следующим образом:

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Основной цвет 1275, Фон 1275, Непрозрачный 1275, Рамка 1275, Шрифт 1275, Всплывающая подсказка 1276, Фокусируемый 1276, Всплывающее меню 1276

Пользовательские свойства

ТЕКСТ(Свойство по умолчанию 1274)

Текст метки независимой кнопки.

Имя свойства: **text**

Тип свойства: **String**

ВЫБРАН

Флажок, указывающий, что кнопка была выбрана (отмечена).

Имя свойства: **selected**

Тип свойства: **Boolean**

ИЗОБРАЖЕНИЯ

Выбирает изображения в формате PNG для состояний "выбрано" и "не выбрано". Необходимо выбрать изображения для обоих состояний.

Имя свойства: **icons**

Тип свойства: **Data Table**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.9.12 Поле со списком

Компонент, сочетающий в себе редактируемое поле и раскрывающийся список. Пользователь может выбрать значение из раскрывающегося списка. Если вы составляете редактируемый раскрывающийся список, он будет включать в себя редактируемое поле, в которое пользователь может ввести значение.

Поле со списком выглядит следующим образом:

Тип 1274 свойства Выбранный элемент 975 и тип данных в его колонке **Значение** табличного свойства Допустимые значения 975 являются динамическими. Их невозможно вручную изменить с помощью Редактора виджетов, однако, они изменяются привязкой 1295, цель которой указывает на свойство Допустимые значения 975 раскрывающегося списка. Данная привязка перезаписывает таблицу **Допустимых значений**. Привязка автоматически создается во время перетаскивания таблицы со значениями на поле со списком 442.

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Основной цвет 1275, Фон 1275, Непрозрачный 1275, Рамка 1275, Шрифт 1275, Курсор 1276, Всплывающая подсказка 1276, Фокусируемый 1276, Всплывающее меню 1276

Пользовательские свойства

МАКСИМАЛЬНОЕ КОЛИЧЕСТВО ОТОБРАЖАЕМЫХ РЯДОВ

Задает максимальное число рядов, которые отображает раскрывающийся список. Если количество рядов превышает это число, в списке появится прокрутка.

Имя свойства: **maximumRowCount**

Тип свойства: **Integer**

ДОПУСТИМЫЕ ЗНАЧЕНИЯ

Список элементов в раскрывающемся списке:

Имя свойства: **options**

Тип свойства: **Data Table**

Можно заполнить раскрывающийся список допустимыми значениями 49 из полей заполненной таблицы данных при помощи привязки 1295. Например, привязка, которая заполнит раскрывающийся список странами из поля Страна переменной Информация о пользователе 1609, определенной в контексте Пользователь пользователя `admin`, будет выглядеть следующим образом:

Цель: `form/comboBox1:options`

Выражение: `{users.admin:childInfo$country#options}`

ВЫБРАННЫЙ ЭЛЕМЕНТ (Свойство по умолчанию 1274)

Элемент, выбранный по умолчанию.

Имя свойства: **selectedItem**

Тип свойства: **Dynamic**

ЦВЕТ ВЫБРАННОГО ЭЛЕМЕНТА (Свойство по умолчанию 1274)

Определяет увет выбранного элемента.

Имя свойства: **selectedItemColor**

Тип свойства: **Color**

ИСПОЛЬЗОВАТЬ ПОЛЬЗОВАТЕЛЬСКИЙ ЦВЕТ КНОПОК

Активирует меню пользовательских свойств кнопок, определяемых свойством **Свойства кнопок**.

Имя свойства: **useCustomButtonProperties**

Тип свойства: **Boolean**

СВОЙСТВА КНОПОК

Задает цвет фона, основной цвет и цвет границ кнопки с выпадающим списком.

Имя свойства: **buttonProperties**

Тип свойства: **Data Table**

Поля Свойств кнопок:

Имя поля	Тип	Описание
foreground	Color	Основной цвет кнопки с выпадающим списком
background	Color	Цвет фона кнопки с выпадающим списком
border	Color	Цвет границ кнопки с выпадающим списком

С ВОЗМОЖНОСТЬЮ ПОИСКА

Активирует автодополнение ввода значений, вводимых в поле со списком.

Имя свойства: **searchable**

Тип свойства: **Boolean**

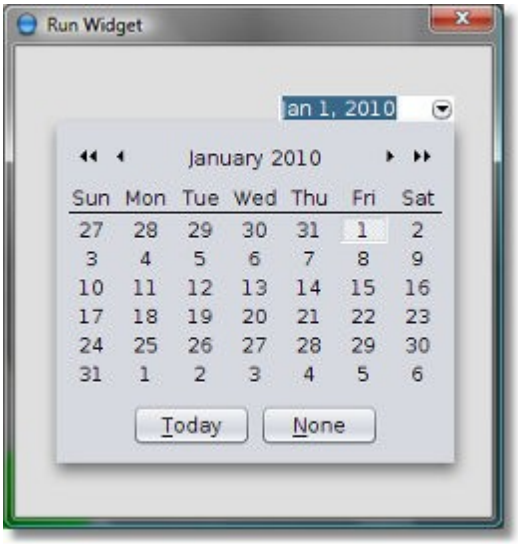
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.9.13 Дата/время

Компонент, используемый для выбора даты при помощи раскрываемого диалогового окна. Он сочетает в себе редактируемое поле и диалоговое окно.

Компонент дата/время выглядит следующим образом:



Компонент поддерживает следующие операции ввода с клавиатуры:

Клавиша	Действие
ALT + DOWN	Показывает диалоговое окно.
ESC	Скрывает окно без изменения выбора.
LEFT	Предыдущий день до выбранного на данный момент.
RIGHT	Следующий день после выбранного на данный момент.
UP	Тот же день на прошлой неделе.
DOWN	Тот же день на следующей неделе.
HOME	Первый день текущего месяца.
END	Последний день текущего месяца.
PAGE_UP	Тот же день прошлого месяца.
PAGE_DOWN	Тот же день следующего месяца.
CTRL + PAGE_UP	Тот же день прошлого года.
CTRL+ PAGE_DOWN	Тот же день следующего года
ENTER	Выбирает выделенную дату и закрывает диалоговое окно.

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Основной цвет 1275, Фон 1275, Непрозрачный 1275, Рамка 1275, Шрифт 1275, Всплывающая подсказка 1276, Фокусируемый 1276, Всплывающее меню 1276

Пользовательские свойства

ДАТА (Свойство по умолчанию 1274)

Выбранная дата.

Имя свойства: **date**

Тип свойства: **Date**

РЕДАКТИРУЕМЫЙ

Определяет, является ли текстовое поле редактируемым. Если установлено на false, дату можно выбирать только из раскрываемого диалогового окна, а не при помощи клавиатуры.

Имя свойства: **editable**

Тип свойства: **Boolean**

ФОРМАТ

Формат даты/времени 2146 для значения, представленного в поле.

Имя свойства: **format**

Тип свойства: **String**

МИНИМАЛЬНАЯ ДАТА

Минимально допустимая дата.

Имя свойства: **minDate**

Тип свойства: **Date**

МАКСИМАЛЬНАЯ ДАТА

Максимально допустимая дата.

Имя свойства: **maxDate**

Тип свойства: **Date**

ОТОБРАЖАЕМОЕ ВРЕМЯ

Определяет, будет ли показываться редактируемое поле времени в диалоговом окне выбора даты. Включение данного свойства позволяет задать время вместе с датой.

Имя свойства: **timeDisplayed**

Тип свойства: **Boolean**

ФОРМАТ ВРЕМЕНИ

Определяет формат времени 2146, показываемого в диалоговом окне.

Имя свойства: **timeFormat**

Тип свойства: **String**

ОТОБРАЖАТЬ КНОПКУ "СЕГОДНЯ"

Контролирует видимость кнопки "Сегодня".

Имя свойства: **showTodayButton**

Тип свойства: **Boolean**

ОТОБРАЖАТЬ КНОПКУ "НЕТ ДАТЫ"

Контролирует видимость кнопки "Нет даты". Данная кнопка позволяет выбирать пустое значение (NULL) даты.

Имя свойства: **showNoneButton**

Тип свойства: **Boolean**

ОТОБРАЖАТЬ КНОПКУ "ОК"

Контролирует видимость кнопки OK.

Имя свойства: **showOKButton**

Тип свойства: **Boolean**

ОТОБРАЖАТЬ НОМЕРА ДНЕЙ НЕДЕЛИ

Определяет видимость номеров дней недели на панели компонента.

Имя свойства: **editable**

Тип свойства: **Boolean**

ИСПОЛЬЗОВАНИЕ ПОЛЬЗОВАТЕЛЬСКИХ ЦВЕТОВ КНОПОК

Позволяет пользователю выбирать цвета кнопок.

Имя свойства: **useCustomButtonProperties**

Тип свойства: **Boolean**

СВОЙСТВА КНОПОК

Настраивает цвет фона и основной цвет кнопок для элементов Дата/Время пользовательского интерфейса.

Имя свойства: **buttonProperties**

Тип свойства: **Data Table**

Общие события

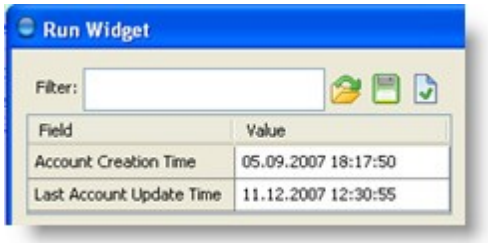
Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.9.14 Редактор таблицы данных

Компонент "Редактор таблицы данных" предназначен для просмотра и редактирования табличных данных (таблиц данных 49).

Более подробную информацию о внешнем виде и поведении данного компонента см. здесь 382.

Компонент "Редактор таблицы данных" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Рамка 1275

Пользовательские свойства

ТУЛБАР

Контролирует отображение тулбара Редактора таблицы данных.

Имя свойства: **toolbar**

Тип свойства: **Boolean**

РАЗРЕШИТЬ ВЕРТИКАЛЬНЫЙ РЕЖИМ

Если включен, то позволяет редактору перейти в вертикальный (с одной записью) режим 382, если формат таблицы имеет как минимальное, так и макимальное количество записей, выставленных на один.

Имя свойства: **allowVerticalMode**

Тип свойства: **Boolean**

ВКЛЮЧИТЬ ВСПЛЫВАЮЩИЕ ПОДСКАЗКИ

Контролирует отображение всплывающих подсказок Редактора таблицы данных.

Имя свойства: **enableTooltips**

Тип свойства: **Boolean**

ВКЛЮЧИТЬ КОНТЕКСТНОЕ МЕНЮ

Контролирует отображение контекстного меню Редактора таблицы данных.

Имя свойства: **enablePopupMenu**

Тип свойства: **Boolean**

ОТОБРАЖАТЬ МЕТАДАННЫЕ

Контролирует отображение метаданных Редактора таблицы данных.

Имя свойства: **showMetadata**

Тип свойства: **Boolean**

ПОСТОЯННЫЙ ФОРМАТ

Определяет, должен ли Редактор таблицы данных принимать только таблицы заранее установленного формата. Если эта опция включена, возможно редактировать свойство **Формат** (см. ниже) и после определения формата таблицы заполнить таблицу заранее определенными данными путем редактирования свойства **Таблица данных** (также описана ниже).

В момент включения этого флага данные в свойстве **Таблица данных** конвертируются, чтобы соответствовать данному значению **Формата**.

Имя свойства: **fixedFormat**

Тип свойства: **Boolean**

ФОРМАТ

Позволяет указать постоянный формат таблицы, чтобы можно было его просматривать/редактировать в Редакторе Таблицы Данных. Смотрите описание опций формата и отдельные опции поля в разделе Формат таблицы данных 50.

В момент, когда меняется **Формат**, данные в свойстве **Таблица данных** конвертируются, чтобы соответствовать новому формату.

Имя свойства: **format**

Тип свойства: **Data Table**

ТАБЛИЦА ДАННЫХ (Свойство по умолчанию 1274)

Таблица данных по умолчанию, отображаемая и редактируемая этим компонентом.

Когда знчение этой опции меняется на новую Таблицу данных, система преобразует данную таблицу, чтобы она соответствовала свойству **Формат** (это происходит только если флажок устновлен на **Фиксированный формат**).

Имя свойства: **dataTable**

Тип свойства: **Data Table**

ВИДИМОСТЬ ГОРИЗОНТАЛЬНЫХ ЛИНИЙ СЕТКИ

Активирует отображение горизонтальных линий сетки.

Имя свойства: **showHorizontalGrid**

Тип свойства: **Boolean**

ВИДИМОСТЬ ВЕРТИКАЛЬНЫХ ЛИНИЙ СЕТКИ

Активирует отображение вертикальных линий сетки.

Имя свойства: **showVerticalGrid**

Тип свойства: **Boolean**

СВОЙСТВА ЗАГОЛОВКА ТАБЛИЦЫ

Определяет видимость и перенос строк заголовка таблицы, а также его основной цвет и цвета фона.

Имя свойства: **headerProperties**

Тип свойства: **Data Table**

ПЕРЕНОС СТРОК

Определяет пепенос строк в ячейках таблицы.

Имя свойства: **lineWrap**

Тип свойства: **Boolean**

ОТОБРАЖАТЬ НОМЕРА ЗАПИСЕЙ

Определяет видимость столбца с номерами записей.

Имя свойства: **showRecordNumbers**

Тип свойства: **Boolean**

ЦВЕТ ВЫБРАННОГО РЯДА

Определяет пользовательский цвет для выбранных рядов.

Имя свойства: **selectedRowTable**

Тип свойства: **Data Table**

ВЫРАЖЕНИЕ ОБРАБОТКИ ДОБАВЛЕННОЙ ЗАПИСИ

Выражение, вычисляемое для обработки каждой новой записи таблицы. Используется при выполнении функции **Обработать изменения**. Для более подробной информации см. раздел документации о функции Обработать изменения.

Имя свойства: **addedRecordProcessingExpression**

Тип свойства: **String**

ВЫРАЖЕНИЕ ОБРАБОТКИ ИЗМЕНЕННОЙ ЗАПИСИ

Выражение, вычисляемое для обработки каждой измененной записи таблицы. Используется при выполнении функции **Обработать изменения**. Для более подробной информации см. раздел документации о функции Обработать изменения.

Имя свойства: **changedRecordProcessingExpression**

Тип свойства: **String**

ВЫРАЖЕНИЕ ОБРАБОТКИ УДАЛЕННОЙ ЗАПИСИ

Выражение, вычисляемое для обработки записей, удаленных из таблицы. Используется при выполнении функции **Обработать изменения**. Для более подробной информации см. раздел документации о функции Обработать изменения.

Записи, обрабатываемые при помощи данного выражения, не видны в таблице. Они хранятся во внутреннем списке удаленных записей.

Имя свойства: **removedRecordProcessingExpression**

Тип свойства: **String**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285

Пользовательские функции

ОБРАБОТАТЬ ИЗМЕНЕНИЯ

Функция не имеет входных параметров. Она дает команду редактору таблиц обработать внутренний список добавленных, измененных или удаленных записей:

- Для каждой добавленной записи (даже если позднее она была изменена), вычисляется **Выражение обработки добавленной записи**. Если выражение возвращает иные значения, кроме `false`, внутренний

флажок записи "новая запись" убирается. Выражение может, например, вызывать функцию Выполнить запрос аккаунта устройства SQL базы данных 647 на стороне сервера, чтобы произвести INSERT новой записи в произвольную базу данных.

- Для каждой измененной записи, вычисляется **Выражение обработки измененной записи**. Если выражение возвращает иные значения, кроме `false`, внутренний флажок записи "измененная запись" убирается. Выражение может, например, выполнить UPDATE запрос для внешней базы данных, подключенной к AtomMind Server через драйвер устройства SQL базы данных.
- Для каждой удаленной записи (за исключением записей, которые были предварительно добавлены в сам Редактор таблиц данных), вычисляется **Выражение обработки удаленной записи**. Если выражение возвращает иные значения, кроме `false`, запись исключается из внутреннего списка удаленных записей. Выражение может, например, выполнить DELETE-запрос для внешней базы данных, чтобы удалить запись базы данных с ключом, взятым из данных записи.

Added/Changed/Removed Record Processing Expression Среда вычисления 114 Выражения обработки добавленной/измененной/удаленной записи:	
Контекст по умолчанию 119	Контекст по умолчанию 946 виджета.
Таблица данных по умолчанию 120	Добавленная/измененная/удаленная обрабатываемая запись. Запись помещается в таблицу с одной ячейкой. Обратите внимание, что таблица по умолчанию НЕ является целой таблицей, хранящейся в Редакторе таблиц данных.
Ряд по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

Связанный самоучитель:

- Управление таблицей внешней базы данных 1726

Имя функции: **processChanges**

Пользовательские события

ВЫБОР СТРОКИ

Это событие формируется, когда выбираются или не выбираются какие-либо строки таблицы.

Имя события: **rowSelection**

Поля события:

Поле	Имя	Тип	Описание
Is Adjusting	isAdjusting	Boolean	Возвращает одно событие из серии множества событий, в которых все еще происходят изменения.
First Row	firstRow	Integer	Индекс первой строки, чей выбор мог измениться.
Last Row	lastRow	Integer	Индекс последней строки, чей выбор мог измениться.
First Record	firstRecord	Integer	Индекс первой строки записи Таблица данных, чей выбор мог измениться. Может отличаться от Первой Строки из-за сортировки, фильтрации, группировки таблицы и т.д.
Last Record	lastRecord	Integer	Индекс последней строки записи Таблица данных, чей выбор мог измениться. Может отличаться от Последней Строки из-за сортировки, фильтрации, группировки таблицы и т.д.
Selection	selection	Data Table	Список выбираемых строк, с двумя полями: • selectionRow (целое) - индекс выбираемой строки • selectionRecord (целое) - индекс выбираемой записи

КЛИКНУТА ССЫЛКА

Это событие возникает при клике на ссылку в таблице.

Имя события: **referenceClicked**

Поля события:

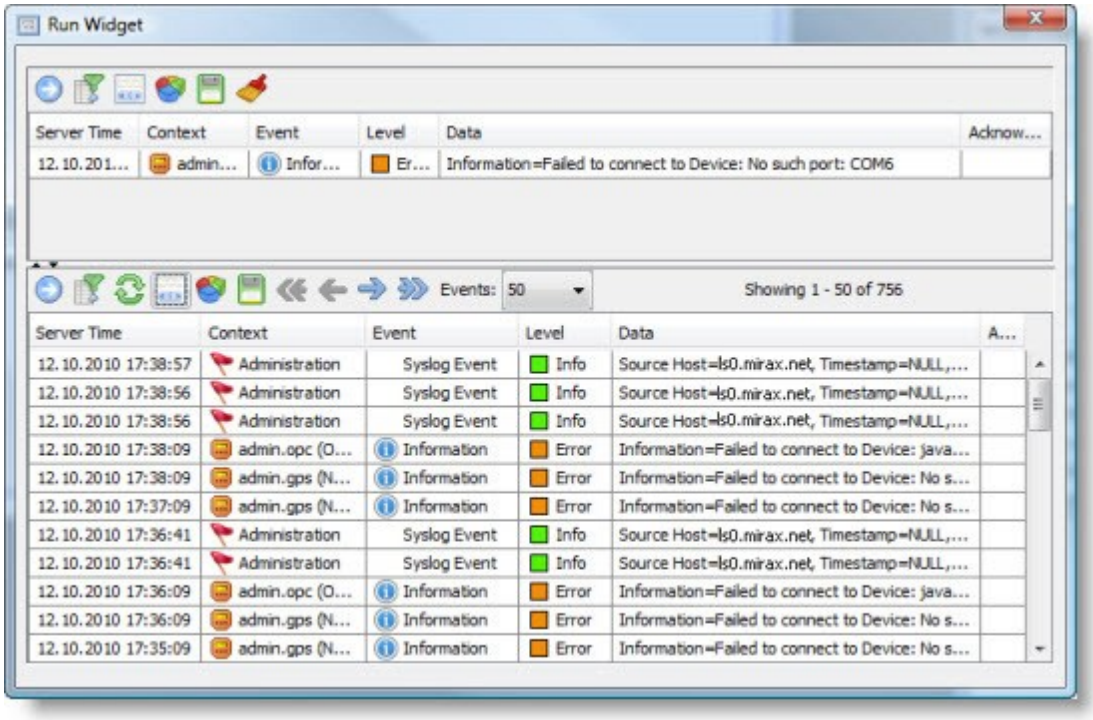
Поле	Имя	Тип	Описание
Reference	reference	String	Возвращает выполненную ссылку.
Field	field	String	Возвращает имя поля ячейки, где помещалась ссылка.
Record	record	Integer	Возвращает номер записи ячейки с выполненной ссылкой.
Data	data	Data Table	Возвращает значение ячейки с запущенной ссылкой.

13.4.9.15 Лог событий

Компонент "Лог событий" предназначен для просмотра и управления событиями 73 AtomMind.

Более подробную информацию о внешнем виде и поведении данного компонента см. здесь 398.

Лог событий выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Рамка 1275

Пользовательские свойства

РЕЖИМ

Компонент "Лог событий" может работать в двух основных режимах:

- **Список событий**. В данном режиме он показывает только те события, которые определены свойством "события". Видимые столбцы задаются свойством компонента "Лог событий" (см. далее). Дополнительные поля не показываются, фильтрация и выделение не выполняются.
- **Фильтр событий**. В данном режиме "Лог событий" активирует серверный фильтр событий 762, заданный настройкой "Фильтр событий", и использует данный фильтр для выбора и выделения событий. Столбцы отображаются в соответствии с настройкой фильтра.

Имя свойства: **mode**

Тип свойства: **Integer**

СОБЫТИЯ

Таблица, которая определяет, какие события должны отображаться. Она состоит из двух столбцов:

- **Маска контекстов**. Показаны только те события, которые возникли в контекстах, соответствующих данной маске 44.
- **Событие**. Имя события.

Имя свойства: **eventList**

Тип свойства: **Data Table**

ФИЛЬТР СОБЫТИЙ

Используемый путь контекста фильтра событий 762.

Если значение настройки является NULL, компонент "Лог событий" позволит выбрать фильтр из контекстного списка.

Имейте в виду, что для разделов "Текущие события" и "История событий" могут быть выбраны различные фильтры.

Имя свойства: **filter**

Тип свойства: **String**

РАЗДЕЛЫ

Определяет, какие разделы будут отображаться: **Текущие события**, **История событий** и **Оба раздела**.

Имя свойства: **sections**

Тип свойства: **Integer**

ПРЕДВАРИТЕЛЬНО ЗАГРУЖАТЬ ИСТОРИЮ СОБЫТИЙ

Определяет загрузку истории событий при запуске виджета.

Имя свойства: **preloadHistory**

Тип свойства: **Boolean**

ПОКАЗЫВАТЬ ИМЕНА КОНТЕКСТОВ

Флажок, определяющий видимость столбца "Контекст".

Имя свойства: **showContexs**

Тип свойства: **Boolean**

ПОКАЗЫВАТЬ ИМЕНА СОБЫТИЙ

Флажок, определяющий видимость столбца "Имя события".

Имя свойства: **showNames**

Тип свойства: **Boolean**

ПОКАЗЫВАТЬ УРОВНИ СОБЫТИЙ

Флажок, определяющий видимость столбца "Уровень".

Имя свойства: **showLevels**

Тип свойства: **Boolean**

ПОКАЗЫВАТЬ ДАННЫЕ СОБЫТИЙ

Флажок, определяющий видимость столбца "Данные".

Имя свойства: **showData**

Тип свойства: **Boolean**

ПОКАЗЫВАТЬ ПОДТВЕРЖДЕНИЯ СОБЫТИЙ

Флажок, определяющий видимость столбца "Подтверждения".

Имя свойства: **showAcknowledgements**

Тип свойства: **Boolean**

ПОКАЗЫВАТЬ ОБОГАЩЕНИЯ СОБЫТИЙ

Флажок, определяющий видимость столбца "Обогащения".

Имя свойства: **showEnrichments**

Тип свойства: **Boolean**

ПАРАМЕТРЫ ФИЛЬТРА

Таблица параметров, используемая для параметризованного фильтра 772. Это свойство обычно записывается во время работы привязкой, которая читает таблицу параметров фильтра, используя функцию getParameters() 1509 контекста **Фильтр Событий** и заполняет эту таблицу по ходу работы (например, при помощи функции языка выражений 124 `set()`).

Имя свойства: **filterParameters**

Тип свойства: **Data Table**

ПРЕДПОЧТИТЕЛЬНОЕ ДЕЙСТВИЕ

Действие, запускаемое кликом на исходный контекст события. Если не указано, будет запущено действие по умолчанию 88 контекста.

Имя свойства: **preferredAction**

Тип свойства: **String**

СВОЙСТВА ТЕКУЩИХ СОБЫТИЙ

Настраивает цвета фона и изображения, видимость линий сетки и подсказок, свойства заголовка и кнопок раздела Текущие события в Логе событий.

Имя свойства: **realTimeProperties**

Тип свойства: **Data Table**

СВОЙСТВА ИСТОРИИ СОБЫТИЙ

Настраивает цвета фона и изображения, видимость линий сетки и подсказок, свойства заголовка и кнопок для раздела истории Лога событий.

Имя свойства: **historyProperties**

Тип свойства: **Data Table**

Общие события

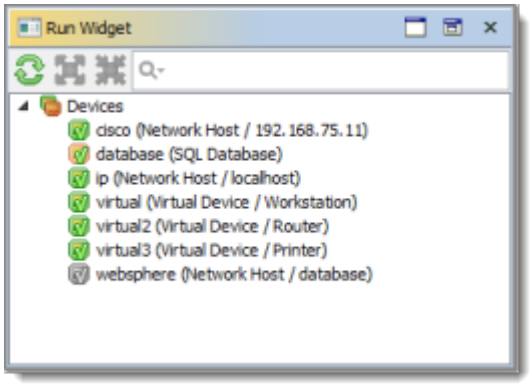
Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.9.16 Системное дерево

Компонент "Системное дерево" позволяет операторам виджетов управлять частью контекстного дерева 41 AtomMind Server.

Дополнительную информацию о том, как выглядит и ведет себя этот компонент можно найти здесь 370.

Системное дерево выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Рамка 1275

Пользовательские свойства

КОРЕНЬ

Определяет путь контекста, который будет использовать корень контекстного поддерева, представленный компонентом "Системное дерево".

Имя свойства: **root**

Тип свойства: **String**

ПОКАЗАТЬ ПАНЕЛЬ ИНСТРУМЕНТОВ

Определяет видимость панели инструментов системного дерева.

Имя свойства: **showToolbar**

Тип свойства: **Boolean**

КОНТЕКСТНОЕ МЕНЮ

Определяет видимость контекстного меню.

Имя свойства: **contextMenu**

Тип свойства: **Boolean**

ДЕЙСТВИЯ

Определяет видимость и расположение панели Действия: Скрытый, Снизу или Справа.

Имя свойства: **relatedActions**

Тип свойства: **Integer**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

Пользовательские события

ВЫБОР ДЕРЕВА

Это событие формируется, когда выбираются или не выбираются какие-либо узлы дерева.

Имя события: **treeSelection**

Поля события:

Поле	Имя	Тип	Описание

Добавляется	isAdded	Boolean	True, если первый элемент пути был добавлен к выбору, и false, если первый путь был удален из выбора.
Локальный путь	localPath	String	Локальный путь Системного Дерева для узла, добавленного/удаленного из выбора.
Удаленный путь	remotePath	String	Контекстный путь удаленного сервера для узла, добавленного/удаленного из выбора.
Выбор	selection	Data Table	Список отбираемых узлов, с двумя полями: • localSelectionPath (String) - Локальный путь Системного Дерева для выбираемого узла. • remoteSelectionPath (String) - Контекстный путь удаленного сервера для выбираемого узла.

13.4.9.17 Регулятор

Позволяет пользователю выбирать значение, графически перемещая ползунок регулятора в ограниченном интервале. Регулятор может указывать на большие метки деления, а также на малые между ними. Количество значений между метками деления контролируется свойствами "Расстояние между основными делениями" и "Расстояние между второстепенными делениями".

Регулятор выглядит следующим образом:

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Основной цвет 1275, Фон 1275, Непрозрачный 1275, Рамка 1275, Шрифт 1275, Всплывающая подсказка 1276, Фокусируемый 1276, Всплывающее меню 1276

Пользовательские свойства

ЗНАЧЕНИЕ (Свойство по умолчанию 1274)

Значение регулятора по умолчанию.

Имя свойства: **value**

Тип свойства: **Integer**

ОРИЕНТАЦИЯ

Положение регулятора: **горизонтально** или **вертикально**.

Имя свойства: **orientation**

Тип свойства: **Integer**

МИНИМУМ

Минимальное значение регулятора.

Имя свойства: **minimum**

Тип свойства: **Integer**

МАКСИМУМ

Максимальное значение регулятора.

Имя свойства: **maximum**

Тип свойства: **Integer**

РАССТОЯНИЕ МЕЖДУ ВТОРОСТЕПЕННЫМИ ДЕЛЕНИЯМИ

Расстояние, измеряемое в значениях, между каждой разметкой малых делений. Если интервал вашего ползунка ограничен значениями от 0 до 50 и размещение малых делений установлено на 10, малые деления будут около 0, 10, 20, 30, 40, и 50.

Имя свойства: **minorTickSpacing**

Тип свойства: **Integer**

РАССТОЯНИЕ МЕЖДУ ОСНОВНЫМИ ДЕЛЕНИЯМИ

Расстояние, измеряемое в значениях, между каждой разметкой больших делений. Если интервал вашего ползунка ограничен значениями от 0 до 50 и размещение больших делений установлено на 10, большие деления будут около 0, 10, 20, 30, 40, и 50.

Имя свойства: **majorTickSpacing**

Тип свойства: **Integer**

ПОКАЗЫВАТЬ ШКАЛУ

Определяет, будет ли отображаться шкала регулятора.

Имя свойства: **paintTrack**

Тип свойства: **Boolean**

ПОКАЗЫВАТЬ РАЗМЕТКУ

Определяет, будет ли отображаться разметка делений регулятора.

Имя свойства: **paintTicks**

Тип свойства: **Boolean**

ПОКАЗЫВАТЬ МЕТКИ

Определяет, будут ли отображаться метки регулятора.

Имя свойства: **paintLabels**

Тип свойства: **Boolean**

ПОЛЬЗОВАТЕЛЬСКИЕ МЕТКИ

Таблица, содержащая пользовательские метки регулятора. Каждая метка характеризуется значением регулятора, согласно которому она отображается.

Имя свойства: **customLabels**

Тип свойства: **Data Table**

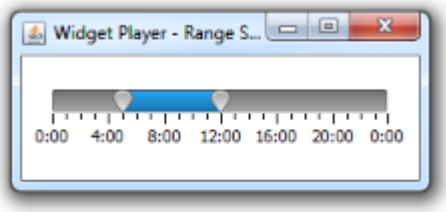
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.9.18 Регулятор диапазона

Компонент, который позволяет пользователю выбирать диапозон чисел или дат, передвигая кнопки в пределах ограниченного интервала. Регулятор может показывать как основные метки делений, так и второстепенные метки делений между ними. Количество значений между метками делений контролируют свойства расстояния между основными делениями и свойства расстояния между второстепенными делениями.

Вот так выглядит регулятор:



Основные свойства

Ширина 1274, Высота 1274, Привязки 1275, Активен 1275, Видимый 1275, Передний план 1275, Фон 1275, Непрозрачный 1275, Граница 1275, Шрифт 1275, Курсор 1276, Всплывающая подсказка 1276, Фокусируемый 1276, Всплывающее меню 1276

Дополнительные свойства

ОРИЕНТАЦИЯ

Ориентация регулятора: Горизонтальная или Вертикальная.

Имя свойства: **orientation**

Тип свойства: **Целое**

ПОКАЗЫВАТЬ ДОРОЖКУ

Определяет, где на регуляторе показана дорожка.

Имя свойства: **paintTrack**

Тип свойства: **Логическое**

ПОКАЗЫВАТЬ МЕТКИ

Определяет, где на регуляторе показаны метки делений.

Имя свойства: **paintTicks**

Тип свойства: **Логическое**

ПОКАЗЫВАТЬ ПОДПИСИ

Определяет, показаны ли на регуляторе подписи.

Имя свойства: **paintLabels**

Тип свойства: **Логическое**

ПОКАЗЫВАТЬ ДИАПАЗОН

Определяет видимость диапазона.

Имя свойства: **rangeVisible**

Тип свойства: **Логическое**

ВЫБОР ДИАПАЗОНА

Делает возможным выбор диапазона.

Имя свойства: **rangeSelectionEnabled**

Тип свойства: **Логическое**

ФОРМА ДИАПАЗОНА

Форма регулятора: **Круг**, **Квадрат, Прямоугольник**, **Капля** или **Отсутствует**.

Имя свойства: **thumbShape**

Тип свойства: **Строка**

ДИЗАЙН РЕГУЛЯТОРА

Дизайн регулятора: **Светлый**, **Темный**, **Сталь** или **Темная сталь**.

Имя свойства: **thumbDesign**

Тип свойства: **Строка**

ШИРИНА ДОРОЖКИ

Определяет ширину дорожки: **Тонкий**, **Средний** или **Толстый**.

Имя свойства: **thumbDesign**

Тип свойства: **Строка**

ТЕМНАЯ ДОРОЖКА

Позволяет устанавливать темный фон для регулятора.

Имя свойства: **darkTrack**

Тип свойства: **Логическое**

ЦВЕТ ДИАПАЗОНА

Определяет цвет диапазона.

Имя свойства: **rangeColor**

Тип свойства: **Цвет**

НЕСТАНДАРТНЫЕ ПОДПИСИ

Таблица, содержащая нестандартные подписи для регулятора. Каждая подпись характеризуется описанием подписи и значением диапазона, на котором она показана.

Имя свойства: **customLabels**

Тип свойства: **Таблица данных**

ТИП ДИАПАЗОНА

Определяет тип диапазона: **Числовой диапазон** или **Диапазон дат**.

Имя свойства: **rangeType**

Тип свойства: **Целое**

МИНИМУМ

Минимальное значение регулятора.

Имя свойства: **minimum**

Тип свойства: **Целое**

МАКСИМУМ

Максимальное значение регулятора.

Имя свойства: **maximum**

Тип свойства: **Целое**

РАССТОЯНИЕ МЕЖДУ ВТОРОСТЕПЕННЫМИ ДЕЛЕНИЯМИ

This is the distance, measured in values, between each minor tick mark. If you have a slider with a range from 0 to 50 and the minor tick spacing is set to 10, you will get minor ticks next to 0, 10, 20, 30, 40, and 50.

Имя свойства: **minorTickSpacing**

Тип свойства: **Целое**

РАССТОЯНИЕ МЕЖДУ ОСНОВНЫМИ ДЕЛЕНИЯМИ

Это расстояние, измеряемое в значениях, между каждой основной меткой деления. Если у вас регулятор с диапазоном от 0 до 50, и расстояние между основными делениями выставлено 10, вы получите следующие основные деления: 0, 10, 20, 30, 40 и 50.

Имя свойства: **majorTickSpacing**

Тип свойства: **Целое**

НИЖНЕЕ ЗНАЧЕНИЕ (Свойство по умолчанию 1274)

Нижнее значение на регуляторе по умолчанию.

Имя свойства: **lowerValue**

Тип свойства: **Целое**

ВЕРХНЕЕ ЗНАЧЕНИЕ

Верхнее значение на регуляторе по умолчанию.

Имя свойства: **upperValue**

Тип свойства: **Целое**

МИНИМАЛЬНАЯ ДАТА

Минимальная дата регулятора.

Имя свойства: **minimumDate**

Тип свойства: **Дата**

МАКСИМАЛЬНАЯ ДАТА

Максимальная дата регулятора.

Имя свойства: **maximumDate**

Тип свойства: **Дата**

РАССТОЯНИЕ МЕЖДУ ВТОРОСТЕПЕННЫМИ ДЕЛЕНИЯМИ

Временной интервал между каждой второстепенной меткой деления.

Имя свойства: **minorDateTickSpacing**

Тип свойства: **Длинное целое**

РАССТОЯНИЕ МЕЖДУ ОСНОВНЫМИ ДЕЛЕНИЯМИ

Временной интервал между каждой основной меткой деления.

Имя свойства: **majorDateTickSpacing**

Тип свойства: **Длинное целое**

НИЖНЯЯ ДАТА

Значение для нижней даты диапазона по умолчанию.

Имя свойства: **lowerDate**

Тип свойства: **Дата**

ВЕРХНЯЯ ДАТА

Значение для верхней даты диапазона по умолчанию.

Имя свойства: **upperDate**

Тип свойства: **Дата**

ТЕКУЩИЕ ИЗМЕНЕНИЯ

Если это свойство активно, значение диапазона обновляется при каждом движении регулятора. Если неактивно, обновления происходят, только когда отпущена кнопка мыши.

Имя свойства: **liveChanges**

Тип свойства: **Логическое**

Общие события компонентов

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287.

13.4.9.19 Счетчик

Одна строка поля ввода позволяет выбрать пользователю номер или значение из упорядоченной последовательности. Счетчики включают в себя две маленькие кнопки в виде стрелок, нажатие которых обеспечивает пролистывание элементов последовательности. Нажатие клавиш клавиатуры "вверх"/"вниз" также может использоваться для пролистывания элементов. Пользователь также может ввести (допустимое) значение напрямую в счетчик.

Счетчик выглядит следующим образом:

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Основной цвет 1275, Фон 1275, Непрозрачный 1275, Рамка 1275, Шрифт 1275, Всплывающая подсказка 1276, Фокусируемый 1276, Всплывающее меню 1276

Пользовательские свойства

ЗНАЧЕНИЕ (Свойство по умолчанию 1274)

Значение по умолчанию счетчика.

Имя свойства: **value**

Тип свойства: **Integer**

ИСПОЛЬЗОВАНИЕ НАСТРАИВАЕМОГО ЦВЕТА КНОПОК

Позволяет настроить цвета кнопок.

Имя свойства: **useCustomArrowButton**

Тип свойства: **Boolean**

СВОЙСТВА КНОПОК

Задает основной цвет и цвет фона для кнопок со стрелками.

Имя свойства: **arrowButtonProperties**

Тип свойства: **Data Table**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.9.20 Индикатор выполнения

Компонент, который по умолчанию отображает целочисленное значение в пределах заданного интервала. Индикатор выполнения обычно показывает процесс выполнения какой-либо задачи в процентах.

Индикатор выполнения выглядит следующим образом:

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Рамка 1275, Всплывающая подсказка 1276, Фокусируемый 1276, Всплывающее меню 1276

Пользовательские свойства

ЗНАЧЕНИЕ (Свойство по умолчанию 1274)

Свойство по умолчанию индикатора выполнения.

Имя свойства: **value**

Тип свойства: **Integer**

ОРИЕНТАЦИЯ

Ориентация индикатора выполнения: **горизонтально** или **вертикально**.

Имя свойства: **orientation**

Тип свойства: **Integer**

МИНИМУМ

Минимальное значение индикатора выполнения.

Имя свойства: **minimum**

Тип свойства: **Integer**

МАКСИМУМ

Максимальное значение индикатора выполнения.

Имя свойства: **maximum**

Тип свойства: **Integer**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.9.21 Изображение

Данный компонент показывает изображение.

Компонент "Изображение" выглядит следующим образом:

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Рамка 1275, Всплывающая подсказка 1276, Фокусируемый 1276, Всплывающее меню 1276

Пользовательские свойства

ВЫРАВНИВАНИЕ

Выравнивание текста по горизонтали в области изображения.

Возможные значения:

Описание	Значение
Слева	2
По центру	0
Справа	4

Имя свойства: **horizontalAlignment**

Тип свойства: **Integer**

ВЫРАВНИВАНИЕ ПО ВЕРТИКАЛИ

Выравнивание текста по вертикали в области изображения.

Возможные значения:

Описание	Значение
Сверху	1
По центру	0
Снизу	3

Имя свойства: **verticalAlignment**

Тип свойства: **Integer**

ТАБЛИЦА ИЗОБРАЖЕНИЙ

Далее приведена таблица свойств, состоящая из двух столбцов: **ID изображения** и **Данные изображения.** Для более подробной информации см. Выражение изображения 995.

Поле	Тип	Описание
идентификатор изображения	String	Уникальный идентификатоор изображения в данной таблице.
Данные изображения	Data Block	Данные изображения

Имя свойства: **imageTable**

Тип свойства: **Data Table**

Пример: Для отображения фотографии пользователя admin в компоненте изображения создайте следующую привязку:

Цель: `form/image1:imageTable$imageData[0]`

Выражение: `{users.admin:photo$photo}`

ВЫРАЖЕНИЕ ИЗОБРАЖЕНИЯ

Когда выражение изображения задано, оно оценивается следующим образом:

- Если выражение оценивается как строка (String), изображение с идентификатором, совпадающим с данной строкой, выбирается из таблицы изображений и показывается данным компонентом.
- Если выражение оценивается как изображение (Image), оно автоматически показывается компонентом. Для более подробной информации о том, как выражение разрешается в изображение, обратитесь к разделам свойства определения переменной и свойства контекста в статье "Ссылки".

Имя свойства: **expression**

Тип свойства: **String**

Поскольку выражение изображения не является частью привязки виджета, оно не будет пересчитываться, когда изменяются свойства и компоненты, которые на него ссылаются. Оно будет пересчитано только в следующих случаях:

- При запуске виджета
- Когда изменяется само свойство "Выражение изображения" привязкой виджета.

Таким образом, если вы хотите, чтобы содержание изображения изменялось при определенном событии, сделайте следующее:

- Создайте виджет, привязав цель к выражению изображения, например: `form/image1:expression`
- Создайте выражение привязки, что создаст другое выражение (выражение изображения), которое будет использовано для нахождения изображения в таблице изображений. Например, чтобы выбрать изображение согласно значению, выбранному **в поле со списком**, используйте следующее выражение привязки: `"'" + {form/comboBox1:selectedItem} + "'"`. Оно составит следующую строку выражения для получения изображения: `'image1'`.

Чтобы сделать анимированное изображение, добавьте два изображения в таблицу изображений и создайте периодическую привязку:

Цель: `form/image1:expression`

Выражение: `{form/image1:expression}=="'image1'"?"'image2'":"'image1'"`

Данная привязка будет переключать изображение при каждом его выполнении.

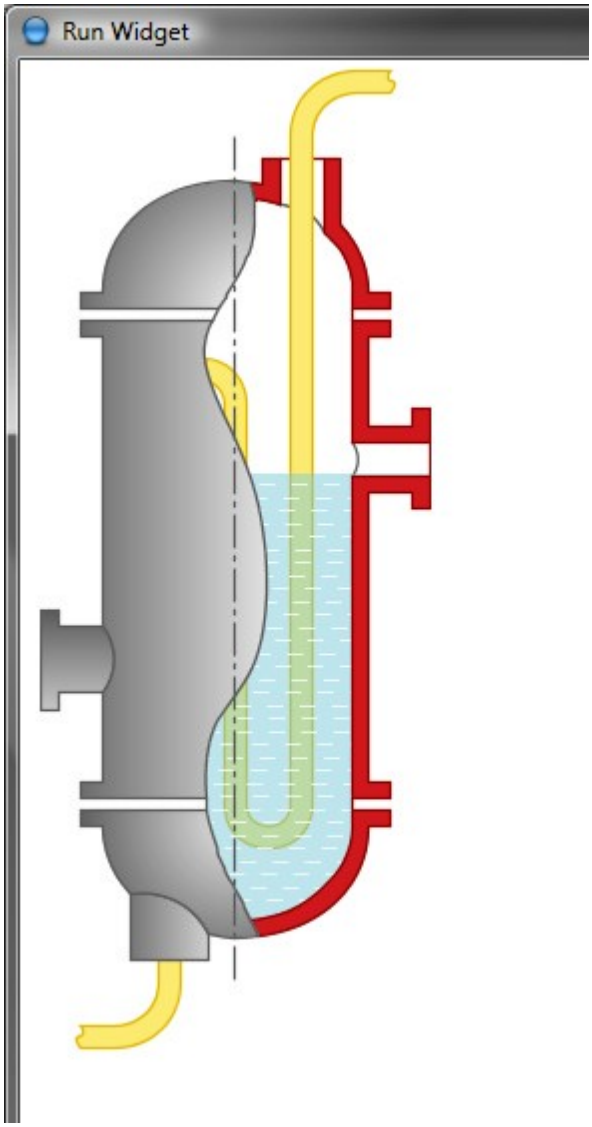
Общие события

Скрытие, Показ, Перемещение, Изменение размеров, Клик мыши, Нажатие кнопки мыши, Отпускание кнопки мыши, Вход мыши, Выход мыши, Перемещение мыши, Вращение колесика мыши, Печать клавиши, Нажатие клавиши, Отпускание клавиши, Получение фокуса, Потеря фокуса

13.4.9.22 Векторное изображение

Данный компонент отображает рисунок в масштабируемой векторной графике (SVG).

Компонент "Векторное изображение" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Рамка 1275, Курсор 1276, Всплывающее меню 1276

Пользовательские свойства

ФАЙЛ SVG (свойство по умолчанию 1274)

Отображаемый файл SVG.

Имя свойства: **drawing**

Тип свойства: **Data Block**

ВКЛЮЧИТЬ ВЗАИМОДЕЙСТВИЯ

Когда включены взаимодействия, то векторным рисунком в окне виджета можно управлять при помощи следующих сочетаний клавиш и мыши:

Сочетание	Значение
клавиша `Ctrl` + кнопка мыши `button 1`	Увеличить (приблизить область)
клавиша `Shift` + кнопка мыши `button 2`	Масштабировать
клавиша `Shift` + кнопка мыши `button 1`	Панорамирование

клавиша Ctrl + кнопка мыши button 2	Повернуть
клавиша Ctrl + клавиша Shift + кнопка мыши button 2	Сброс изменений

АНИМАЦИОННЫЙ

Свойство должно быть активировано для изображений, содержащих SVG анимацию.

Имя свойства: **animated**

Тип свойства: **Boolean**

КВАДРАТНОЕ ВРАЩЕНИЕ

Вращает изображение на определенные углы: 0°, 90°, 180°, 270°. Значением является количество дуг в 90°, по которым происходит вращение. Если включено **сохранить форматное соотношение** и изображение находится в компоновке Абсолютное позиционирование 954, вращение происходит на 90° или 270° от ширины/высоты рамки компонента.

Имя свойства: **quadrantRotation**

Тип свойства: **Integer**

Когда квадратное вращение отличается от 0°, отключается и игнорируется свойство **Вращение**.

ПЕРЕВЕРНУТЬ ПО ГОРИЗОНТАЛИ

Переворачивает рисунок по горизонтальной оси.

Имя свойства: **flipHorizontal**

Тип свойства: **Boolean**

ПЕРЕВЕРНУТЬ ПО ВЕРТИКАЛИ

Переворачивает рисунок по вертикальной оси.

Имя свойства: **flipVertical**

Тип свойства: **Boolean**

МАСШТАБ

Коэффициент масштабирования. Значение 1.0 используется для отключения масштабирования.

Если масштабированный и повернутый рисунок не входит по размеру в компонент, он будет обрезан.

Имя свойства: **scale**

Тип свойства: **Float**

ВРАЩЕНИЕ

Угол масштабирования, в градусах.

Если масштабированный и повернутый рисунок не подходит по размеру компоненту, он будет обрезан.

Имя свойства: **rotation**

Тип свойства: **Float**

СОХРАНИТЬ ФОРМАТНОЕ СООТНОШЕНИЕ

Включает/выключает сохранение форматного соотношения изображения. Если сохранение выключено, изображение растягивается, занимая все пространство компонента. Когда сохранение включено и изображение

находится в компоновке Абсолютное позиционирование 954, границы компонента пропорционально меняют свой размер.

Имя свойства: **preserveAspectRatio**

Тип свойства: **Boolean**

Когда сохранение форматного соотношения выключено, также выключаются свойства **Горизонтальное/Вертикальное выравнивание**.

Игнорирование форматного соотношения полезно, когда вам нужно непропорциональное масштабирование, например, при выстраивании трубопровода, состоящего из различных отрезков трубы.

ВЫРАВНИВАНИЕ ПО ГОРИЗОНТАЛИ

Определяет выравнивание рисунка по горизонтали, когда область компонента шире рисунка:

Слева, **По центру**, **Справа**.

Имя свойства: **horizontalAlignment**

Тип свойства: **String**

ВЫРАВНИВАНИЕ ПО ВЕРТИКАЛИ

Определяет выравнивание рисунка по вертикали, когда область компонента больше рисунка по высоте: **Сверху**, **По центру**, **Снизу**.

Имя свойства: **verticalAlignment**

Тип свойства: **String**

МАКСИМИЗИРОВАТЬ

Включает автоматическое увеличение изображения, чтобы отбросить его поля (пустое пространство вокруг изображения).

Когда **Максимизировать** проверено, включаются свойства **Процент отступа X/Y**. Настройка маленьких полей X/Y может использоваться для предотвращения обрезания краев, когда автоматически определяется размер изображения.

Имя свойства: **maximize**

Тип свойства: **Boolean**

ПРОЦЕНТ ОТСТУПА ПО X

Отступ от границы X в виде процентного отношения к ширине рисунка. Включено при выборе свойства **Растянуть**.

Имя свойства: **xMarginPercent**

Тип свойства: **Float**

ПРОЦЕНТ ОТСТУПА ПО Y

Отступ от границы Y в виде процентного отношения к высоте рисунка. Включено при выборе свойства **Растянуть**.

Имя свойства: **yMarginPercent**

Тип свойства: **Float**

ПОЛЬЗОВАТЕЛЬСКИЕ СВОЙСТВА SVG

SVG рисунки, входящие в состав некоторых пакетов дистрибутива AtomMind, усовершенствованы для поддержки дополнительных свойств. Компонент Векторное изображение извлекает эти свойства из SVG файлов и представляет их как "обычные" свойства компонента, позволяя редактировать их в Редакторе 423, управлять ими при помощи привязок виджета 1295 и др.

Пользовательские свойства видны в отдельной группе **Свойства SVG.** Типичными свойствами SVG являются:

- Цвет заливки (например, цвета элементов)
- Ширина и цвет штриха (например, штрихов контура)

- Угол вращения (например, стрелки индикатора)
- Триггер анимации (например, переключатели запуска/остановки вентилятора)
- Позиции элементов (например, уровни резервуаров, позиции амортизаторов)
- И многое другое (позиции кнопок и т.д.)

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

Пользовательские события

СОБЫТИЕ "ЩЕЛЧОК МЫШИ"

Это событие создается, когда кликнута кнопка мыши по элементу векторного изображения.

Имя события: **elementMouseClicked**

Поля события: Событие "щелчок мыши" имеет все поля стандартных событий мыши 1288 (исключая **Идентификатор**, **Модификаторы**, **Нажатие Alt Graph**, **Расширенные модификаторы** и **Триггер всплывающего меню**). Оно также определяет следующие поля:

Поле	Имя	Тип	Описание
Идентификатор	id	String	ID элемента векторного изображения.
Класс	class	String	Класс элемента векторного изображения.
Заголовок	title	String	Заголовок элемента векторного изображения.

СОБЫТИЕ "КУРСОР В ОБЛАСТИ ОБЪЕКТА"

Это событие создается, когда курсор мыши находится в области элемента векторного изображения.

Имя события: **elementMouseEntered**

Поля события: Событие "курсор мыши в области объекта" имеет все поля события "щелчок мыши" 999.

СОБЫТИЕ "КУРСОР ЗА ПРЕДЕЛАМИ ОБЪЕКТА"

Это событие создается, когда курсор мыши покидает область элемента векторного изображения.

Имя события: **elementMouseExited**

Поля события: Событие "курсор мыши покинул объект" имеет все поля события "щелчок мыши" 999.

13.4.9.22.1 Создание динамических векторных рисунков

Платформа AtomMind позволяет простым способом сделать неподвижное изображение подвижным и интерактивным. Вы можете добавлять пользовательские параметры к рисунку. Когда SVG рисунок с пользовательскими параметрами загружается в компонент Векторное изображение 995, параметры отображаются в качестве свойств компонента в отдельной вкладке в Редакторе свойств 431 Редактора виджетов 423. Их изменение может динамически изменять различные свойства рисунка: цвет, толщину линий, угол вращения, включение/отключение анимации и т.д.

Пространство XML имен

Все пользовательские теги имеют отдельную область имен: agg. Поэтому она должна быть объявлена в заголовке документа SVG для корректного формирования документа:

```
<svg version="1.1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/x
```

Теги XML

Струтура объявления пользовательских параметров следующая:

- Должен быть один корневой узел `<agg:params>`, включающий все другие объявления.
- Корневой элемент `<agg:params>` может включать множество элементов `<agg:param>`.
- Каждый элемент `<agg:param>` может включать несколько элементов `<agg:state>` с элементами `<agg:param>`.

AGG:PARAMS

Является корневым контейнером для всех пользовательских объявлений. В документе должен присутствовать только один тег `<agg:params>`.

AGG:PARAM

Имеет атрибут, который:

- содержит информацию о свойстве компонента,
- определяет правила применения значения свойства к элементам документа SVG.

Когда тег расположен в корневом узле `<agg:params>`, он создается в качестве свойства компонента: как применять пользовательские свойства и правила.

Когда тег расположен в корневом узле `<agg:state>`, применяются только правила, свойства компонента не создаются (см. AGG:STATES 1001).

Атрибуты используются для определения свойств компонента:

Атрибут	Тип	Описание
name	строка	Имя свойства компонента.
description	строка	Описание свойства в редакторе свойств.
type	строка	Текстовый код для типа свойства. Поддерживаются следующие типы: `I` (целое), `F` (плавающее), `S` (строка), `C` (цвет), `state`, `level`.
min	плавающее	Минимальное ограничение значения свойства для числовых типов или более низкое ограничение значения параметра для типа `level`.
max	плавающее	Максимальное ограничение значения свойства для числовых типов или более высокое ограничение значения параметра для типа `level`.

Тип **Level** подразумевает, что свойство компонента является целым числом (**Integer**) в пределе `[0..100]`. Каждое значение свойства в данном диапазоне пропорционально транслируется в диапазон `[min..max]` во время применения к документу. Числа `min` и `max` определяются соответствующими атрибутами параметра. Имейте в виду, что значение `min` может быть больше значения `max`.

Тип **State** необходим для применения заданных изменений к документу (см. AGG:STATES 1001).

Атрибуты, определяющие правила изменений SVG:

Атрибут	Тип	Описание
ids	строка	Список идентификаторов элементов документа SVG, разделенные запятой. Изменения параметров влияют на элементы с данными идентификаторами.
classes	строка	Список разделенных запятой классов CSS. Изменения параметров влияют на элементы с данными CSS.
attributes	строка	Список разделенных запятой атрибутов. Атрибуты элементов документа, значения которых перезаписываются новым значением при изменении параметра.
cssAttributes	строка	Разделенные запятыми CSS атрибуты. CSS атрибуты будут переписаны новыми значениями при изменении.
pattern	строка	Шаблон строки, содержащий "`{0}`". Используется для трансформации значения свойства в строку документа, заменяя `{0}` на значение.
forceRepaint	логическое	Данная кнопка-флажок контролирует политику изменения цвета рисунка SVG при изменении параметра. Если установлена на `false`, цвет изменяется частично на основе определения измененных областей (быстро). Если установлена на `true`, цвет рисунка меняется полностью при каждом изменении параметра (медленно). В определенных

		случаях обработчик рисунка SVG не может правильно управлять изменениями динамического документа (например, изменением масок обрезки), поэтому цвет рисунка должен быть изменен полностью. Принудительная перезаливка приведет к снижению динамической производительности SVG. Значение по умолчанию -- `false`.
animation *	"проиграть/остановить"	Включает/отключает анимацию элементов с определенными идентификаторами и классами.
value *	строка	Устанавливает значение для атрибута с определенными идентификаторами и классами.

Атрибуты **Animation** и **value** используются только тогда, когда `<agg:param>` определяет состояние. **Value** содержит постоянное строковое значение, которое применяется к документу при установлении необходимого состояния параметра компонента. **Animation** начинает/останавливает анимацию с данным `id` во время установки необходимого состояния.

AGG:STATE

Состояния используются для применения к документу некоторых определенных заранее изменений (состояний). Например, для включения/выключения вращения вентилятора. Параметр Состояние нескалярный и отображается в Редакторе свойств 431 в виде раскрывающегося списка с заданными опциями.

Он используется для установки множества значений для нескольких элементов DOM, когда применяется состояние. **Принцип определения состояния очень прост**: создайте `<agg:param>` с типом "state" и перечислите в нем состояния при помощи `<agg:state>`. Каждый тег `<agg:state>` может иметь множество тегов `<agg:param/>`, определяющих, как DOM должен измениться при применении состояния.

Он имеет следующие параметры:

Атрибут	Тип	Описание
name	String	Имя состояния.
description	String	Описание состояния в редакторе свойств.

Примеры

СТИЛЬ

Вы можете контролировать стиль элементов документа SVG, изменяя их атрибуты CSS. Чтобы изменить цвет различных частей рисунка, нужно использовать следующие объявления параметров:

```
<agg:params>
     <agg:param name="color" description="Color" type="C" cssAttributes="fill"
classes="mainColor"/>
</agg:params>
```

Когда изображение загружается в Редактор виджетов 423, оно будет содержать свойство компонента. Каждый раз, когда `Color` получает новое значение, обработчик изображения будет находить все элементы документа, у которых класс является `mainColor` и устанавливать новое значение цвета для атрибута CSS `fill`. Это приведет к изменению цвета частей изображения.

Чтобы сделать SVG более удобным для настройки, быстрой загрузки и обслуживания, вы можете использовать CSS (каскадные таблицы стилей) в том же файле. См. раздел **Стилевое оформление с CSS** в статье Спецификация W3C SVG.

ВРАЩЕНИЕ

Предположим, что документ SVG содержит элемент, представляющий узел с id `knob`. Применение вращения в точке (61.542, 61.792) повернет узел. Чтобы создать параметр для вращения на определенный угол, вы можете написать следующее:

```
<agg:params>
     <agg:parameter name="angle" description="Angle" type="I" ids="knob"
attributes="translation" pattern="rotate({0} 61.542 61.792)"/>
</agg:params>
```

Это описывает один параметр SVG `Angle` с типом **Integer**, который будет отображаться в качестве свойства компонента. Если параметр установлен на `45`, тогда атрибут элемента DOM `knob` будет установлен на `rotate(45 61.542 61.792)` и узел повернется на 45 градусов.

УРОВЕНЬ

Предположим, что уровень жидкости в документе SVG изображения реактора зависит от значения атрибута `y` тега `<clipPath id="clipPathId"...>:`, когда `y` равен `66.9`, реактор полон, когда `y` равен `660`, реактор пуст. Таким образом, для контроля уровня мы должны создать параметр с типом `level:`

```
<agg:params>

     <agg:param name="level" description="Level" type="level" attributes="y" min="660"
max="66" ids="clipPathId" forceRepaint="true"/>

</agg:params>
```

Данный параметр создает свойство компонента с типом **Integer** с ограничениями от `0` до `100`. Атрибуты `min="660"` и `max="66"` определяют диапазон атрибута `y.` Каждый раз, когда свойство компонента приобретает новое значение, оно пропорционально переводится из диапазона [0, 100] в [600, 66] и атрибут `y` элемента `clipPathId` устанавливается на данное переведенное значение. Если свойство `Level` установлено на `50`, то `y` будет установлено на `297`. Если свойство `Level` установлено на `100`, то `y` будет установлено на `66`.

Флажок `forceRepaint` используется для полного изменения цвета изображения при каждом изменении свойства `Level.` Поскольку динамические изменения маски обрезки не поддерживаются, вы можете получить неожиданные результаты, не изменив полностью цвет изображения.

СОСТОЯНИЯ

Для управления состояниями изображения, таких как "блок открыт" или "выключатель включен", вы можете использовать параметр state. Например, чтобы создать динамический тумблер с левой и правой позициями, вы можете создать документ SVG с двумя группами элементов. Одну для левой позиции выключателя, другую -- для правой. Итак, чтобы контролировать состояния тумблера, вам необходимо следить за видимостью групп. Определить параметр для переключения состояния можно следующим образом:

```
<agg:params>

     <agg:param name="state" description="State" type="state">

          <agg:state name="left" description="Left">

               <agg:param attributes="visibility" ids="switch-left"
value="visible"/>

               <agg:param attributes="visibility" ids="switch-right"
value="hidden"/>

          </agg:state>

          <agg:state name="right" description="Right">

               <agg:param attributes="visibility" ids="switch-left" value="hidden"/>

               <agg:param attributes="visibility" ids="switch-right"
value="visible"/>

          </agg:state>

     </agg:param>

</agg:params>
```

Это определяет свойство компонента `State` с двумя значениями: `Left` и `Right.` Если `State` установлено на `Left`, атрибут `visibility` элемента DOM `switch-left` получает значение `visible,` атрибут `visibility` элемента DOM `switch-right` получает "скрытое" значение. Если `State` установлено на `Right, switch-left` становится скрытым, а `switch-right` становится `visible.`

АНИМАЦИЯ

Допустим, у вас есть изображение вентилятора со следующей анимацией:

```
<animateTransform id="rotate" attributeName="transform" attributeType="XML"
type="rotate" from="0 45 45" to="360 45 45" begin="indefinite" dur="10s"
repeatCount="indefinite"/>
```

Для управления анимацией вы можете задать параметр состояния:

```
<agg:params>
      <agg:param name="state" description="State" type="state">
            <agg:state name="on" description="On">
                  <agg:param animation="play" ids="rotate"/>
            </agg:state>
            <agg:state name="off" description="Off">
                  <agg:param animation="stop" ids="rotate"/>
            </agg:state>
      </agg:param>
</agg:params>
```

Это определяет свойство компонента `State` с двумя значениями: `On` и `Off`. Когда оно получает значение `On`, запускается анимация `rotate`. Когда значение установлено на `Off`, анимация `rotate` прекращается.

Другие примеры использования пользовательских параметров SVG приведены в Библиотеке символов 1974.

13.4.9.23 Видеопроигрыватель

Компонент "Видеопроигрыватель" используется для отображения видеопотока или статических изображений JPEG, передаваемых по сетевой камере или видеосерверу.

Видеопроигрыватель выглядит следующим образом:

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активен 1275, Видимый 1275, Рамка 1275, Курсор 1276, Всплывающая подсказка 1276, Всплывающее меню 1276

Пользовательские свойства

URL (свойство по умолчанию 1274)

URL видеопотока или отображаемое JPEG изображение.

Имя свойства: **url**

Тип свойства: **String**

РЕЖИМ ВИДЕО

Формат видео: *FFmpeg*, *MJPEG Stream*, *JPEG Stills* или *LibVLC*. Режим *FFmpeg* активирует автоматическое определение конкретного типа потока. Поскольку AtomMind использует FFmpeg библиотеку для декодирования видеопотоков, актуальный список поддерживаемых форматов и видеокодеков можно найти на сайте FFmpeg: https://www.ffmpeg.org/general.html. Режим *LibVLC* также активирует автоматическое определение конкретного типа потока. Этот режим требует установки VLC программы.

Имя свойства: **videoMode**

Тип свойства: **Integer**

ЧАСТОТА ОБНОВЛЕНИЯ

Период обновления изображения (при использовании **режима видео** JPEG).

Имя свойства: **refresh**

Тип свойства: **Long**

МАСШТАБИРОВАТЬ ВИДЕО

Если включено, видеокадр уменьшается, чтобы соответствовать размеру компонента. Если выключено, видеокадр обрезается при условии, что он больше размера компонента.

Имя свойства: **scaleVideo**

Тип свойства: **Boolean**

ПОКАЗАТЬ СТАТУС

Определяет отображение частоты кадров и другие детали видео.

Имя свойства: **showStatus**

Тип свойства: **Boolean**

ИМЯ ПОЛЬЗОВАТЕЛЯ

Имя пользователя для аутентификации при входе в защищенный паролем URL.

Имя свойства: **username**

Тип свойства: **String**

ПАРОЛЬ

Пароль для аутентификации при входе в защищенный паролем URL.

Имя свойства: **password**

Тип свойства: **String**

АРГУМЕНТЫ VLC

Служит командной строкой для for the VLC медиаплеера. Все доступные команды и их характеристики см. по ссылке Командная строка VLS медиаплеера.

Имя свойства: **vlcArguments**

Тип свойства: **String**

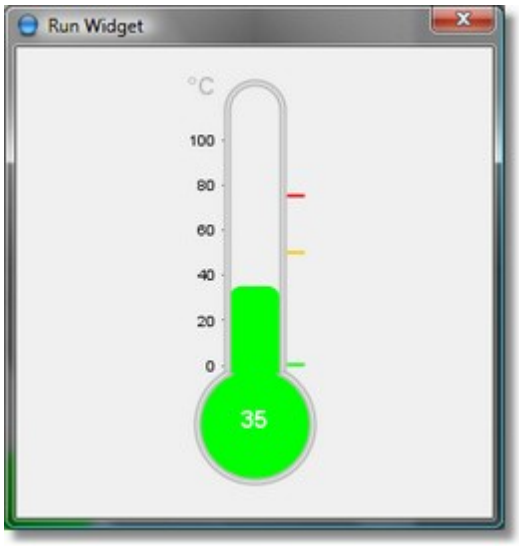
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.9.24 Измеритель

Данный компонент отображает измеритель, который лучше всего использовать для указания температуры, давления и подобных значений. Измеритель состоит из трех поддиапазонов, выделенных разным цветом.

Компонент "Измеритель" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активен 1275, Видимый 1275, Рамка 1275, Курсор 1276, Всплывающее меню 1276

Пользовательские свойства

ЗНАЧЕНИЕ (Свойство по умолчанию 1274)

Значение данных, отображаемое измерителем.

Имя свойства: **value**

Тип свойства: **Float**

ОБЛАСТЬ ПОСТРОЕНИЯ

Конфигурация области построения измерителя, т.е. "графическая часть" измерителя.

Поля области построения измерителя:

Поле	Тип	Описание
Нижнее ограничение	плавающее	Нижняя граница измерителя.
Верхнее ограничение	плавающее	Верхняя граница измерителя.
Положение оси	целое	Ось может быть расположена слева или справа от измерителя или же вобще не отображаться.
Прозрачность фона	плавающее	Значение альфа-канала прозрачности при заливке фона области построения.
Окраска фона	таблица данных	Цвет 1279 заливки фона области построения.
Радиус шарика	целое	Радиус шарика измерителя.
Радиус столбца	целое	Радиус основного столбца измерителя.
Использовать поддиапазоны для данных	логическое	Флажок, контролирующий, изменяется ли диапазон оси автоматически для отображения только того поддиапазона, который передает значение текущих данных.

Промежуток	целое	Промежуток между контурами для представления измерителя.
Отступы	таблица данных	Пустое пространство вокруг внешних границ области построения.
Окраска ртутного столбца	таблица данных	Цвет1279 ртути по умолчанию, когда значение вне всех поддиапазонов.
Окраска окантовки	таблица данных	Цвет1279 контура области построения.
Штрих окантовки	таблица данных	Штрих1282 контура области построения.
Видимость окантовки	логическое	Флажок, контролирующий видимость контура.
Заполнение	таблица данных	Заполнение (сверху, слева, снизу, справа) вокруг измерителя.
Окраска измерителя	таблица данных	Цвет1279 контура измерителя.
Штрих измерителя	таблица данных	Штрих1282 контура измерителя.
Единицы	целое	Тип единиц измерения. Может быть установлен на градусы по **Фаренгейту**, **Цельсию** или **Кельвину**, в противном случае на "**Не отображать единицы**".
Использовать окраску поддиапазонов	логическое	Флажок, контролирующий, приобретает ли ртуть цвет поддиапазонов измерителя.
Шрифт значения	таблица данных	Шрифт1278 для отображения метки значения, если она видимая.
Положение значения	целое	Положение, в котором будет отображаться текущее значение ("Нет", "Справа", "Слева", или "Шарик"). Имейте в виду, что **цвет значения** по умолчанию -- белый, поэтому если вы меняете положение, вам также нужно будет изменить цвет так, чтобы метка была видна.
Окраска значения	таблица данных	Цвет1279 метки значения, если она видимая.

Имя свойства: **plot**

Тип свойство: **Data Table**

ОСЬ

Конфигурация оси измерителя.

Поля оси измерителя:

Свойство	**Тип**	**Описание**
Метка	строка	Метка оси обычно описывает то, что измеряет ось.
Угол метки	плавающее	Угол вращения метки оси.
Шрифт метки	таблица данных	Шрифт1278 метки оси.
Отступы метки	таблица данных	Пустое пространство вокруг метки оси.
Окраска меток	таблица данных	Цвет1279 метки оси.
Шрифт меток делений	таблица данных	Шрифт меток делений.
Отступы меток делений	таблица данных	Пустое пространство вокруг меток делений.

Окраска меток делений	таблица данных	Цвет[1279] меток делений.
Видимость меток делений	логическое	Флажок, контролирующий видимость меток делений.
Внутренняя длина делений	плавающее	Расстояние, на которое деление выступает внутри области построения.
Внешняя длина делений	плавающее	Расстояние, на которое деление выступает за пределы области построения.
Окраска делений	таблица данных	Цвет[1279] делений.
Штрих делений	таблица данных	Штрих[1282] делений.
Видимость делений	логическое	Флажок, контролирующий видимость делений.
Видимость	логическое	Флажок, контролирующий видимость оси.

Имя свойства: **axis**

Тип свойства: **Data Table**

ПОДДИАПАЗОНЫ

Таблица поддиапазонов измерителя, используемая для отображения нормальных, предупреждающих и критических значений.

Поля поддиапазонов:

Поле	**Тип**	**Описание**
Поддиапазон	строка	Имя поддиапазона, доступно только для чтения.
Нижнее ограничение	плавающее	Нижнее ограничение поддиапазона.
Верхнее ограничение	плавающее	Верхнее ограничение поддиапазона.
Нижнее ограничение изображения	плавающее	Нижнее ограничение изображения. Ограничения изображения используются в том случае, когда диапазон оси измерителя автоматически настраивается для показа текущего поддиапазона (т.е. при включенном флажке Слежение за данными в поддиапазоне области построения измерителя).
Верхнее ограничение изображения	плавающее	Верхнее ограничение изображения.
Цвет	таблица данных	Цвет[1279] ртути для отображения значений в поддиапазоне.

Имя свойства: **subranges**

Тип свойства: **Data Table**

Общие события

Компонент убран с экрана[1285], Компонент появился на экране[1285], Компонент перемещается[1285], Изменен размер компонента[1285], Щелчок кнопки мыши[1286], Нажатие кнопки мыши[1286], Отпускание кнопки мыши[1286], Появление курсора мыши в компоненте[1286], Выход курсора мыши из компонента[1286], Перемещение мыши[1286], Прокрутка колеса мыши[1286], Введен символ[1287], Нажата клавиша[1287], Отпущена клавиша[1287], Фокус ввода получен[1287], Фокус ввода потерян[1287]

13.4.9.25 Компас

Данный компонент представляет собой компас с одной и более стрелками.

Компонент "Компас" выглядит следующим образом::

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Рамка 1275, Курсор 1276, Всплывающее меню 1276

Пользовательские свойства

ЗНАЧЕНИЯ ДАННЫХ (Свойство по умолчанию 1274)

Список значений, отображаемых компасом. Данное свойство является индексированным 1274.

Имя свойства: **datasets**

Тип свойства: **Data Table**

ОБЛАСТЬ ПОСТРОЕНИЯ

Конфигурация области построения компаса.

Свойства области построения компаса:

Свойство	Тип	Описание
Прозрачность фона	плавающее	Значение прозрачности, используемое при заливке фона области построения.
Окраска фона	таблица данных	Цвет 1279 заливки фона области построения.
Прорисовка фона	логическое	Флажок, определяющий, будет ли прорисован фон.
Отступы	таблица данных	Пустое пространство вокруг внешних границ области построения.
Размер оборота	плавающее	Длина полного оборота компаса в градусах. Значение по умолчанию - 360.0.
Окраска фона циферблата	таблица данных	Цвет 1279 заливки внутреннего пространства шкалы компаса.
Окраска внешней границыы циферблата	таблица данных	Цвет 1279 краев внешней границы циферблата.
Цвет розы ветров	таблица данных	Цвет 1279, заполняющий внешние границы циферблата компаса.

Имя свойства: **plot**

Тип свойства: **Data Table**

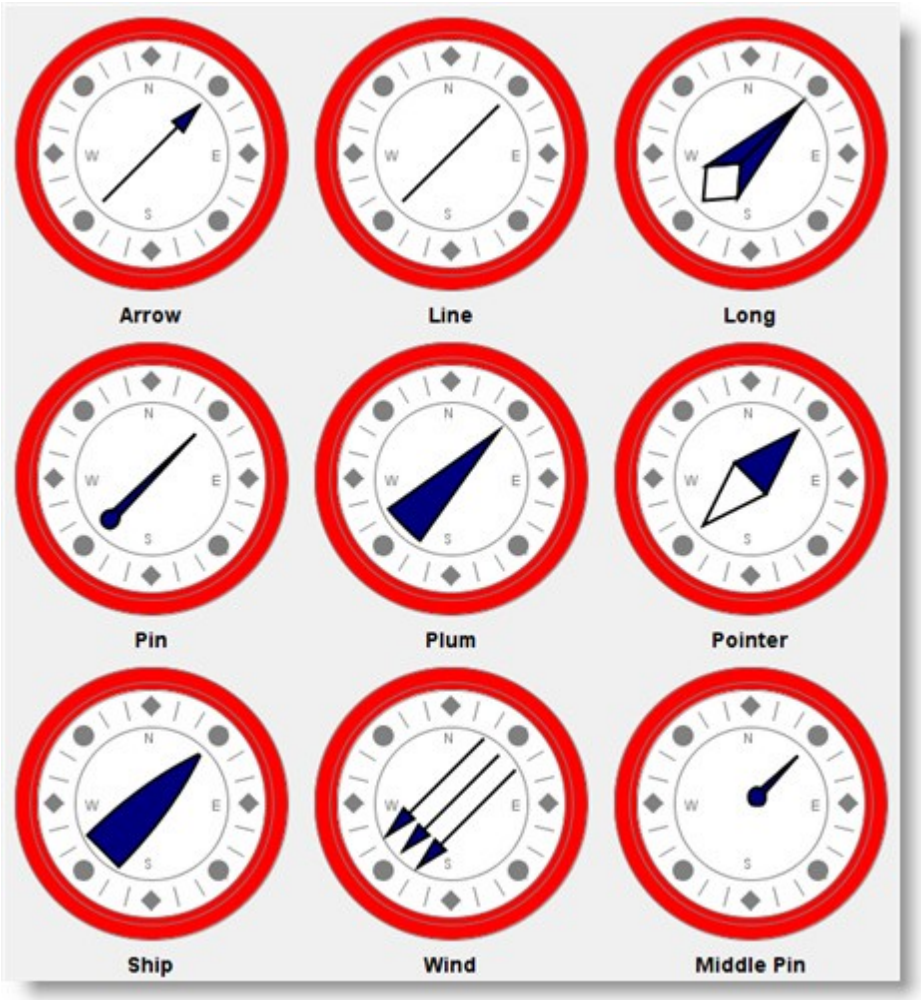
СТРЕЛКИ

Список стрелок компаса. Каждая стрелка указывает на одно значение данных 1008 соответствующего цвета.

Поля стрелок:

Поле	Тип	Описание
Тип стрелки	целое	Тип стрелки.
Окраска окантовки	таблица данных	Цвет 1279 контура стрелки.
Штрих окантовки	таблица данных	Штрих 1282 контура стрелки.
Окраска	таблица данных	Цвет 1279 заливки стрелки.

Типы стрелок:



Имя свойства: **needles**

Тип свойства: **Data Table**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1288, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.9.26 Карта

Данный компонент содержит слои 1013 (отображает и динамически обновляет геозоны, устройства, выражения таблицы узлов), отображает географическую карту, дорожную карту, карту местности или спутниковое изображение. В качестве источника геоинформационных данных и изображений используются карты Google, Bing Maps, OpenStreetMap, Yandex Maps и 2GIS.

Компонент Карт" выглядит следующим образом:

Масштабирование и прокрутка

Возможна **прокрутка** карты. Для этого надо, нажав клавишу мыши в определенном месте карты, перетащить ее. При этом карта будет перемещаться к центру.

Масштабирование (приближение/отдаление) осуществляется при помощи прокрутки колеса мыши. Другой способ приблизить карту - использовать счетчик масштабитрования и кнопки масштабирования, расположеные в верхнем левом углу компонента Карта.

Система координат

Данный компонент использует WGS84 систему координат. Любое устройство, предоставляющее данные о местоположении AtomMind, должно быть настроено на использование WGS84 координат для корректного отображения на карте.

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Рамка 1275, Всплывающее меню 1276

Пользовательские свойства

АВТООПРЕДЕЛЕНИЕ УРОВНЯ МАСШТАБИРОВАНИЯ

Дает команду отрисовщику карты настроить уровень масштабирования автоматически, чтобы центр карты, все метки и пути отображались в окне компонента без прокрутки.

Имя свойства: **detectZoom**

Тип свойства: **Логическое**

УРОВЕНЬ МАСШТАБИРОВАНИЯ

Масштаб определяет разрешение текущего вида. Возможны уровни масштабирования от 0 (самый низкий уровень, при котором на карте отображен весь мир) до 21+ (вплоть до отдельных зданий).

Каждое последующее повышение уровня масштабирования удваивает точность в горизонтальном и вертикальном направлениях.

Не все уровни масштабирования действуют на всех участках планеты. Они отличаются в зависимости от местоположения, т.к. данные некоторых точек земного шара являются более детальными, нежели данные других мест карты.

Если для данного местоположения нет доступных данных и уровня масштабирования, на карте появится пустое изображение.

Имя свойства: **zoom**

Тип свойства: **Целое**

ТИПЫ КАРТ

Выбирает поставщик географических данных для карт. Поддерживаются следующие источники карт:

- **Offline Map**. Использует офлайн карты, локально хранящиеся на компьютере, запускающем виджет с компонентом карты. Подробности смотрите в разделе **Папка хранения офлайн карт**.
- **Open Street Map - Mapnic**. Классическая, поддерживаемая обществом версия Open Street Map.
- **Open Street Map - MapQuest**. Другая версия Open Street Map.
- **Open Street Map - IP / Hostname**. Имя хоста/ IP адрес локального сервера OpenStreetMap.
- **Bing Maps - Roadmap**. Стандартное изображение карты дорог, использующее Bing Maps в качестве поставщика. Требует установки **Bing API Key**.
- **Bing Maps - Aerial**. Аэроснимки, использующие Bing Maps в качестве поставщика. Требует установки **Bing API Key**.
- **Bing Maps - Hybrid**. Комбинированное (карта дорог + аэроснимки) изображение, использующие Bing Maps в качестве поставщика. Требует установки **Bing API Key**.
- **Google Maps - Roadmap.** Стандартное изображение карты дорог, использующее Google Maps в качестве поставщика. Требует установки **Google Client ID** и **Google API Key**.
- **Google Maps - Satellite.** Спутниковое изображение, использующее Google Maps в качестве поставщика. Требует установки **Google Client ID** и **Google API Key**.
- **Google Maps - Terrain.** Изображение рельефа, использующее Google Maps в качестве поставщика. Требует установки **Google Client ID** и **Google API Key**.
- **Google Maps - Hybrid.** Комбинированное изображение (карта дорог + спутник), использующее Google Maps в качестве поставщика. Требует установки **Google Client ID** и **Google API Key**.
- **Yandex Maps - Roadmap.** Стандартное изображение карты дорог, использующее Yandex Maps в качестве поставщика.
- **Yandex Maps - Satellite.** Спутниковое изображение, использующее Yandex Maps в качестве поставщика.
- **Yandex Maps - Hybrid.** Комбинированное изображение (карта дорог + спутник), использующее Yandex Maps в качестве поставщика.
- **2GIS**. Очень подробное изображение определенных городов (в данный момент доступно на территории Российской Федерации).

Имя свойства: **mapType**

Тип свойства: **Строка**

ПАПКА ХРАНЕНИЯ ОФЛАЙН КАРТ

Определяет папку, хранящую данные офлайн карт. Доступна, когда **Источник карты** настроен на **Офлайн карты**. Этот путь папки может быть абсолютным или относительным в системе файлов компьютера, запускающего виджет, который содержит компонент Карта.

Значение по умолчанию - `tiles`, то есть данные офлайн карт хранятся в подпапке `/tiles` папки, в которой установлен AtomMind Client или Проигрыватель виджетов.

См. Использование офлайн карт 1022, чтобы узнать, как подготовить данные офлайн карт.

Имя свойства: **offlineMapStorage**

Тип свойства: **Строка**

IP / АДРЕС ЛОКАЛЬНОГО СЕРВЕРА

Определяет имя хоста/ IP адрес локального сервера Open Street Map. Доступно при настройке **Источник карты** на **Open Street Map - IP / Hostname.**

Имя свойства: **ipHostname**

Тип свойства: **Строка**

ИЗОБРАЖЕНИЕ ПРИ ОШИБКЕ

Изображение, появляющееся при ошибке извлечения фрагмента карты.

Имя свойства: **errorImage**

Тип свойства: **Блок данных**

ИДЕНТИФИКАТОР КЛИЕНТА GOOGLE

Используемый идентификатор Клиента, если Источник Карты настроен на Google.

Имя свойства: **googleClientId**

Тип свойства: **Строка**

КЛЮЧ API GOOGLE

Используемый ключ API для Google Maps.

Имя свойства: **googleApiKey**

Тип свойства: **Строка**

КЛЮЧ API BING

Используемый ключ API для Bing Maps.

Имя свойства: **bingApiKey**

Тип свойства: **Строка**

Эти серверы предоставляют ограниченную информацию пользователям без зарегистрированного ключа API и идентификатора клиента. Если вы хотите активно использовать карту, пожалуйста, определите данные настройки.

Свойства положения

ТИП

Данная настройка определяет центрирование карты:

- **Автоматически**. Карта центрируется для отображения всех устройств и путевых точек. Данную опцию можно сочетать с автоматически определяемым уровнем масштабирования 1010, чтобы убедиться, что карта отображает все доступные данные.
- **Геолокация**. Позволяет центрировать карту при помощи геокодирования, т.е. названия города, адреса улицы или почтового индекса.
- **Широта/Долгота**. Ручной ввод координат центра карты (широта и долгота).

Имя свойства: **locationType**

Тип свойства: **Целое**

ГЕОЛОКАЦИЯ

Данная опция включена, если типом местонахождения 1012 является Геолокация. Она определяет геокод центрирования карты, например `Berkeley,CA`.

Имя свойства: **geoLocation**

Тип свойства: **Строка**

Если вы хотит использовать сервис геолокации без ограничений, пожалуйста, введите **Ключ API Google** с **Идентификатором клиента Google**.

ШИРОТА

Данная опция включена, если типом местонахождения 1012 является Широта/Долгота. Она определяет широту фиксированного центра карты.

Имя свойства: **latitude**

Тип свойства: **Плавающее**

ДОЛГОТА

Данная опция включена, если типом местонахождения 1012 является Широта/Долгота. Она определяет долготу фиксированного центра карты.

Имя свойства: **longitude**

Тип свойства: **Плавающее**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1288, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

Пользовательские события

Клик мыши 1286 и Двойной клик мыши 1286

Имеют дополнительные поля:

Поле	Имя	Тип	Описание
широта	latitude	с плавающей запятой	Широта координат клика мыши.
долгота	longitude	с плавающей запятой	Долгота координат клика мыши.

13.4.9.26.1 Слой карты

Этот компонент отображает и динамически обновляет геозоны, устройства и выражения таблицы узлов.

Слой карты может показывать маркеры, определяющие текущее положение устройств, предоставляющих данные GPS и их треки. Он также может отображать связи между устройствами (топологию устройства).

Источником данных для слоя может быть:

- **Маска контекста** - используется для отображения устройств. Все компоненты и выражения топологии 1300 активны. К каждому устройству будет применено выражение.
- **Маска геозоны** - используется для отображения геозон 1438. Долгота, широта и выражения топологии 1300 неактивны. Для каждого ряда в геозоне будет применено выражение.
- **Пользовательские узлы -** используются для отображения пользовательских данных, сгенерированных **Выражением таблицы узлов.** Все выражения компонентов активны, выражения топологии 1300 неактивны. Для каждого ряда в итоговой таблице будет применено выражение.
- **Трек** - используется для отображения трека, т.е. истории передвижения устройства/транспортного средства. Узлы будут добавляться к треку в том же порядке, в котором они появляются в итоговой таблице **Выражения таблицы узлов.** Все выражения компонентов активны, выражения топологии 1300 неактивны.

Основные свойства

ОБЩИЕ СВОЙСТВА

Слой карты имеет следующие основные свойства, общие для всех виджетов:

Видимый 1275, Рамка 1275, Шрифт 1275, Точки прикрепления 1284.

ВИЗУАЛИЗАЦИЯ ТОПОЛОГИИ

Слой карты наследует следующие базовые свойства Визуализации топологии 1300:

Всплывающее меню вершин 1300, Всплывающее меню связей 1300.

ГЛУБИНА

Глубина слоя, которая также называется *z-order*. Слои с меньшей глубиной отображаются над слоями с большей глубиной.

Имя свойства: **depth**

Тип свойства: **Integer**

свойства содержимого

ТИП

Выбирает тип источника данных. Доступные следующие типы источников данных:

- **Маска контекста** (Маска контекстов устройства)
- **Маска геозоны** (Маска контекстов геозоны)
- **Выражение таблицы узлов** (Выражение, которое должно возвращать таблицу данных)

Имя свойства: **type**

Тип свойства: **String**

МАСКА КОНТЕКСТА

Этот тип источника позволяет выбирать маску контекстов устройств в качестве источника данных слоя.

Маска по умолчанию `users.*.devices.*` совместима со всеми устройствами в системе.

Имя свойства: **contextMask**

Тип свойства: **String**

Формат полей: **Устройство** 1494

МАСКА ГЕОЗОНЫ

Этот тип источника позволяет выбирать маску контекстов геозон в качестве источника данных слоя.

Маска по умолчанию `geofences.*` совместима со всеми устройствами в системе.

Имя свойства: **geofenceMask**

Тип свойства: **String**

Формат полей: **Геозоны** 1438

ВЫРАЖЕНИЕ ТАБЛИЦЫ УЗЛОВ

Этот тип источника позволяет использовать выражение (должно возвращать Таблицу данных) в качестве источника данных слоя.

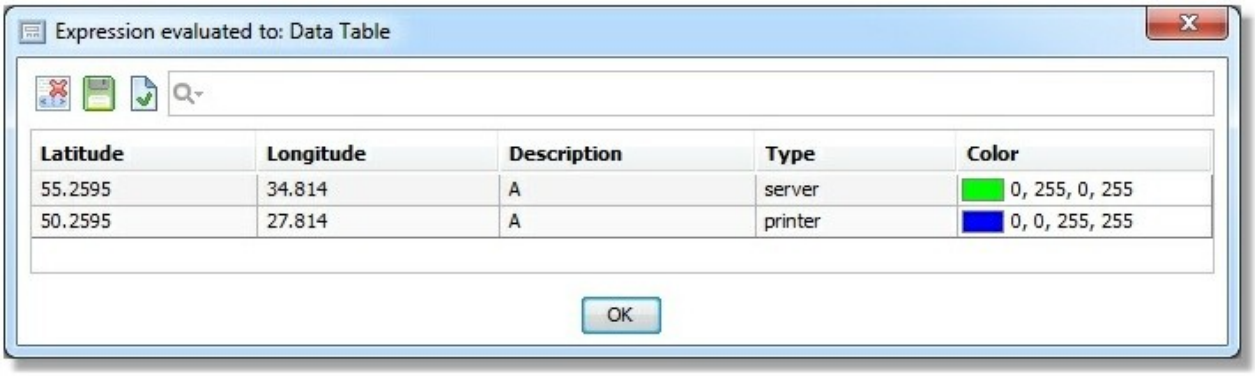
Контекст по умолчанию 119	Отсутствует.
Таблица данных по умолчанию 120	Отсутствует.
Ряд по умолчанию 119	Отсутствует.

Переменные среды	Только стандартные переменные.

Имя свойства: **nodeTableExpression**

Тип свойства: **String**

Например, выражение `"table("<<latitude><E>><<longitude><E>><<description><S>><<type><S>>,<<color><C>>", 55.2595, 34.814, "A","server",color(0,255,0), 50.2595, 27.814, "A","printer",color(0,0,255))"` создает таблицу данных с двумя записями.



ДЛИНА ОТОБРАЖАЕМОГО ТРЕКА (ПУТЕВЫХ ТОЧЕК)

Количество путевых точек Device для отображения на карте.

Хранение истории настроек Device должно быть активно для настройки Расположение (`location`) Device, чтобы хранить историческую информацию трека и отображать ее на карте.

Имя свойства: **trackLength**

Тип свойства: **Integer**

свойства выражений

TOPOLOGY VISUALIZATION

Слой карты наследует следующие свойства выражений Визуализации топологии:

Выражение имени предпочитаемого действия, Выражение цвета, Выражения для получения изображения, Выражение азимута, Выражение размера маркера.

ТИП ЦВЕТА

Тип цвета для рисования объектов слоя.

Имя свойства: **contentColorType**

Тип свойства: **String**

Допустимые значения:

- **Статические** (Маска контекстов устройств)
- **Динамические** (Маска контекстов геозон)

ЦВЕТ

Цвет объектов слоя.

Имя свойства: **contentColor**

Тип свойства: **Color**

ВЫРАЖЕНИЕ ПОДСКАЗКИ

Выражение, которое возвращает подсказки объектов слоя.

Контекст по умолчанию 119	Контекст слоя (для типа Геозона), контекст устройства (для типа Контекст), отсутствует (для типа Выражение таблицы узлов).
Таблица данных по умолчанию 120	Итоговая таблица Выражения таблицы узлов (для типа Выражение таблицы узлов), отсутствует (для других типов).
Ряд по умолчанию 119	Указывает на обрабатываемый в данный момент ряд таблицы данных по умолчанию, отсутствует (для типа Контекст).
Переменные среды 123	Только стандартные 123 переменные.

Имя свойства: **tooltipExpression**

Тип свойства: **String**

ВЫРАЖЕНИЕ ШИРОТЫ

Выражение, которое возвращает широту объекта слоя (не активно для типа Геозона).

Контекст по умолчанию 119	Контекст слоя (для типа Геозона), контекст устройства (для типа Контекст), отсутствует (для типа Выражение таблицы узлов).
Таблица данных по умолчанию 120	Итоговая таблица Выражения таблицы узлов (для типа Выражение таблицы узлов), отсутствует (для других типов).
Ряд по умолчанию 119	Указывает на обрабатываемый в данный момент ряд таблицы данных по умолчанию, отсутствует (для типа Контекст).
Переменные среды 123	Только стандартные 123 переменные.

Имя свойства: **latitudeExpression**

Тип свойства: **String**

ВЫРАЖЕНИЕ ДОЛГОТЫ

Выражение, которое возвращает долготу объекта слоя (не активно для типа Геозона).

Контекст по умолчанию 119	Контекст слоя (для типа Геозона), контекст устройства (для типа Контекст), отсутствует (для типа Выражение таблицы узлов).
Таблица данных по умолчанию 120	Итоговая таблица Выражения таблицы узлов (для типа Выражение таблицы узлов), отсутствует (для других типов).
Ряд по умолчанию 119	Указывает на обрабатываемый в данный момент ряд таблицы данных по умолчанию, отсутствует (для типа Контекст).
Переменные среды 123	Только стандартные 123 переменные.

Имя свойства: **longitudeExpression**

Тип свойства: **String**

свойства привязок

Слой карты использует свойство Привязки 1275, общее для всех виджетов.

свойства топологии

TOPOLOGY VISUALIZATION

Слой карты наследует следующие свойства Визуализации топологии 1300:

Провайдер 1300, Узлы являются контекстами 1300, Показывать несвязанные узлы 1300, Выражение топологии 1300, Выражение узлов 1300, Выражение связей 1300, Выражение идентификаторов связей 1300, Выражение идентификаторов узлов 1300, Выражение источника 1300, Выражение назначения 1300, Выражение интерфейса 1300, Выражение направленности 1300, Выражение ширины 1300, Выражение описания связи 1300, Показывать описания связи 1300, Выражение цвета связи 1300, Показывать граничащие узлы 1300, Выражение подсказки узла 1300, Выражение подсказки связи 1300.

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

Пользовательские события

НАЖАТИЕ НА УЗЕЛ

Это событие запускается при клике на узел карты.

Имя события: **nodeClicked**

В случае клика по узлу Устройство/Контекст

Поле	Имя	Тип	Описание
Node	node	String	Путь выбранного узла/контекста.
Latitude	latitude	Float	Широта контекста сервера, на который наведен курсор мыши.
Longitude	longitude	Float	Долгота контекста сервера, на который наведен курсор мыши.
Type	type	String	Тип выбранного узла/контекста (принтер ,сервер, и т.д.).
Color	color	Color	Цвет выбранного узла/контекста.
Labels	labels	Data Table	См. Выражение описания узла 1300
Image	image	Data	Изображение выбранного узла/контекста.
Tooltip	tooltip	String	Подсказка выбранного узла/контекста.
Device Name	deviceName	String	Имя выбранного узла/контекста.
Device Description	deviceDescription	String	Описание выбранного узла/контекста.
Azimuth	azimuth	Double	Угловое измерение в сферической системе координат выбранного узла/контекста.
Ratio	ratio	Double	Пропорции размера изображения выбранного узла/контекста.

Поля событий в случае клика по узлу, который является точкой Геозоны:

Поле	Имя	Тип	Описание
Description	description	String	Описание выбранной точки.
Latitude	latitude	Float	Широта точки, на которую наведен курсор мыши.
Longitude	longitude	Float	Долгота точки, на которую наведен курсор мыши.
Type	type	String	Тип выбранной точки.
Color	color	Color	Цвет выбранной точки.

В случае клика по узлу, который является точкой Выражения таблицы узлов, поле события содержит запись из итоговой таблицы Выражения таблицы узлов. См. пример 1015.

ДВОЙНОЕ НАЖАТИЕ НА УЗЕЛ

Это событие запускается, при двойном клике на узел карты.

Имя события: **nodeDoubleClick**

В случае клика по узлу Устройство/Контекст

Поле	Имя	Тип	Описание
Node	node	String	Путь выбранного узла/контекста.
Latitude	latitude	Float	Широта контекста сервера, на который наведен курсор мыши.
Longitude	longitude	Float	Долгота контекста сервера, на который наведен курсор мыши.
Type	type	String	Тип выбранного узла/контекста (принтер ,сервер, и т.д.).
Color	color	Color	Цвет выбранного узла/контекста.
Labels	labels	Data Table	См. Выражение описания узла 1300
Image	image	Data	Изображение выбранного узла/контекста..
Tooltip	tooltip	String	Подсказка выбранного узла/контекста.
Device Name	deviceName	String	Имя выбранного узла/контекста.
Device Description	deviceDescription	String	Описание выбранного узла/контекста.
Azimuth	azimuth	Double	Угловое измерение в сферической системе координат выбранного узла/контекста.
Ratio	ratio	Double	Пропорции размера изображения выбранного узла/контекста

Поля событий в случае клика по узлу, который является точкой Геозоны:

Поле	Имя	Тип	Описание
Описание	description	String	Описание выбранной точки.
Широта	latitude	Float	Широта точки, на которую наведен курсор мыши.
Долгота	longitude	Float	Долгота точки, на которую наведен курсор мыши.
Тип	type	String	Тип выбранной точки.
Цвет	color	Color	Цвет выбранной точки.

В случае клика по узлу, который является точкой Выражения таблицы узлов, поле события содержит запись из итоговой таблицы Выражения таблицы узлов. См. пример 1015.

Наведение курсора мыши на узел

Это событие запускается, когда указатель мыши наведен на узел, отображенный на карте.

Имя события: **nodeEnter**

В случае клика по узлу Устройство/Контекст:

Поле	Имя	Тип	Описание
Node	node	String	Путь выбранного узла/контекста.
Latitude	latitude	Float	Широта контекста сервера, на который наведен курсор мыши.
Longitude	longitude	Float	Долгота контекста сервера, на который наведен курсор мыши.
Type	type	String	Тип выбранного узла/контекста (принтер ,сервер, и т.д.).
Color	color	Color	Цвет выбранного узла/контекста.

Labels	labels	Data Table	См. Выражение описания узла 1300
Image	image	Data	Изображение выбранного узла/контекста..
Tooltip	tooltip	String	Подсказка выбранного узла/контекста.
Device Name	deviceName	String	Имя выбранного узла/контекста.
Device Description	deviceDescription	String	Описание выбранного узла/контекста.
Azimuth	azimuth	Double	Угловое измерение в сферической системе координат выбранного узла/контекста.
Ratio	ratio	Double	Пропорции размера изображения выбранного узла/контекста

Поля событий в случае клика по узлу, который является точкой Геозоны:

Поле	**Имя**	**Тип**	**Описание**
Description	description	String	Описание выбранной точки.
Latitude	latitude	Float	Широта точки, на которую наведен курсор мыши.
Longitude	longitude	Float	Долгота точки, на которую наведен курсор мыши.
Type	type	String	Тип выбранной точки.
Color	color	Color	Цвет выбранной точки.

В случае клика по узлу, который является точкой Выражения таблицы узлов, поле события содержит запись из итоговой таблицы Выражения таблицы узлов. См. пример 1015.

Вывод курсора мыши с узла

Это событие запускается, когда указатель мыши отведен с узла, отображенного на карте.

Имя события: **nodeLeave**

В случае клика по узлу Устройство/Контекст:

Поле	**Имя**	**Тип**	**Описание**
Node	node	String	Путь выбранного узла/контекста.
Latitude	latitude	Float	Широта контекста сервера, на который наведен курсор мыши.
Longitude	longitude	Float	Долгота контекста сервера, на который наведен курсор мыши.
Type	type	String	Тип выбранного узла/контекста (принтер ,сервер, и т.д.).
Color	color	Color	Цвет выбранного узла/контекста.
Labels	labels	Data Table	См. Выражение описания узла 1300
Image	image	Data	Изображение выбранного узла/контекста..
Tooltip	tooltip	String	Подсказка выбранного узла/контекста.
Device Name	deviceName	String	Имя выбранного узла/контекста.
Device Description	deviceDescription	String	Описание выбранного узла/контекста.
Azimuth	azimuth	Double	Угловое измерение в сферической системе координат выбранного узла/контекста.
Ratio	ratio	Double	Пропорции размера изображения выбранного узла/контекста

Поля событий в случае клика по узлу, который является точкой Геозоны:

Поле	Имя	Тип	Описание
Description	description	String	Описание выбранной точки.
Latitude	latitude	Float	Широта точки, на которую наведен курсор мыши.
Longitude	longitude	Float	Долгота точки, на которую наведен курсор мыши.
Type	type	String	Тип выбранной точки.
Color	color	Color	Цвет выбранной точки.

В случае клика по узлу, который является точкой Выражения таблицы узлов, поле события содержит запись из итоговой таблицы Выражения таблицы узлов. См. пример 1015.

Нажатие на полигон

Это событие запускается при нажатии на полигон.

Имя события: **polygonClick**

Поле	Имя	Тип	Описание
Context	context	String	Путь выбранного полигона.
Name	name	String	Имя выбранного полигона.
Type	type	String	Тип выбранного полигона.

ДВОЙНОЕ НАЖАТИЕ НА ПОЛИГОН

Это событие запускается при двойном нажатии на полигон.

Имя события: **polygonDoubleClick**

Поле	Имя	Тип	Описание
Context	context	String	Путь выбранного полигона.
Name	name	String	Имя выбранного полигона.
Type	type	String	Тип выбранного полигона.

НАВЕДЕНИЕ КУРСОРА МЫШИ НА ПОЛИГОН

Это событие запускается, когда указатель мыши наведен на полигон, отображенный на карте.

Имя события: **polygonEnter**

Поле	Имя	Тип	Описание
Context	context	String	Путь выбранного полигона.
Name	name	String	Имя выбранного полигона.
Type	type	String	Тип выбранного полигона.

ВЫВОД КУРСОРА МЫШИ С ПОЛИГОНА

Это событие запускается, когда указатель мыши отведен с полигона, отображенного на карте.

Имя события: **polygonLeave**

Поле	Имя	Тип	Описание
Context	context	String	Путь выбранного полигона.

Name	name	String	Имя выбранного полигона.
Type	type	String	Тип выбранного полигона.

НАВЕДЕНИЕ КУРСОРА МЫШИ НА СОЕДИНЕНИЕ

Это событие запускается, когда указатель мыши наведен на соединение между двумя узлами, отображенными на карте.

Имя события: **edgeEnter**

Поле	Имя	Тип	Описание
Source	source	String	Имя контекста источника соединения.
Destination	destination	String	Имя контекста ответчика соединения.
Source Interface	sourceInterface	String	Имя интерфейса источника соединения.
Destination Interface	destinationInterface	String	Имя интерфейса ответчика соединения.

ВЫВОД КУРСОРА МЫШИ С СОЕДИНЕНИЯ

Это событие запускается, когда указатель мыши отведен с соединения между двумя узлами, отображенными на карте.

Имя события: **edgeLeave**

Поле	Имя	Тип	Описание
Source	source	String	Имя контекста источника соединения.
Destination	destination	String	Имя контекста ответчика соединения.
Source Interface	sourceInterface	String	Имя интерфейса источника соединения.
Destination Interface	destinationInterface	String	Имя интерфейса ответчика соединения.

НАЖАТИЕ НА СОЕДИНЕНИЕ

Это событие запускается при щелчке по соединению между двумя узлами.

Имя события: **edgeClick**

Поле	Имя	Тип	Описание
Source	source	String	Имя контекста источника соединения.
Destination	destination	String	Имя контекста ответчика соединения.
Source Interface	sourceInterface	String	Имя интерфейса источника соединения.
Destination Interface	destinationInterface	String	Имя интерфейса ответчика соединения.

ПОКАЗ КОНТЕКСТНОГО МЕНЮ УЗЛА

Событие возникает при вызове контекстного меню кликом правой кнопки мыши на узел.

Имя события: **nodeShowPopup**

Поля события:

Поле	Имя	Тип	Описание
Node	node	String	Путь узла/контекста, для которого отображается контекстное меню.

ПОКАЗ КОНТЕКСТНОГО МЕНЮ СВЯЗИ

Событие возникает при вызове контекстного меню связи кликом правой кнопки мыши на связь между двумя узлами

Имя события: **edgeShowPopup**

Поля события:

Поле	Имя	Тип	Описание
Source	source	String	Узел/контекст источника связи.
Destination	destination	String	Узел/контекст адресата связи.
Source Interface	sourceInterfaceId	String	Интерфейс источника связи. Необходимо когда между двумя узлами существует множество связей.
Destination Interface	destinationInterfaceId	String	Интерфейс адресата связи. Необходимо когда между двумя узлами существует множество связей.

ПОКАЗ КОНТЕКСТНОГО МЕНЮ ГРАФА

Событие возникает при вызове контекстного меню графа кликом правой кнопки мыши на его холст.

Имя события: **graphShowPopup**

Событие не имеет полей.

13.4.9.26.2 Использование офлайн карт

Если свойство **Источник карты** компонента виджета Карта установлено на **Офлайн карта**, компонент пытается загрузить карты из локальной папки компьютера, запускающего виджет. Положение карт определено свойством **Папка хранения офлайн карт**.

Перед использованием офлайн карт, необходимо создать их с помощью Maperitive или MapTiler. Ниже приведена инструкция, как пользоваться офлайн картами через бесплатное приложение Maperitive.

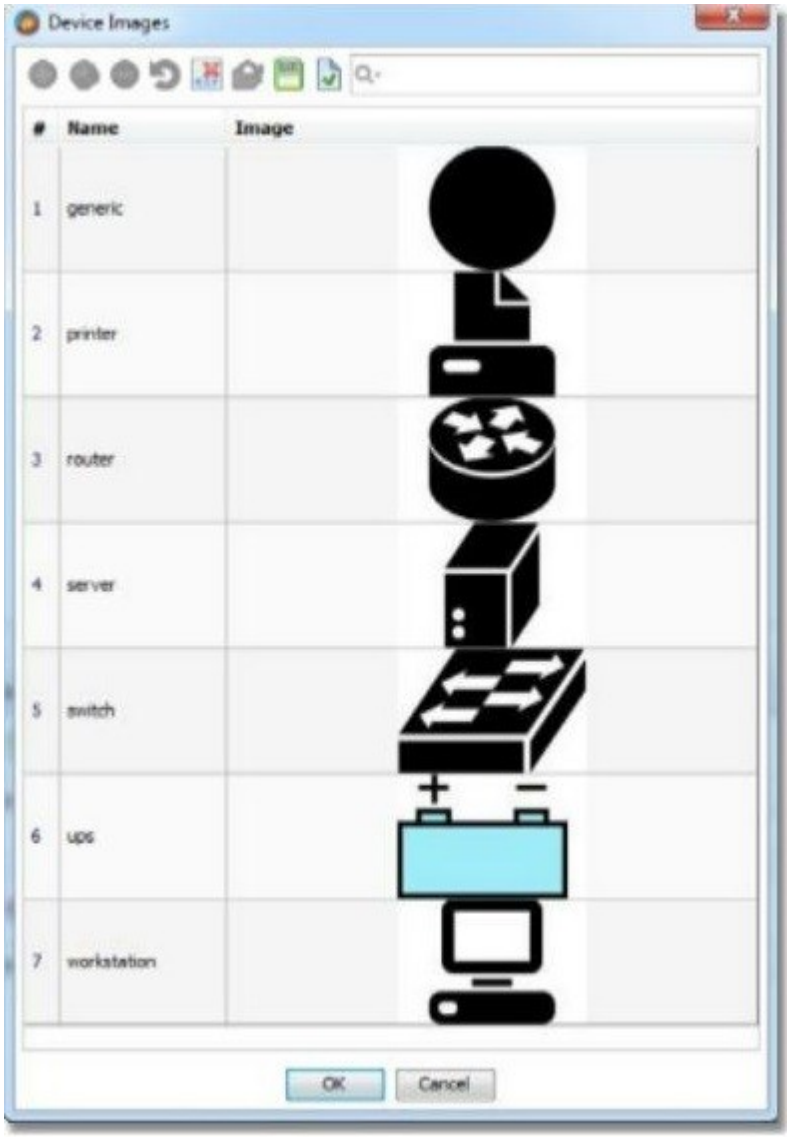
- Скачайте Maperitive с сайта http://maperitive.net, распакуйте и запустите.
- При запуске Maperitive, он покажет OSM веб-карту. Стили карт Maperitive (или правила рендеринга) работают на векторных данных, таких как OSM или GPX файлы, поэтому сначала необходимо загрузить векторный файл в Maperitive и посмотреть, какие правила уже действуют. Можно использовать команду меню **File > Open Map Sources**, чтобы открыть один или более источников карт (или просто перетащить файлы в Maperitive). Можно также использовать команду меню **Map > Edit Rendering Rules**, чтобы редактировать правила рендеринга карты в текстовом редакторе. Более подробно см. Документацию по Maperitive.
- Сохраните спроектированную карту, используя команду меню **Tools > Generate Tiles**.
- После создания вашей карты, она окажется в папке **Maperitive root directory> Tiles**. Дополнительно, можно указать эту же папку в качестве **Папка-хранилище офлайновой карты** вашего компонента Карта и запустить офлайн карту.

13.4.9.26.3 Изображения устройства

Данная модель содержит изображения, которые могут использоваться в выражении изображения в Слое карты 1015 или Визуализации топологии 1300.

Поле	Тип	Описание
Name	Строка	Имя изображения.
Image	Блок данных	Содержит изображение.

По умолчанию, модель содержит следующие изображения:



13.4.9.27 Устройство

Данный компонент представляет изображение какого-либо аппаратного устройства и показывает динамически созданные сообщения о статусе устройства рядом или над его изображением.

Компонент "Устройство" выглядит следующим образом:

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Фон 1275, Непрозрачный 1275, Рамка 1275, Всплывающая подсказка 1276, Фокусируемый 1276, Всплывающее меню 1276

Пользовательские свойства

ИСТОЧНИК

Путь контекста 41, откуда можно получить данные. В большинстве случаев этот путь указывает на контекст устройства 1494.

Имя свойства: **source**

Тип свойства: **String**

ИЗОБРАЖЕНИЕ

Изображение устройства (поддерживаются растровые и векторные изображения).

Имя свойства: **image**

Тип свойства: **Data Block**

МЕТКИ

Конфигурация динамических меток, указывающих на текущий статус устройства.

Свойства метки:

<table>
<tr><th>Поле</th><th>Тип</th><th>Описание</th></tr>
<tr><td>Выражение</td><td>строка</td><td>Выражение 112 текста метки. Это выражение рассчитывается каждый Период. Его результат конвертируется в строку и отображается на метке.

<table>
<tr><td colspan="2">Среда вычисления 114 выражения метки:</td></tr>
<tr><td>Конекст по умолчанию 119</td><td>Контекст, определенный свойством Источник этого компонента устройства. Если Источник не определен, то это контекст по умолчанию 946 данного действующего виджета.</td></tr>
<tr><td>Таблица данных по умолчанию 120</td><td>Таблица параметров виджета 949.</td></tr>
<tr><td>Строка по умолчанию 119</td><td>0</td></tr>
<tr><td>Переменные среды 123</td><td>Только стандартные 123 переменные.</td></tr>
</table>
</td></tr>
<tr><td>Выравнивание по вертикали</td><td>целое</td><td>Вертикальное выравнивание метки.</td></tr>
<tr><td>Выравнивание по горизонтали</td><td>целое</td><td>Горизонтальное выравнивание метки.</td></tr>
<tr><td>Шрифт</td><td>таблица данных</td><td>Шрифт 1278 метки.</td></tr>
<tr><td>Основной цвет</td><td>цвет</td><td>Цвет метки.</td></tr>
<tr><td>Период</td><td>длинное</td><td>Период обновления текста метки, т.е. период переоценки Выражения.</td></tr>
</table>

Имя свойства: **labels**

Тип свойства: **Data Table**

СТАТУС

Числовое значение компонента *Статус*. Статус используется для определения цвета изображения устройства согласно **Таблице статусов**.

Если значение статуса не является NULL, устройство отображается таким же цветом, что и соответствующая строка в таблице статусов.

Если значение статуса является NULL, цвет устройства не меняется.

Имя свойства: **status**

Тип свойства: **Integer**

ТАБЛИЦА СТАТУСОВ

Таблица, содержащая значение цвета для каждого **статуса**, используемого данным компонентом устройства.

Поля таблицы статусов:

Поле	Тип	Описание
Статус	целое	Значение свойства **Статус**.
Цвет	цвет	Цвет изображения устройства соответственно его статусу.

Имя свойства: **statusTable**

Тип свойства: **Data Table**

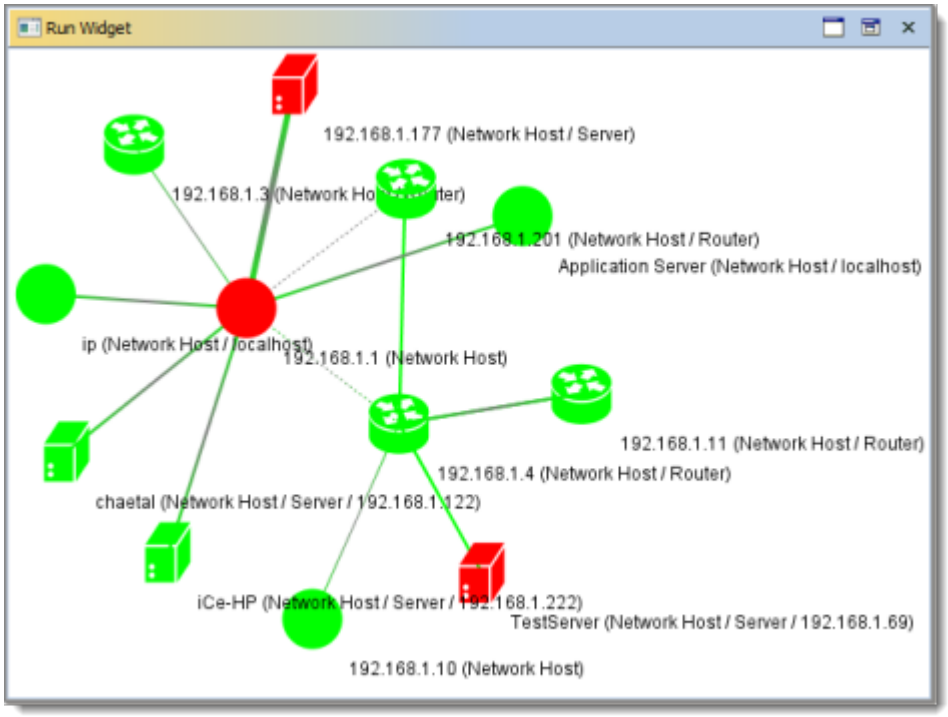
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.9.28 Граф

Этот компонент отображает и динамически обновляет граф любого типа, например, граф топологии сети.

Так выглядит топология сети, представленная в виде графа:



Элементы графа

Графы, представленные компонентом Граф, состоят из *узлов* и *ребер*. Узлы - это вершины графа, ребра - связи между узлами. Например, в графе топологии сети хосты представлены узлами, а связи между ними - ребрами.

Два узла могут иметь более одной связи между собой. В этом случае, ребро берет начало в каком-либо *интерфейсе* одного узла и соединяется с другим интерфейсом второго узла.

Фон графа называется *холстом*.

Каждый узел графа имеет *опорный объект*. Это значит, что каждый отображенный узел представляет какой-либо контекст. Получить доступ к опорному объекту можно через действия с графом, описанные ниже. Например, узлы топологии сети могут представлять сетевые устройства.

ЦВЕТОВОЕ КОДИРОВАНИЕ УЗЛОВ

Цвет узлов графа указывает на состояние соответствующих объектов.

Пример: на графе топологии сети зеленый узел обозначает, что устройство сети подключено и не имеет проблем. Красные узлы - это неподключенные устройства или устройства, имеющие проблемы.

МЕТКИ УЗЛОВ

Каждый узел графа имеет метку, описывающую соответствующий объект. Метки узлов задаются свойством Выражение описания узда 1300.

Пример: на топологической карте сети метки узлов показывают описания контекстов устройств.

ШИРИНА И ЦВЕТОВОЕ КОДИРОВАНИЕ РЕБЕР (СВЯЗЕЙ)

Ширина ребер диаграммы может разниться в зависимости от типа опорного объекта.

Пример: на топологической карте сети более узкие ребра соответствуют низкой скорости соединения (например, соединение в 100 Мбит/с), в то время как широкие ребра представляют соединения высокой скорости (например, гигабит Ethernet).

Цвет ребра обозначает статус соединения узел к узлу.

Пример: на топологической карте сети каждое ребро может иметь различную окраску на противоположных сторонах. Зеленая сторона ребра указывает на то, что интерфейс сети, соответствующий этому соединению, находится в режиме онлайн, а красная сторона указывает на его офлайн режим.

действия с графами

ОТКРЫТИЕ УЗЛА

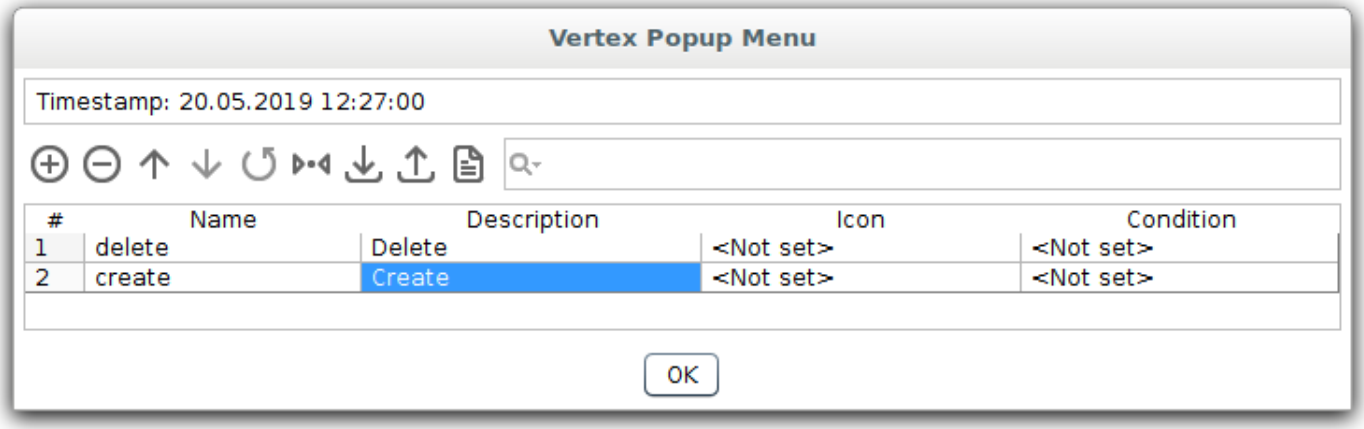
Чтобы выполнить желаемое действие, можно дважды щелкнуть мышью по узлу графа. Например, можно открыть инструментальные панели устройств, соответствующих узлам графа.

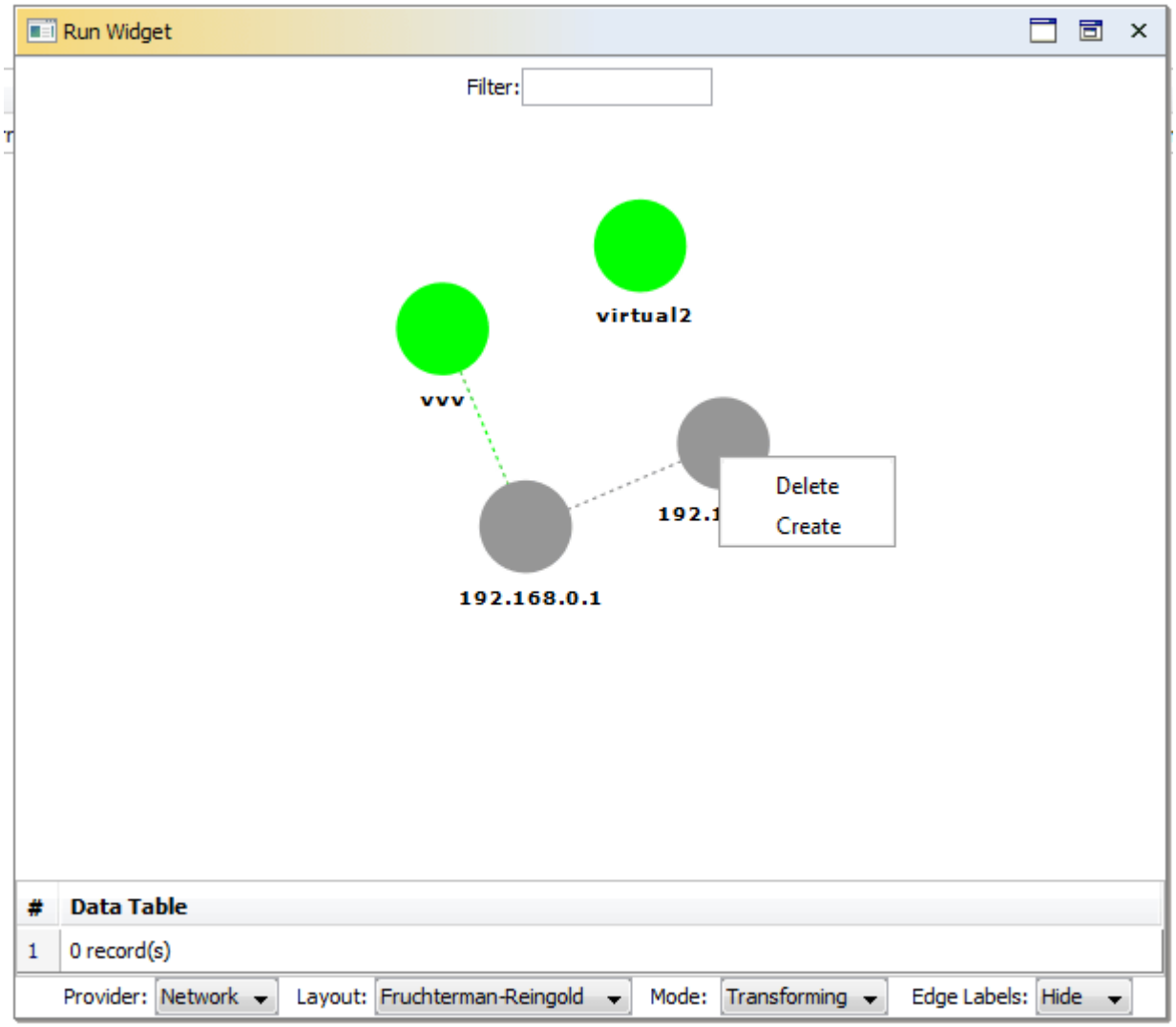
Если Выражение имени предпочитаемого действия 1300 не указано, будет запущено действие по умолчанию 88 контекста.

МЕНЮ ДЕЙСТВИЙ С УЗЛАМИ И РЕБРАМИ

Щелчок правой кнопкой мыши по элементу графа открывает меню действий.

Для узлов открывается меню, определенное свойством Всплывающее меню вершин 1300. Для ребер открывается меню, определенное свойством Всплывающее меню связей 1300.





ИЗМЕНЕНИЕ МАСШТАБА И ПЕРЕМЕЩЕНИЕ ГРАФА

Граф можно **переместить**, нажав на определенную точку и сбросив ее в другой точке карты. Это соответственно переместит центр графа.

Перенос работает только тогда, когда граф находится в режиме **Трансформация**.

Если граф находится в режиме **Выбор**, можно кликнуть мышью по узлу с одновременным нажатием клавиши "Ctrl", чтобы центрировать граф на этом узле.

Изменить масштаб в месте текущего положения курсора, можно при помощи колесика мыши.

ВРАЩЕНИЕ И ПОВОРОТ ГРАФА

Щелчок мышью с одновременным нажатием клавиши "Shift" в окне графа позволяет **вращать** граф.

Щелчок мышью с одновременным нажатием клавиши "Control" в окне графа позволяет **поворачивать** граф.

Вращение и поворот возможны только если граф находится в режиме **Трансформация**.

ВЫБОР И ПЕРЕНОС УЗЛОВ

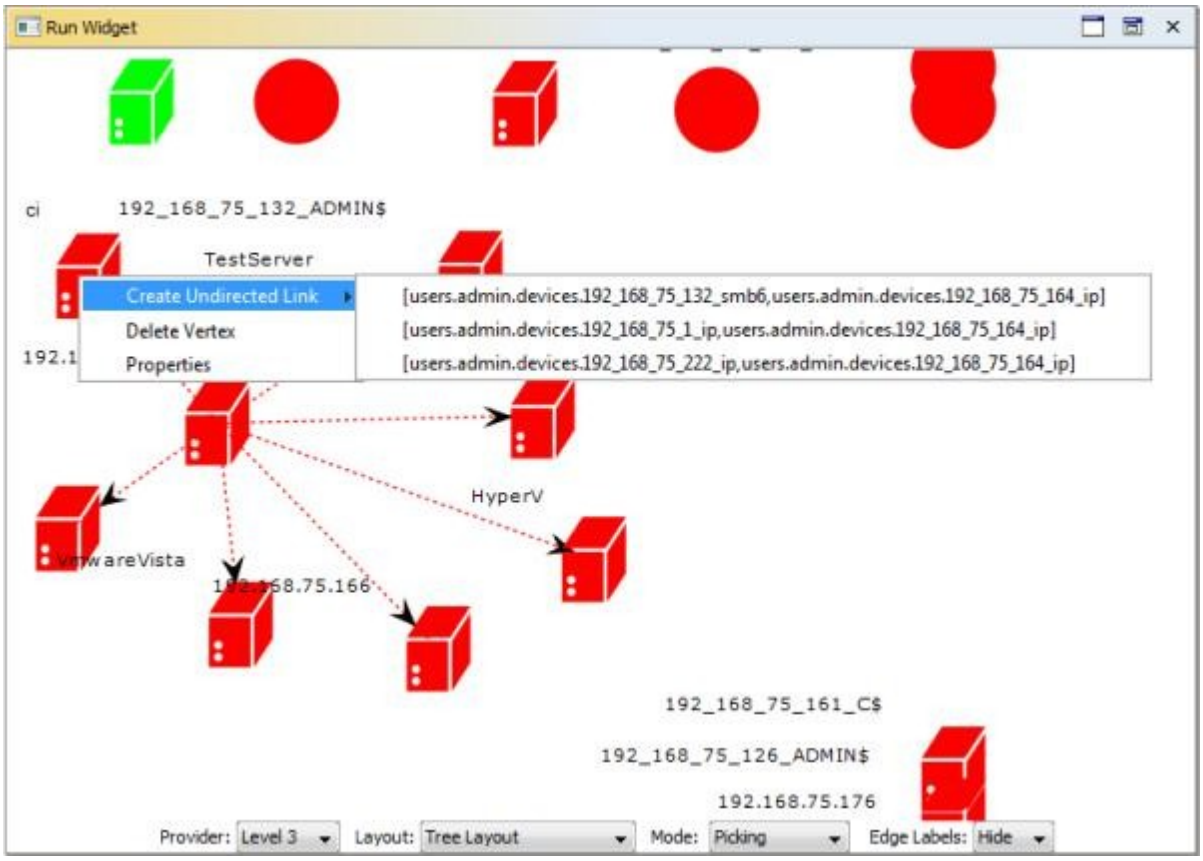
Если граф находится в режиме **Выбор**, ее узлы могут выбираться и перемещаться путем перетаскивания мышью.

Множество выбрать множество узлов:

- Щелкнув за пределами узла и перетащив мышку с целью выбора определенной области графа, включающей несколько узлов.
- Щелчком кнопки мыши с одновременным нажатием клавиши "Shift" на нескольких узлах.

СОЗДАНИЕ СВЯЗЕЙ

Для создания новой связи, выберите группу узлов в режиме Выбор, кликнете правой кнопкой мыши по одному из них, наведите курсор на "Создать ненаправленную связь" и выберите новую связь.



УДАЛЕНИЕ РЕБЕР (СВЯЗЕЙ)

Для удаления связи создайте действие `deleteEdge` во Всплывающем меню связей[1300]. Затем кликните правой кнопкой мыши на ребро (связь) выберите опцию удаления.

УДАЛЕНИЕ УЗЛОВ (ВЕРШИН)

Для удаления узла (вершины) создайте действие `deleteVertex` во Всплывающем меню вершин[1300]. Затем кликните правой кнопкой мыши на узел и выберите опцию удаления.

СОХРАНЕНИЕ, УДАЛЕНИЕ И ВОССТАНОВЛЕНИЕ КОМПОНОВКИ (РАСПОЛОЖЕНИЯ) ГРАФА

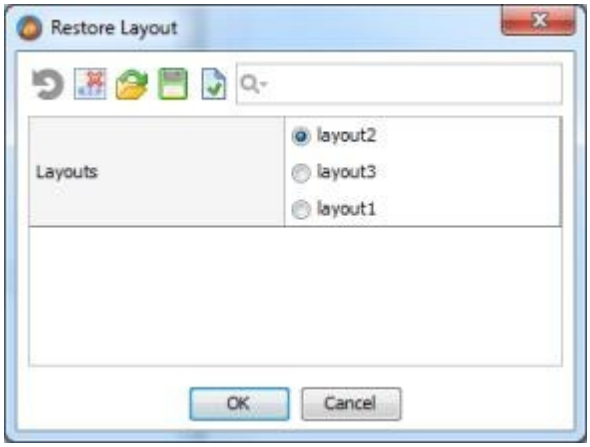
Схемы расположение диаграммы определяются позицией их узлов.

Компонент Граф может сохранять текущую компоновку в базе данных AtomMind Server без вмешательства пользователя. Сохранение текущей компоновки является прозрачной операцией. Если пользователь открывает инструментальную панель или виджет с компонентом Граф, его расположение восстанавливается из базы данных в зависимости от провайдера топологии и типа компоновки графа. Если виджет или инструментальная панель с компонентом Граф закрывается, измененные узлы графа сохраняются в базе данных. Проконтролировать сохранение и восстановление можно при помощи меню контекста компонента, которое открывается при нажатии правой кнопкой мыши:

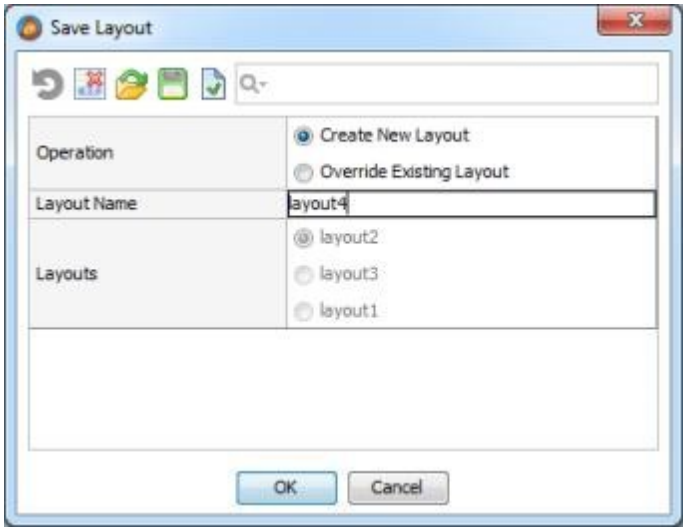
Элемент меню	Описание
Восстановить расположение	Восстановить позиции узлов в зависимости от имени выбранного расположения диаграммы.

Сохранить расположение	Сохранить текущее расположение в базе данных сервера под выбранным именем.
Удалить расположение	Удалить выбранные расположения из базы данных.

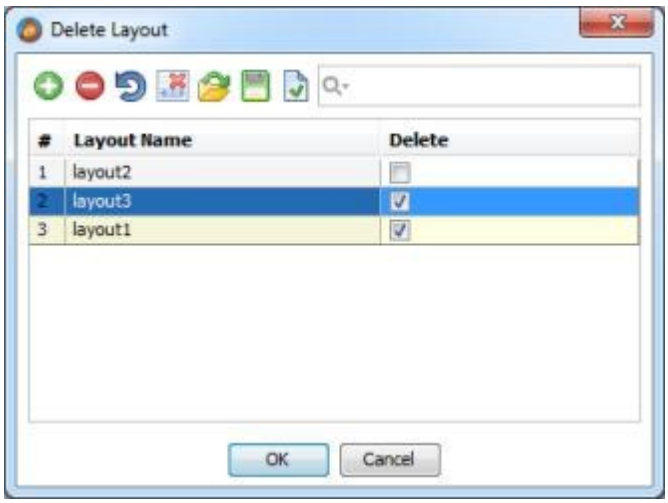
Для восстановления ранее сохраненного расположения, нажмите правой кнопкой мыши на холст и выберите "Восстановить расположение". Затем выберите компоновку для восстановления.



Для сохранения расположения нажмите правой кнопкой мыши на холст и выберите "Сохранить расположение". После этого выберите компоновку для перезаписи или выберите "Создать новое расположение" и введите новое имя расположения. Этот функционал можно отключить при помощи свойства Только чтение 1300.



Для удаления расположения нажмите правой кнопкой мыши на холст и выберите "Удалить расположение". После этого выберите компоновки для удаления, проставив галочки в соответствующих им полях. Этот функционал можно отключить при помощи свойства Только чтение 1300.



Основные свойства

ОБЩИЕ СВОЙСТВА

Граф имеет следующие основные свойства, общие для всех виджетов:

Видимый 1275, Фон 1275, Рамка 1275, Шрифт 1275, Курсор 1276, Точки прикрепления 1284, Всплывающее меню 1276.

ВИЗУАЛИЗАЦИЯ ТОПОЛОГИИ

Граф наследует следующие базовые свойства Визуализации топологии 1300:

Выражение имени предпочитаемого действия 1300, Всплывающее меню вершин 1300, Всплывающее меню связей 1300, Только чтение 1300.

РЕЖИМ

Режимы графа: **Трансформация, Выбор** или **Редактирование**. В режиме Трансформации можно менять масштаб графа, перемещать, вращать и поворачивать. В режиме Выборы узлы графа можно перемещать по отдельности. В режиме Редактирование можно создавать связи между узлами.

Можно поменять режим графа, используя привязку 1295, например, добавить к виджету выбор режима через Поле со списком 975.

Имя свойства: **mode**

Тип свойства: **Integer**

ПЕРИОД ОБНОВЛЕНИЯ

Период обновления графа. Используется для перечитывания информации о топологии с сервера и обновления графа.

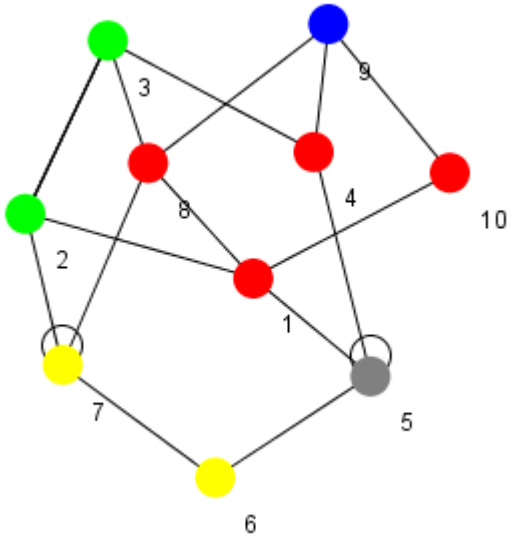
Имя свойства: **updatePeriod**

Тип свойства: **Long**

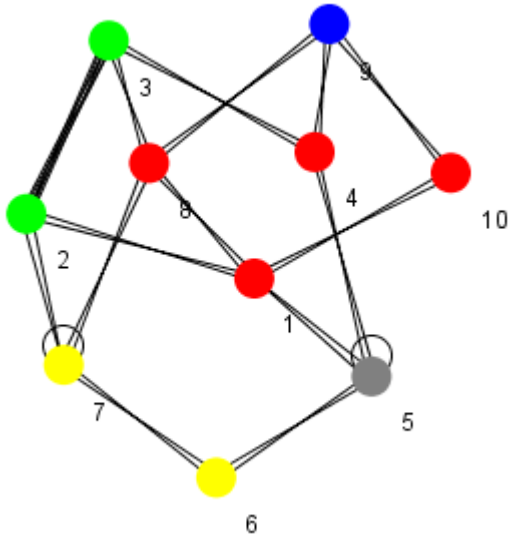
ФОРМА РЕБЕР (СВЯЗЕЙ)

Определяет внешний вид ребер диаграммы. Возможные следующие варианты:

1. Линия

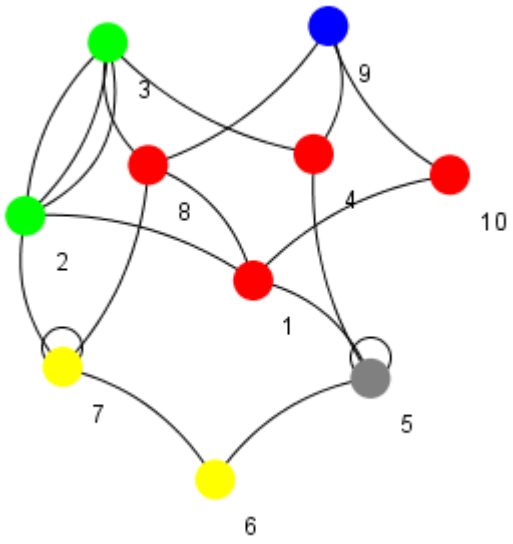


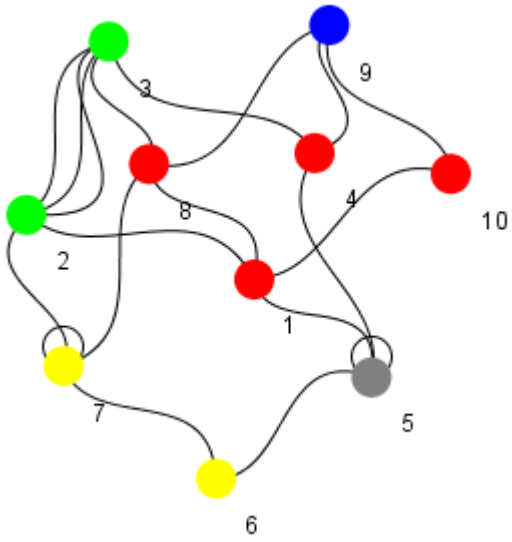
2. Клин



3. Четырехугольная кривая

4. Кубическая кривая





Имя свойства: **edgeShape**

Тип свойства: **Integer**

свойства размера и положения

ОБЩИЕ СВОЙСТВА

Граф имеет следующие свойства размера и положения, общие для всех виджетов:

Ширина 1274, Высота 1274.

ОГРАНИЧЕНИЯ ПОЗИЦИОНИРОВАНИЯ

В зависимости от выбранного позиционирования виджета 950, граф может использовать ограничения либо абсолютного позиционирования 954, либо сетки 950.

свойства привязок

Граф использует свойство Привязки 1275, общее для всех виджетов.

свойства топологии

ВИЗУАЛИЗАЦИЯ ТОПОЛОГИИ

Граф наследует следующие свойства от Визуализации топологии 1300:

Провайдер 1300, Узлы являются контекстами 1300, Показывать несвязанные узлы 1300, Выражение топологии 1300, Выражение узлов 1300, Выражение связей 1300, Выражение идентификаторов связей 1300, Выражение идентификаторов узлов 1300, Выражение источника 1300, Выражение назначения 1300, Выражение цвета 1300, Выражение типа 1300, Выражения для получения изображения 1300, Выражение интерфейса 1300, Выражение направленности 1300, Выражение ширины 1300, Выражение описания связи 1300, Показывать описания связи 1300, Выражение цвета связи 1300, Показывать граничащие узлы 1300, Выражение подсказки узла 1300, Выражение подсказки связи 1300, Выражение описания узла 1300.

МАСКА ВЕРШИН

Маска 44 контекстов, откуда берутся топологические данные. В большинстве случаев каждый контекст, соответствующий этой маске, появляется как отдельный узел графа (вершина).

Имя свойства: **vertexMask**

Тип свойства: **String**

Свойства компоновки

ВИЗУАЛИЗАЦИЯ ТОПОЛОГИИ

Граф наследует следующие свойства компоновки от Визуализации топологии 13001:

Выражение азимута 13001, Выражение размера маркера 13001.

КОМПОНОВКА

Тип компоновки графа. Поддерживаемые компоновки:

- Камада-Кавай
- Фрухтерман-Рейнгольд
- Круг
- Самоорганизующийся Мэйера
- Дерево
- Круговая
- Радиальная

Древовидные, круговые и радиальные компоновки игнорируют первоначальную информацию направления связи. Для данных компоновок направление каждой связи устанавливается автоматически алгоритмом компоновки.

Property name: **layout**

Property type: **Integer**

ПРИТЯЖЕНИЕ

Коэффициент притяжения для компоновки Фрухтерман-Рейнгольд.

Имя свойства: **attraction**

Тип свойства: **Double**

ОТТАЛКИВАНИЕ

Коэффициент отталкивания для компоновки Фрухтерман-Рейнгольд.

Имя свойства: **repulsion**

Тип свойства: **Double**

ПОПРАВКА НА ГРАВИТАЦИЮ

Флажок поправки на гравитацию, используемый компоновкой Камада-Кавай.

Имя свойства: **adjustForGravity**

Тип свойства: **Boolean**

МУЛЬТИПЛИКАТОР РАССТОЯНИЯ МЕЖДУ НЕСВЯЗАННЫМИ ВЕРШИНАМИ

Мультипликатор расстояния между несвязанными вершинами для компоновки Камада-Кавай.

Имя свойства: **disconnectedDistanceMultiplier**

Тип свойства: **Double**

ОБМЕН ВЕРШИНАМИ

Флажок об обмене вершинами для компоновки Камада-Кавай.

Имя свойства: **exchangeVertices**

Тип свойства: **Boolean**

ФАКТОР ДЛИНЫ

Фактор длины для компоновки Камада-Кавай.

Имя свойства: **lengthFactor**

Тип свойства: **Double**

УДАЛЁННОСТЬ ВЕРШИН ПО ГОРИЗОНТАЛИ

Определяет расстояние между узами (вершинами) по горизонтали.

Имя свойства: **horizontalVertexSpacing**

Тип свойства: **Integer**

УДАЛЁННОСТЬ ВЕРШИН ПО ВЕРТИКАЛИ

Определяет расстояние между узами (вершинами) по вертикали.

Имя свойства: **verticalVertexSpacing**

Тип свойства: **Integer**

ФОНОВОЕ ИЗОБРАЖЕНИЕ

Фоновое изображение на холсте.

Имя свойства: **backgroundImage**

Тип свойства: **Data Block**

ФИЛЬТР

Фильтр узлов (вершин). Только те узлы, для которых данная строка указана в их метке, будут выделены на графе.

Имя свойства: **filter**

Тип свойства: **String**

ОБРАТНОЕ НАПРАВЛЕНИЕ РЕБЕР

Определяет направление стрелок для компоновок дерево, круговая и радиальная.

Имя свойства: **revertEdges**

Тип свойства: **Boolean**

ВЫРАЖЕНИЕ X-КООРДИНАТЫ УЗЛА

Выражение, которое возвращает X-координату узла внутри **Пользовательской раскладки**.

Имя свойства: **customLayoutNodeXExpression**

Тип свойства: **String**

ВЫРАЖЕНИЕ Y-КООРДИНАТЫ УЗЛА

Выражение, которое возвращает Y-координату узла внутри **Пользовательской раскладки**.

Имя свойства: **customLayoutNodeTExpression**

Тип свойства: **String**

ВЫРАЖЕНИЕ АЗИМУТА

Выражение, используемое для определения ориентации изображения узла. Должно возвращать угол поворота в радианах.

Имя свойства: **azimuthExpression**

Тип свойства: **String**

ВЫРАЖЕНИЕ РАЗМЕРА МАРКЕРА

Выражение, используемое для определения относительного размера изображения узла. Должно возвращать значения больше 1 для увеличения масштаба изображения, и значения меньше 1 для уменьшения масштаба изображения.

Имя свойства: **ratioExpression**

Тип свойства: **String**

ПОЛЬЗОВАТЕЛЬСКИЕ СВОЙСТВА ИЗОБРАЖЕНИЯ

Это свойство позволяет настраивать свойства CSS-стилей, наследованных в изображениях SVG в результате **Выражения для получения изображения**.

Каждая строка в таблице Пользовательские свойства изображения позволяет настраивать одно динамическое свойство CSS:

- **Имя класса CSS**. Имя класса CSS, используемого в SVG изображении узла.
- **Свойство CSS**. Имя свойства класса CSS, которое будет динамически вычисляться.
- **Выражение значения свойства CSS**. Выражение, которое должно возвращать строковое значение нового свойства.

Среда вычисления 114 Выражения значения свойства CSS	
Контекст по умолчанию 119	Контекст соответствующего узла графа, чье SVG-изображаение обрабатывается.
Таблица данных по умолчанию 120	Результат **Выражения узлов**.
Ряд по умолчанию 119	Обрабатываемый ряд таблицы по умолчанию.
Переменные среды 123	Только стандартные 123 переменные.

пользовательские функции

Граф имеет следующие пользовательские функции.

ВОССТАНОВИТЬ РАСПОЛОЖЕНИЕ

Переключает текущее расположение графа на заданный преднастроенный макет. Можно использовать эту функцию для отмены внесенных в расположение изменений, либо для загрузки пользовательских расположений.

Имя функции: **restoreLayout**

Имя поля	Тип поля	Описание поля
layoutName	String	Имя макета расположения.

СОХРАНИТЬ РАСПОЛОЖЕНИЕ

Сохраняет текущее расположение графа как макет расположения с указанным именем. Можно использовать эту функцию для создания пользовательских расположений.

Имя функции: **saveLayout**

Имя поля	Тип поля	Описание поля
layoutName	String	Имя макета расположения.

УДАЛИТЬ РАСПОЛОЖЕНИЕ

Удаляет определенный макет расположения графа. Можно использовать эту функцию для удаления существующих пользовательских расположений.

Имя функции: **deleteLayout**

Имя поля	Тип поля	Описание поля

layoutName	String	Имя макета расположения

ПОЛУЧИТЬ ИМЕНА РАСПОЛОЖЕНИЙ

Возвращает список существующих макетов расположений.

Имя функции: **getLayoutsNames**

Эта функция не имеет входных параметров.

Выходной формат:

Имя поля	Тип поля	Описание поля
layoutsName	String	Имена существующих макетов расположений.

ПОЛУЧИТЬ РАСПОЛОЖЕНИЕ

Возвращает расположение графа в виде таблицы с двумя подтаблицами: координаты точек и трансформации. Позднее эту таблицу можно использовать как аргумент функции **Обновить расположение**.

Имя функции: **getLayout**

Имя поля	Тип поля	Описание поля
layoutName	String	Имя макета расположения.

ОБНОВИТЬ РАСПОЛОЖЕНИЕ

Обновляет расположение графа новыми параметрами расположения.

Имя функции: **updateLayout**

Имя поля	Тип поля	Описание поля
layoutName	String	Имя макета расположения.
layoutData	Data Table	Данные расположения. Обычно являются результатом функции **Получить расположение**.

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

Пользовательские события

НАЖАТИЕ НА УЗЕЛ

Это событие формируется при нажатии на узел карты.

Имя события: **nodeClicked**

Поля события:

Поле	Имя	Тип	Описание
Node	node	String	Путь нажатого узла/контекста.

НАВЕДЕНИЕ КУРСОРА НА УЗЕЛ

Это событие формируется, когда курсор мыши наведен на узел, отображенный на карте.

Имя события: **nodeHover**

Event fields:

Поле	Имя	Тип	Описание
Node	node	String	Путь узла/контекста, на который наведен курсор мыши.

ОТВЕДЕНИЕ КУРСОРА С УЗЛА

Это событие формируется, когда курсор мыши отводится от узла на карте.

Имя события: **nodeUnhover**

Поля события:

Поле	Имя	Тип	Описание
Node	node	String	Путь узла/контекста, от которого отведен курсор мыши.

НАЖАТИЕ НА СВЯЗЬ

Это событие формируется при нажатии на связь между двумя узлами.

Имя события: **edgeClicked**

Поля события:

Поле	Имя	Тип	Описание
Source	source	String	Узел/контекст источника связи.
Destination	destination	String	Узел/контекст адресата связи.
Source Interface	sourceInterfaceId	String	Интерфейс источника связи. Необходимо когда между двумя узлами существует множество связей.
Destination Interface	destinationInterfaceId	String	Интерфейс адресата связи. Необходимо когда между двумя узлами существует множество связей.

НАВЕДЕНИЕ КУРСОРА НА СВЯЗЬ

Это событие формируется, когда указатель мыши расположен на связи между двумя узлами.

Имя события: **edgeHover**

Поля события:

Поле	Имя	Тип	Описание
Source	source	String	Узел/контекст источника связи.
Destination	destination	String	Узел/контекст адресата связи.
Source Interface	sourceInterfaceId	String	Интерфейс источника связи. Необходимо когда между двумя узлами существует множество связей.
Destination Interface	destinationInterfaceId	String	Интерфейс адресата связи. Необходимо когда между двумя узлами существует множество связей.

ОТВЕДЕНИЕ КУРСОРА ОТ СВЯЗИ

Это событие формируется, когда курсор мыши отведен от связи между двумя узлами.

Имя события: **edgeUnhover**

Поля события:

Поле	Имя	Тип	Описание
Source	source	String	Узел/контекст источника связи.
Destination	destination	String	Узел/контекст адресата связи.

Source Interface	sourceInterfaceId	String	Интерфейс источника связи. Необходимо когда между двумя узлами существует множество связей.
Destination Interface	destinationInterfaceId	String	Интерфейс адресата связи. Необходимо когда между двумя узлами существует множество связей.

СОЗДАТЬ СВЯЗЬ

Это событие формируется, когда в режиме редактирования курсор мыши перетаскивается с одного узла на другой.

Имя события: **edgeCreate**

Поля события:

Поле	Имя	Тип	Описание
Source	source	String	Узел/контекст источника связи.
Destination	destination	String	Узел/контекст адресата связи.
Source Interface	sourceInterfaceId	String	Интерфейс источника связи. Необходимо когда между двумя узлами существует множество связей.
Destination Interface	destinationInterfaceId	String	Интерфейс адресата связи. Необходимо когда между двумя узлами существует множество связей.

УДАЛИТЬ СВЯЗЬ

Это специальное событие с дополнительными функциями генерируется при создании всплывающего меню `deleteEdge`.

Имя события: **edgeDelete**

Поля события:

Поле	Имя	Тип	Описание
Source	source	String	Узел/контекст источника связи.
Destination	destination	String	Узел/контекст адресата связи.
Source Interface	sourceInterfaceId	String	Интерфейс источника связи. Необходимо когда между двумя узлами существует множество связей.
Destination Interface	destinationInterfaceId	String	Интерфейс адресата связи. Необходимо когда между двумя узлами существует множество связей.

ПОКАЗ КОНТЕКСТНОГО МЕНЮ УЗЛА

Событие возникает при вызове контекстного меню кликом правой кнопки мыши на узел.

Имя события: **nodeShowPopup**

Поля события:

Поле	Имя	Тип	Описание
Node	node	String	Путь узла/контекста, для которого отображается контекстное меню.

ПОКАЗ КОНТЕКСТНОГО МЕНЮ СВЯЗИ

Событие возникает при вызове контекстного меню связи кликом правой кнопки мыши на связь между двумя узлами.

Имя события: **edgeShowPopup**

Поля события:

Поле	Имя	Тип	Описание
Source	source	String	Узел/контекст источника связи.

Destination	destination	String	Узел/контекст адресата связи.
Source Interface	sourceInterfaceId	String	Интерфейс источника связи. Необходимо когда между двумя узлами существует множество связей.
Destination Interface	destinationInterfaceId	String	Интерфейс адресата связи. Необходимо когда между двумя узлами существует множество связей.

ПОКАЗ КОНТЕКСТНОГО МЕНЮ ГРАФА

Событие возникает при вызове контекстного меню графа кликом правой кнопки мыши на его холст.

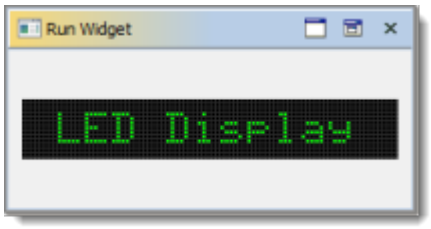
Имя события: **graphShowPopup**

Событие не имеет полей.

13.4.9.29 Светодиодное табло

Этот компонент отображает строку или число, моделируя шкальный или световой индикатор.

Светодиодное табло выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Основной цвет 1275, Фон 1275, Рамка 1275, Курсор 1276, Всплывающая подсказка 1276, Всплывающее меню 1276

Пользовательские свойства

ТЕКСТ (свойство по умолчанию 1274)

Текст, отображаемый светодиодным компонентом. Имейте в виду, что не все символы поддерживаются. Неподдерживаемые символы отображаются в виде знака вопроса.

Имя свойства: **text**

Тип свойства: **String**

ВЫРАВНИВАНИЕ ПО ГОРИЗОНТАЛИ

Выравнивание текста по горизонтали внутри светодиодного табло: По Центру, Слева или Справа.

Имя свойства: **horizontalAlignment**

Тип свойства: **Integer**

ВЫРАВНИВАНИЕ ПО ВЕРТИКАЛИ

Выравнивание текста по вертикали внутри светодиодного табло: По центру, Сверху или Снизу.

Имя свойства: **verticalAlignment**

Тип свойства: **Integer**

ЦВЕТ ТОЧЕК В ОФЛАЙН

Цвет точек выключенного табло.

Имя свойства: **dotOffColor**

Тип свойства: **Color**

ШИРИНА ТОЧКИ

Ширина одной точки.

Имя свойства: **dotWidth**

Тип свойства: **Integer**

ВЫСОТА ТОЧКИ

Высота одной точки.

Имя свойства: **dotHeight**

Тип свойства: **Integer**

ГОРИЗОНТАЛЬНЫЙ ИНТЕРВАЛ

Горизонтальный интервал между точками.

Имя свойства: **horizontalGap**

Тип свойства: **Integer**

ВЕРТИКАЛЬНЫЙ ИНТЕРВАЛ

Вертикальный интервал между точками.

Имя свойства: **verticalGap**

Тип свойства: **Integer**

ВНУТРЕННЕЕ ПОЛЕ

Количество никогда не загорающихся точек на левой, правой, верхней и нижней стороне табло.

Имя свойства: **padding**

Тип свойства: **Data Table**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.9.30 Группа кнопок

Группа кнопок является особым *неотображаемым* компонентом, который логически присоединяет несколько зависимых кнопок 973 или триггерных кнопок 971, чтобы объединить их в группу для работы с одной моделью набора данных (например, полю переменной 61).

Группа кнопок является особым типом компонента. Она не отображается в компоновке виджета.

Группы кнопок используются для логической группировки, а не для визуальной. Чтобы создать визуальную группу зависимых кнопок, вам потребуется создать панель 1045 или подобный контейнер, который будет содержать несколько зависимых кнопок, и выделит ее среди окружающих компонентов границами.

О том, как создать привязки 1295 для групп кнопок см. здесь 442.

Свойства группы кнопок

ПРИВЯЗКИ

Данное свойство отображает список привязок, относящихся к группе кнопок. Дополнительную информацию см. здесь 1275.

Имя свойства: **bindings**

Тип свойства: **Data Table**

ИМЯ ВЫБРАННОГО КОМПОНЕНТА

Имя выбранной (нажатой) зависимой или триггерной кнопки.

Имя свойства: **selectedButton**

Тип свойства: **String**

13.4.10 Контейнеры

Контейнеры могут содержать несколько других компонентов 955. Далее приведен список контейнеров, поддерживаемых процессором виджета:


Панель 1045


Многослойная панель 1049


Панель с вкладками 1045


Панель с разделителями 1047


Подвиджет 1049


Перемещение всех выбранных компонентов (даже если они содержатся в разных контейнерах) приведет к перемещению всех компонентов на целевую панель.

Общие свойства контейнеров

Данная глава описывает свойства, поддерживаемые большинством контейнеров. Описание каждого контейнера включает в себя список общих свойств контейнера.

КОМПОНОВКА

Данное свойство определяет тип компоновки 950 контейнера.

Имя свойства: **layout**

Тип свойства: **Integer**

ПРОКРУТКА

Данное свойство определяет, когда включена прокрутка. Оно имеет следующие опции:

- **Отсутствует**. Горизонтальная и вертикальная прокрутка отключена.
- **По вертикали**. Вертикальная прокрутка включена, горизонтальная отключена.
- **По горизонтали**. Горизонтальная прокрутка включена, вертикальная отключена.
- **В обоих направлениях**. Включены оба варианта прокрутки.

Имя свойства: **scrolling**

Тип свойства: **Integer**

АВТОМАТИЧЕСКОЕ УПРАВЛЕНИЕ ПРОКРУТКОЙ

Данное свойство доступно только тогда, когда прокрутка 1041 установлена на **Вертикальная**, **Горизонтальная** и **В обоих направлениях**. Оно определяет, когда в контейнере появляются полосы прокрутки. Если данная опция включена, полосы прокрутки появляются только тогда, когда содержание контейнера не помещается в область изображения и может потребоваться прокрутка экрана вручную. Если данная опция отключена, полосы прокрутки видимы всегда.

Имя свойства: **scrollingAuto**

Тип свойства: **Boolean**

ИСПОЛЬЗОВАНИЕ НАСТРАИВАЕМОЙ ПРОКРУТКИ

Делает возможным использование настраиваемой прокрутки.

Имя свойства: **useCustomScrollBar**

Тип свойства: **Boolean**

СВОЙСТВА ПОЛОСЫ ПРОКРУТКИ

Позволяет настраивать цвета слайдера и кнопок, а также ширину полосы прокрутки.

Имя свойства: **scrollBarProperties**

Тип свойства: **Data Table**

13.4.10.1 Корневая панель

Корневая панель представляет собой особый тип контейнера Панель 1045 с дополнительными свойствами 1274 и событиями 1285.

Общие свойства

Корневая панель поддерживает следующие общие свойства компонента виджет:

Ширина 1274, Высота 1274, Привязки 1275, Фон 1275, Непрозрачный 1275, Граница 1275, Курсор 1276, Всплывающая посказка 1276, Фокусируемый 1276, Всплывающее меню 1276

И следующие общие свойства контейнера:

Компоновка 1041, Прокрутка 1041, Автоматическое управление прокруткой 1041

Пользовательские свойства

Данный раздел описывает свойства корневой панели 1042 виджета.

ИГНОРИРОВАТЬ ОШИБКИ ПРИВЯЗОК

Если данная опция включена, любые ошибки привязок 1295, возникающие во время выполнения виджета, не показываются пользователю в диалоговом окне сообщения об ошибке. Вместо этого они записываются в файл журнала или консоль оператора (см. тему Ведение журнала 422 для получения более подробной информации). Если данная опция отключена (значение по умолчанию), ошибки привязки показываются пользователю и записываются в журнал.

Имя свойства: **ignoreBindingErrors**

Тип свойства: **Boolean**

ВСЕ ПРИВЯЗКИ

Данное свойство содержит список всех доступных привязок в шаблоне виджета. Более подробную информацию о привязках и их параметрах см. в статье Привязки 1295.

Имя свойства: **allBindings**

Тип свойства: **Data Table**

СКРИПТЫ

Данное свойство содержит таблицу скриптов виджета 1308. Она состоит из двух столбцов: **Имя** и **Код**.

Имя - это имя скрипта. Оно используется для ссылки на данный скрипт из привязок виджета 1295.

Код содержит текст класса, реализующего интерфейс `WidgetScript` , т.е. код скрипта.

Когда в таблицу скриптов добавляется новый скрипт, новая привязка, используемая для его запуска, автоматически добавляется в список Все привязки 1042. Однако начальные свойства 1295 данной привязки не определяют условия запуска скрипта. Поэтому необходимо изменить данные свойства, чтобы обеспечить выполнение скрипта при запуске виджета, какого-либо события или просто время от времени.

Имя свойства: **scripts**

Тип свойства: **Data Table**

ОБЫЧНЫЕ ОДНОВРЕМЕННО ВЫПОЛНЯЕМЫЕ ПРИВЯЗКИ

Определяет размер ядра пула потоков виджета, то есть базовое число одновременно обрабатываемых привязок.

Нулевое значение отключает обработку параллельных привязок.

Для более подробной информации см. Производительность привязок 742.

Эта настройка игнорируется, если виджет используется как вложенный виджет в другом виджете. Вложенные виджеты наследуют пул потока своего родительского виджета.

Имя свойства: **normalConcurrentBindings**

Тип свойства: **Integer**

МАКСИМАЛЬНЫЕ ОДНОВРЕМЕННО ВЫПОЛНЯЕМЫЕ ПРИВЯЗКИ

Определяет максимальный размер пула потоков виджета, то есть количество одновременно обрабатываемых привязок, разрешенных в случае переполнения очереди привязок.

Нулевое значение отключает обработку параллельных привязок.

Для более подробной информации см. Производительность привязок 742.

Эта настройка игнорируется, если виджет используется как вложенный виджет в другом виджете. Вложенные виджеты наследуют пул потока своего родительского виджета.

Имя свойства: **maximumConcurrentBindings**

Тип свойства: **Integer**

МАКСИМАЛЬНАЯ ДЛИНА ОЧЕРЕДИ НЕОБРАБОТАННЫХ ПРИВЯЗОК

Определяет, сколько необработанных операций привязки может быть поставлено в очередь, прежде чем размер пула потоков виджета превысит размер его ядра относительно максимального размера.

Для более подробной информации см. Производительность привязок 742.

Эта настройка игнорируется, если виджет используется как вложенный виджет в другом виджете. Вложенные виджеты наследуют пул потока своего родительского виджета.

Имя свойства: **maximumBindingQueueLength**

Тип свойства: **Integer**

ПАРАЛЛЕЛЬНЫЕ ПРОЦЕССЫ ПРИВЯЗОК ЗАПУСКА

Определяет, могут ли привязки запуска выполняться в параллельных потоках.

Включение этой опции вызывает обработку привязок запуска в случайном порядке.

Имя свойства: **startupBindingsConcurrency**

Тип свойства: **Boolean**

Сенсорный экран

Настройки этой группы относятся к поддержке сенсорной панели 1316.

РАЗМЕР ШРИФТА ВСТРОЕННОЙ ЭКРАННОЙ КЛАВИАТУРЫ

Определяет, какой шрифт используется для подписи клавиш экранной клавиатуры.

Имя свойства: **embeddedOnscreenKeyboardFontSize**

Тип свойства: **Integer**

КОМАНДА ЭКРАННОЙ КЛАВИАТУРЫ

Путь запускаемого исполняемого файла приложения экранной клавиатуры, когда запрашиваются данные, вводимые с клавиатуры.

Существует несколько заранее определенных вариантов выбора (несмотря на другие файлы, на которые можно сослаться):

- Отсутствует (вызывает использование собственной встроенной экранной клавиатуры AtomMind)
- Экранная клавиатура Windows (`C:\Windows/system32/osk.exe`)
- Встроенный GNOME (`onboard`)
- KDE Kvkbd (`kvkbd`)
- Экранная клавиатура Mac OS (`open -a KeyboardViewerServer`)

Имя свойства: **nativeOnscreenKeyboardCommand**

Тип свойства: **String**

Пользовательские функции

ПОКАЗАТЬ ВСПЛЫВАЮЩЕЕ ОКНО

Позволяет отображать определнный виджет как всплывающее окно.

Имя поля	Тип поля	Описание поля
widget	String	Полное имя контекста виджета для отображения как всплывающее окно.
defaultContext	String	Полное имя контекста для использования по умолчанию во всплывающем виджете.
customProperties	Data Table	Таблица дополнительных параметров. Будет применяться как таблица по умолчанию во всплывающем виджете.

СКРЫТЬ ВСПЛЫВАЮЩЕЕ ОКНО

Позволяет скрыть текущее всплывающее окно.

ВЫПОЛНИТЬ ДЕЙСТВИЕ

Позволяет выполнять действие из контекста.

Имя поля	Тип поля	Описание поля
action	String	Имя вызываемого действия.
context	String	Полное имя контекста, где будет выполняться действие.
properties	Data Table	Таблица параметров действия. Будет передана действию в качестве параметров.

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

Пользовательские события

ОТПРАВКА

Данное событие активируется кнопкой действия **Отправить** в списке действий 970.

Имя события: **submit**

ВЫХОД

Данное событие активируется кнопкой действия **Выход** в списке действий 970.

Событие выступает триггером для выхода из текущего проигрывателя виджетов, веб-проигрывателя виджетов или сессии в веб интерфейсе. Оно возвращает пользователя на экран авторизации.

Имя события: **logout**

ЗАКРЫТИЕ

Данное событие активируется кнопкой действия **Закрыть** в списке действий 970.

Имя события: **close**

13.4.10.2 Панель

Панель является типовым контейнером. Представляет собой простую сетчатую компоновку 950 без дополнительного оформления. Панель можно поместить в ячейку сетки компоновки контейнера-родителя для формирования комплексных интерфейсов.

Таким образом выглядит непрозрачная панель с синим фоном, содержащая одно текстовое поле 958:

По умолчанию панель является прозрачной, т.е. её фон не прорисован, чтобы были видны любые компоненты под ним. Непрозрачные панели имеют цветной фон. К панелям можно также добавить рамки.

Общие свойства

Панель поддерживает следующие общие свойства виджета:

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Фон 1275, Непрозрачный 1275, Рамка 1275, Курсор 1276, Всплывающая подсказка 1276, Всплывающее меню 1276

А также следующие общие свойства контейнера:

Компоновка 1041, Прокрутка 1041, Автоматическое управление прокруткой 1041

Пользовательские свойства

Панель не имеет пользовательских свойств.

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.10.3 Панель с вкладками

Панель с вкладками позволяет пользователю переключаться между группами компонентов нажатием вкладки с определенным названием. Каждая вкладка содержит отдельное подокно компоновка 950, в состав которого могут входить другие компоненты или контейнеры.

Панель с двумя пустыми вкладками выглядит следующим образом:

Общие свойства

Панель с вкладками поддерживает следующие общие свойства виджета:

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Фон 1275, Непрозрачный 1275, Курсор 1276, Рамка 1275, Всплывающая подсказка 1276, Фокусируемый 1276, Всплывающее меню 1276

А также следующие общие свойства контейнера:

Прокрутка 1041, Автоматическое управление прокруткой 1041

Пользовательские свойства

АКТИВНАЯ ВКЛАДКА

Имя активной вкладки.

Имя свойства: **activeTab**

Тип свойства: **String**

ВКЛАДКИ ВИДИМЫ

Контролирует видимость переключателя вкладок.

Имя свойства: **tabVisible**

Тип свойства: **Boolean**

РАСПОЛОЖЕНИЕ ВКЛАДОК

Положение заголовков вкладок: Сверху, Слева, Снизу или Справа.

Имя свойства: **tabPlacement**

Тип свойства: **Integer**

ПОЛИТИКА РАСПОЛОЖЕНИЯ ВКЛАДОК

Определяет положение вкладок, если они не помещаются в одном ряду:

- **Перенос вкладок**. Создает множество рядов с вкладками.
- **Прокрутка вкладок**. Добавляет кнопки прокрутки к заголовку подокна с вкладками.

Имя свойства: **tabLayoutPolicy**

Тип свойства: **Integer**

Добавление новых вкладок

Добавить новые вкладки к панели можно при помощи пункта контекстного меню **Добавить вкладку** узла Дерево ресурсов 428 для панели с вкладками.

Свойства вкладок

Общие свойства: Компоновка 1041, Привязки 1275

ПОРЯДКОВЫЙ НОМЕР

Порядковый номер вкладки. Вкладки упорядочены по возрастанию согласно их порядковым номерам.

Имя свойства: **index**

Тип свойства: **Integer**

ЗАГОЛОВОК

Заголовок вкладки.

Имя свойства: **title**

Тип свойства: **String**

ПИКТОГРАММА

Изображение около заголовка вкладки. Если значение свойства является NULL, изображение не отображается.

Имя свойства: **icon**

Тип свойства: **Data Block**

АКТИВНЫЙ

Указывает на то, что вкладка активна.

Имя свойства: **enabled**

Тип свойства: **Boolean**

ОСНОВНОЙ ЦВЕТ

Цвет шрифта заголовка вкладки.

Имя свойства: **foreground**

Тип свойства: **Color**

ФОН

Цвет фона вкладки.

Имя свойства: **background**

Тип свойства: **Color**

ВСПЛЫВАЮЩАЯ ПОДСКАЗКА

Текст всплывающей подсказки вкладки.

Имя свойства: **toolTipText**

Тип свойства: **String**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.10.4 Панель с разделителем

Панель с разделителем используется для разделения двух (и только двух) разных компонентов или контейнеров, которые называются рамки. Данные компоненты отображаются либо сбоку друг от друга, либо друг над другом. Доступное пространство для каждого из этих компонентов может быть интерактивно перераспределено пользователем.

Панель с разделителем горизонтальной ориентации с двумя метками 956 выглядит следующим образом:

Возможно разделить пространство между тремя и более компонентами, поместив разделенные подокна в другие разделенные подокна.

Общие свойства

Панель с разделителем поддерживает следующие общие свойства виджета:

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Фон 1275, Непрозрачный 1275, Рамка 1275, Курсор 1276, Всплывающая подсказка 1276, Всплывающее меню 1276

А также следующие общие свойства контейнера:

Прокрутка 1041, Автоматическое управление прокруткой 1041

Пользовательские свойства

ОРИЕНТАЦИЯ

По горизонтали или **по вертикали**. Панель с горизонтальной ориентацией имеет левую и правую рамки, в то время как панель с вертикальной ориентацией имеет верхнюю и нижнюю рамки.

Имя свойства: **orientaton**

Тип свойства: **Integer**

ПОЛОЖЕНИЕ РАЗДЕЛИТЕЛЯ

Положение разделителя в пикселях в левой или верхней части панели.

Имя свойства: **dividerLocation**

Тип свойства: **Integer**

РАЗМЕР РАЗДЕЛИТЕЛЯ

Размер разделителя в пикселях.

Имя свойства: **dividerSize**

Тип свойства: **Integer**

Свойства рамки

Общие свойства: Компоновка 1041, Привязки 1275

ПОРЯДКОВЫЙ НОМЕР

Порядковый номер рамки. Рамка с меньшим порядковым номером расположена в левой части панели (или в верхней части - при вертикальной ориентации).

Имя свойства: **index**

Тип свойства: **Integer**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.10.5 Многослойная панель

Многослойная панель состоит из нескольких слоев, изображаемых друг над другом. Каждый слой имеет отдельную Компоновку 950.

Далее представлен вид двухслойной панели, содержащей текстовую область 965 и метку 956:

Общие свойства

Панель с разделителем поддерживает следующие общие свойства виджета:

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Фон 1275, Непрозрачный 1275, Рамка 1275, Курсор 1276, Всплывающая подсказка 1276, Всплывающее меню 1276

А также следующие общие свойства контейнера:

Прокрутка 1041, Автоматическое управление прокруткой 1041

Пользовательские свойства

Многоуровневая панель не имеет пользовательских свойств.

Добавление новых слоев

Новые слои можно добавить при помощи пункта контекстного меню **Добавить слой** узла панели Дерево ресурсов 428.

Свойства слоя

Общие свойства: Компоновка 1041, Привязки 1275

ГЛУБИНА

Глубина уровня, также называемая *z-order.* Слои с меньшей глубиной отображаются над уровнями, глубина которых больше.

Имя свойства: **depth**

Тип свойства: **Integer**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.10.6 Подвиджет

Компонент "Подвиджет" служит для отображения виджета внутри другого виджета.

Внешний вид компонента "Подвиджет" полностью настраиваемый, поскольку подвиджет, на самом деле, - это обычный виджет, который просто вставляется в другой виджет. Компонент "Подвиджет" может выглядеть следующим образом:

Вложенный виджет, представленный компонентом "Подвиджет", может взаимодействовать с его виджетом-хостом:

- Во-первых, создается виджет, действующий как подвиджет.
- Во-вторых, некоторые пользовательские свойства 1277 добавляются к корневой панели 1042 виджета. Эти свойства используются как "мост" между виджетом-хостом и подвиджетом.

Пример: Корневая панель подвиджета "Конвейер" (см. изображение сверху) имеет логическое свойство **Обработка**, которое запускает/останавливает конвейерную ленту.

- В-третьих, вышеобозначенные пользовательские свойства привязаны 1295 к некоторым свойствам компонента внутри "вложенного" виджета. Например, в подвиджете "Конвейер".

Пример: Свойство **Обработка** подвиджета "Конвейер" привязано к свойствам **Выполнить** нескольких компонентов Векторного рисунка 995, которые приводят в действие конвейерную ленту внутри вложенного виджета.

- В-четвертых, создается виджет-хост и к нему добавляется компонент "Подвиджет". Свойство **Ссылка** компонента "Подвиджет" выставляется так, чтобы указывать на "вложенный" виджет. После этого свойство **Обработка** вложенного виджета появляется как обычное свойство компонента "Подвиджет" внутри виджета-хоста.
- В-пятых, "унаследованные" свойства подвиджета привязаны 1295 к некоторым свойствам компонента виджета-хоста. Это завершает построение взаимодействия виджета и подвиджета.

Пример: Свойство **Обработка** подвиджета "Конвейер" привязано к свойству **Активный** триггерной кнопки "Главный переключатель" внутри виджета-хоста. Таким образом, нажимая на эту кнопку для запуска/остановки всей линии розлива, вы также запустите/остановите конвейерную ленту.

Пожалуйста, сохраните изменения как виджета, так и подвиджета и вновь откройте их в GUI Builder, чтобы увидеть результат конфигурации виджета и подвиджета.

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Рамка 1275, Всплывающая подсказка 1276

Пользовательские свойства

ССЫЛКА

Путь контекста виджета 1627, соответствующий виджету, который загружает и отображает этот компонент "Подвиджет" изнутри.

Имя свойства: **reference**

Тип свойства: **String**

КОНТЕКСТ ПО УМОЛЧАНИЮ

Выражение 112, которое должно вернуть путь контекста, который станет контекстом по умолчанию 946 для вложенного виджета. `{.:}` (или, аналогичным образом, `dc()`) выражение станет причиной наследования контекста по умолчанию от данного виджета.

Имя свойства: **defaultContext**

Тип свойства: **String**

ОЖИДАТЬ ВЫПОЛНЕНИЯ ПРИВЯЗОК РОДИТЕЛЬСКОГО ЭЛЕМЕНТА

Определяет необходимость выполнения привязок при запуске подвиджета после выполнения корневой панелью всех привязок при запуске.

Отключение этой опции приведет к обработке привязок при запуске подвиджета и корневой панели в случайном порядке.

Имя свойства: **waitParentBindingsExecution**

Тип свойства: **Boolean**

СВОЙСТВА ПОДВИДЖЕТА

Компонент "Подвиджет" представляет пользовательские свойства соответствующей корневой панели 1042 виджета, как собственные, чтобы было возможно взаимодействие между виджетом-хостом и подвиджетом.

Общие свойства

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.11 Графики

Графики используются для графического представления данных. Поддерживаются следующие типы графиков:

ГРАФИКИ КАТЕГОРИЙ


Диаграмма 1067


Диаграмма с областями 1072


Столбчатая диаграмма 1074


Столбчатая диаграмма с интервалом 1085


Статистическая диаграмма 1087


Диаграмма Ганта 1090


Смешанная диаграмма 1092


Диаграмма с общей осью параметров 1096


Диаграмма с общей осью значений 1097

XY-ГРАФИКИ


XY-диаграмма 1100


XY-диаграмма с областями 1106

 XY-столбчатая диаграмма 1110

 Диаграмма интервалов 1119

 Диаграмма погрешностей 1121

 Диаграмма отклонений 1124

 Векторная диаграмма 1126

 Пузырьковая диаграмма 1127

 Финансовая диаграмма 1130

 Блочная диаграмма 1133

 Смешанная XY-диаграмма 1139

 XY-диаграмма с общей осью параметров 1143

 XY-диаграмма с общей осью значений 1146

ПРОЧИЕ ГРАФИКИ

 Лепестковая диаграмма 1149

 Полярная диаграмма 1153

 Круговая диаграмма 1156

 Кольцевая диаграмма 1158

Масштабирование графика

Компонент "График" поддерживает функцию масштабирования при помощи всплывающего меню или перетаскивания мышью отображаемого графика.

Контекстное меню графика

Всплывающее меню графика поддерживает:

- точную настройку свойства графика в процессе работы
- экспорт графиков в файлы изображений
- печать графиков
- копирование данных графиков в буфер обмена данных
- различные опции масштабирования
- просмотр исходных данных графика в Редакторе таблицы данных 382

Обратите внимание: текущая версия Web UI предоставляет доступ только к различным функциям масштабирования через контекстное меню графиков.

Структура графиков

Большинство графиков состоят из следующих основных частей:

- Массив данных 1053, построенный из серверных данных
- Область построения 1185, где отображаются элементы данных
- Отрисовщик 1221, выполняющий отображение в области построения (данная часть является дополнительной, некоторые области построения изображают данные на графике сами).
- Оси 1162, относящиеся к графикам (также являются дополнительными, некоторые графики не имеют осей)

Помимо этого поддерживаются многочисленные элементы графиков:

- Заголовки
- Легенды
- Фреймы
- Примечания
- Линии сетки
- Перекрестия
- Метки элементов данных
- И другие

13.4.11.1 Построение массива данных графика

Графики определяют несколько способов извлечения табличных данных из дерева контекстов 41 AtomMind Server и их применения для построения массива данных графика.

Данный раздел посвящен описанию свойств графика, которые относятся к извлечению данных источника и построения массива данных графика.

Свойства графика, относящиеся к данным

ТИП ИСТОЧНИКА ДАННЫХ

Данный основной параметр определяет, какой тип данных должен использоваться для построения графика. Здесь представлены четыре варианта:

- Пользовательские данные 1057. Использование пользовательских табличных данных для построения графика.
- Событие 1064. Использование данных из истории событий 73 и возникновений новых событий для построения и динамического обновления графика.
- Переменная 1061. Использование данных из истории изменений переменной 61 и изменений переменной в реальном времени для построения и динамического обновления графика.
- Трендинг 1065. Использование массива данных другого графика для построения тренда, такого как "смещенная средняя", линейная/экспоненциальная регрессия и т.д.
- Зависимый 1064. Использование массива данных другого графика для построения зависимого графика, который может обрабатывать исходный массив данных при помощи привязок зависимых графиков.

Пользовательские данные используются во всех графиках. Данные типа **Событие** и **Переменная** доступны для применения в таких графиках, как XY-диаграмма 1100, XY-диаграмма с областями 1106 и XY-столбчатая диаграмма 1110. **Тренды** доступны для этих трех графиков, если они являются частью смешанной XY-диаграммы 1139.

Имя свойства: **source**

Тип свойства: **Целое**

ИСХОДНЫЕ СОБЫТИЯ

Данное свойство определяет, что используется для построения графика, основанного на событии[1064]. Оно имеет табличный формат. Каждая строка в таблице определяет серии данных, которые будут отображаться на графике. Таким образом, один график может показывать изменения различных значений и даже значения, полученные из расчета данных различных событий.

Каждые серии данных обладают следующими свойствами:

<table>
<tr><th>Свойство</th><th>Тип</th><th>Описание</th></tr>
<tr><td>Имя</td><td>строка</td><td>Имя серии данных, указанное на графике.</td></tr>
<tr><td>Контекст</td><td>строка</td><td>Контекст[41] сервера, определяющий событие, из которого берутся данные.</td></tr>
<tr><td>Событие</td><td>строка</td><td>Событие[73], из которого берутся данные.</td></tr>
<tr><td>Выражение</td><td>строка</td><td>Выражение[112], используемое для расчета значений данных. В большинстве случаев оно должно включать ссылки[117] на ячейки таблицы данных события[73].
<table>
<tr><th colspan="2">Среда вычисления[112] выражения серии переменной:</th></tr>
<tr><td>Контекст по умолчанию[119]</td><td>Определяется настройкой серии Контекст.</td></tr>
<tr><td>Таблица данных по умолчанию[120]</td><td>Значение переменной, определяемой настройкой серии Событие.</td></tr>
<tr><td>Номер ряда по умолчанию[119]</td><td>0</td></tr>
<tr><td>Переменные среды[123]</td><td>Только стандартные[123] переменные.</td></tr>
</table></td></tr>
<tr><td>Выражение даты</td><td>строка</td><td>Выражение[112], используемое для расчета временных отметок значения (например, значения осей домена). Если это значение не определено (это поведение по умолчанию), график представляет значения на временных метках, когда их модификация регистрируется сервером.

Выражение даты полезно, когда значение переменной Таблицы данных[49], полученное от устройства (например, управляющее устойство в режиме реального времени), содержит как временные метки, так и значения. В этом случае каждая выборка данных содержит как значения осей домена (временная метка), так и значение промежутков осей (число), точно позиционирующие выборку данных графика.
<table>
<tr><th colspan="2">Среда вычисления[112] выражения даты серии переменной:</th></tr>
<tr><td>Контекст по умолчанию[119]</td><td>Определяется настройкой серии Контекст.</td></tr>
<tr><td>Таблица данных по умолчанию[120]</td><td>Значение переменной, определяемой настройкой серии Событие.</td></tr>
<tr><td>Номер ряда по умолчанию[119]</td><td>0</td></tr>
</table></td></tr>
</table>

		Переменные среды 123 — Только стандартные 123 переменные.
Агрегирование	целое	Определяет дальнейшие действия, если некоторые значения данных существуют в течении данного интервала времени. Возможные варианты: расчет среднего, взятие минимального или максимального значения, расчет суммарного значения, взятие первого/последнего значения для указанного интервала.

Имя свойства: **eventSeries**

Тип свойства: **Таблица данных**

ИСХОДНЫЕ ПЕРЕМЕННЫЕ

Данное свойство определяет, какие данные используются для построения графика, основанного на переменной 1061. Имеет табличный формат. Каждая строка в таблице определяет одну серию данных, отображаемую на графике. Таким образом, один график может показывать изменения нескольких значений и даже значения, рассчитанные из данных, взятых от разных переменных.

Каждая серия данных обладает следующими свойствами:

Свойство	Тип	Описание
Имя	строка	Имя серии данных, указанное на графике. Может быть выражением 112 или просто строковой константой.
Контекст	строка	Контекст 41 сервера, определяющий событие, из которого берутся данные.
Переменная	строка	Переменная 61, из которой берутся данные.
Выражение	строка	Выражение 112, используемое для расчета значений данных. В большинстве случаев включает ссылки 117 на значение переменной. Среда вычисления 112 выражения серии переменной: Контекст по умолчанию 119 — Определяется настройкой серии **Контест**. Таблица данных по умолчанию 120 — Значение переменной, определяемое настройкой серии **Переменная**. Ряд по умолчанию 119 — 0 (или динамическое, если включено свойство размножение серии 1062). Переменные среды 123 — Только стандартные 123 переменные.
Размножать	логическое	Создает отдельную серию данных для каждой записи исходной переменной. Для получения более подробной информации см. размножение серии 1062.
Агрегирование	целое	Определяет дальнейшие действия, если некоторые значения данных существуют в течение определенного интервала времени. Возможные варианты: расчет среднего, взятие минимального или максимального значения, расчет суммарного значения, взятие первого/последнего значения для указанного интервала.
Тип	целое	Тип 1062 серии данных.

Имя свойства: **variableSeries**

Тип свойства: **Таблица данных**

ПЕРИОД ВРЕМЕНИ

Единица времени определяет вычисление для графиков, основанных на событии[1064] и на переменной[1061]. На графике отображается только одно значение данных за единицу времени. Например, если единица времени для графика события является **День** и в течение определенного дня возникло несколько событий, на графике отображаются данные только одного (последнего) события за день.

Доступные единицы времени: **Год**, **Квартал**, **Месяц**, **Неделя**, **День**, **Час**, **Минута** и **Секунда**.

Имя свойства: **timeUnit**

Тип свойства: **Целое**

ВКЛЮЧИТЬ ИСТОРИЧЕСКИЕ ДАННЫЕ

Данное свойство определяет, будет ли основанный на событии[1064] или на переменной[1061] график включать в себя предысторию, т.е. изменения в событиях и переменных, произошедшие до построения графика.

Оцените потребление памяти, которое может быть вызвано загрузкой исторических данных в соответствии с Ограниченным диапазоном времени и Опциями диапазона времени. Загрузка огромного количества исторических данных может привести к отказу сервера или снижению производительности.

Имя свойства: **includeHistory**

Тип свойства: **Логическое**

ВКЛЮЧИТЬ ТЕКУЩИЕ ДАННЫЕ

Данное свойство определяет, будет ли основанный на событии[1064] или на переменной[1061] график включать данные реального времени, т.е. данные изменений, произошедших после построения графика. Если данная опция включена, график будет обновляться каждый раз, когда регистрируется новое событие или изменяется значение переменной.

Имя свойства: **includeRealtime**

Тип свойства: **Логическое**

ОГРАНИЧИТЬ ДИАПАЗОН ВРЕМЕНИ

Данное свойство относится к графикам, основанным на событии[1064] или на переменной[1061]. Поддерживаемые значения:

- **Вся история**. Все исторические значения будут загружены и отображены.
- **Последние значения**. Данные графика будут включать в себя только события и изменения значений переменных, произошедшие в определенный временной период перед построением графика, как определено свойством **Диапазон Времени**.
- **Диапазон дат**. Данные графика будут включать значения, которые получены в период времени, заданный свойствами **Начальная дата** и **Конечная дата**.

Отключение Ограниченного диапазона времени заставит сервер загружать все доступные истории событий/переменных из базы данных в память с целью подготовки графика массива данных. Это может привести к очень высокому потреблению памяти, отказу сервера и снижению производительности!

Имя свойства: **enableTimeRange**

Тип свойства: **Логическое**

ДИАПАЗОН ВРЕМЕНИ (В ПЕРИОДАХ ВРЕМЕНИ)

Данная опция доступна, если включена настройка **Ограничить диапазон времени**. Она определяет количество единиц времени[1053], которые задают временной интервал для графика.

Выставление большого диапазона заставит сервер загружать значительную часть истории событий/переменных из базы данных в память с целью подготовки графика массива данных. Это может привести к очень высокому потреблению памяти, отказу сервера и снижению производительности!

Имя свойства: **timeRange**

Тип свойства: **Целое**

НАЧАЛЬНАЯ ДАТА

Временная метка нижней границы отображенного диапазона исторических значений. Доступно если **Ограничить диапазон времени** выставлен на **Диапазон дат**.

Имя свойства: **startDate**

Тип свойства: **Дата**

КОНЕЧНАЯ ДАТА

Временная метка верхней границы отображенного диапазона исторических значений. Доступно если **Ограничить диапазон времени** выставлен на **Диапазон дат**.

Имя свойства: **endDate**

Тип свойства: **Дата**

ИСХОДНЫЕ ДАННЫЕ

Таблица исходных данных, используемая для построения графиков, основанных на пользовательских данных 1057 .

Имя свойства: **sourceData**

Тип свойства: **Таблица данных**

ПРИВЯЗКИ ИСХОДНЫХ ДАННЫХ

Набор выражений, используемых для получения данных из таблицы Исходных Данных и дальнейшего построения массива данных для графика. Данное свойство является пригодным только для графиков, основанных на пользовательских данных 1057.

Имя свойства: **chartBindings**

Тип свойства: **Таблица данных**

13.4.11.1.1 Пользовательские данные

Этот тип исходных данных поддерживается всеми графиками. Если включен массив данных, график строится в два этапа:

- Чтение исходных табличных данных графика (обычно с сервера) с использованием привязки 1295 и помещение этих данных в свойство **Исходные данные**
- Построчный анализ таблицы **Исходных данных** и заполнение массива данных графика

Построение массива данных графика в AtomMind похоже на Microsoft Excel:

- На первом этапе любые табличные данные читаются с сервера и помещаются в свойство Исходные данные. Это свойство сильно напоминает *лист* Excel, заполненный цифрами и строками, поскольку он может содержать по сути *любые* данные.
- На втором этапе данные из этого "листа" используются для заполнения полученного в результате массива данных, используя выражение 112 AtomMind, которое подобно *формуле* Excel. Например, говоря о круговой диаграмме, одно выражение ("формула") используется для расчета имени **Категории** (имени секции круговой диаграммы), а другое выражение дает **Значение** (ширину секции круговой диаграммы).

Чтение исходных данных

Сначала используется привязка 1295 виджета для заполнения таблицы исходных данных 1057. Выражение 1296 этой привязки должно вернуть таблицу данных 49, например, путем чтения значения переменной контекста или выполнения контекстной функции.

Привязка исходных данных может использоваться для периодического обновления графика или при изменениях серверных данных. Для более подробной информации о привязке **По событию** и **Периодически** обратитесь к разделу Свойства 1295.

Пример привязки исходных данных:

См. Отладка графиков 1067, чтобы понять, как заполнять **Исходные данные** "реальными" значениями из Редактора виджетов 423.

Построение массива данных

Как только свойство "Исходные данные" заполнено, график начинает перестраивать свой внутренний массив данных и обновляется. Это происходит в процессе поочередной обработки каждой записи таблицы **Исходные данные** и использования привязок графика 1057 для получения из нее различных массивов данных (таких как `X`, `Y` или `Category`).

Далее приведено пошаговое описание данного процесса:

1. Отрисовщик графика берет первую запись таблицы **исходных данных**.
2. Каждая запись таблицы **Исходные данные** используется для добавления одной записи массива данных графика. Выражения в каждом поле текущей записи **исходных данных** используются для получения различных значений из текущей записи **исходных данных** и заполнения записи массива данных графика.
3. Процесс повторяется для остальных записей таблицы

   Исходные данные, если их несколько.

См. Отладка графиков 1067, чтобы понять, как проверить график, используя "живые" значения.

Примеры

Так как вышеуказанные понятия могут показаться сложными с первого взгляда, мы предлагаем вам рассмотреть несколько примеров.

КРУГОВАЯ ДИАГРАММА

Данный простой пример показывает, как построить круговую диаграмму 1156, представляющую статистику типов сетевых интерфейсов в системе сетевого управления.

Сначала мы задаем привязку, которая выполняет запрос 829 и вводит в таблицу **Исходные данные** его результаты. Далее приведена таблица исходных данных (имена полей выделены жирным шрифтом):

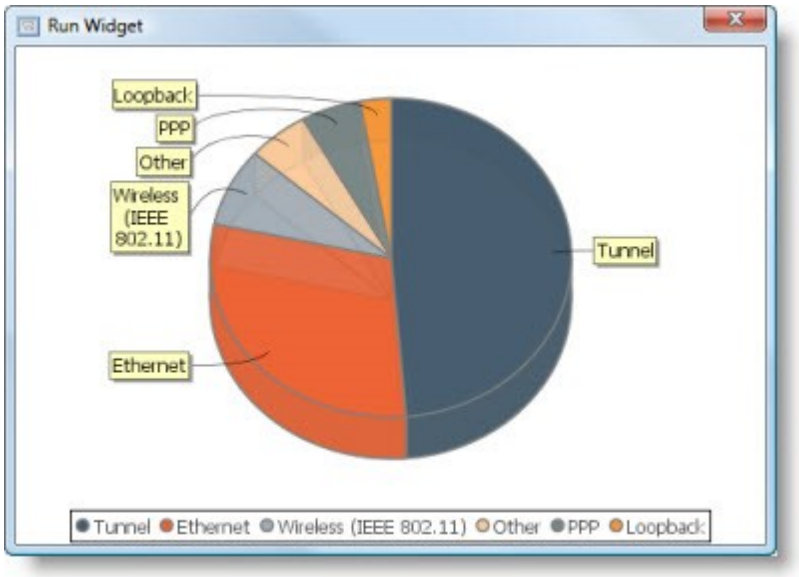
interface_type (`Interface Type`)	**interface_count** (`Interface Count`)
Tunnel	18
Ethernet	11
Wireless (IEEE 802.11)	3
Other	2
PPP	2
Loopback	1

Таблица **Привязки исходных данных** нашего графика выглядит следующим образом:

Легенда	Значение
{interface_type}	{interface_count}

Это значит, что мы будем использовать значения столбца "Тип интерфейса" для вставки легенд массива данных ("названия секторов") и значения из столбца "Количество интерфейсов" для вставки значений массива данных ("размеры секторов").

Полученный график будет выглядеть следующим образом:



СТОЛБЧАЯ ДИАГРАММА

Данный пример показывает, как можно брать значения для различных серий данных из столбцов таблицы **Исходные данные**. Данная диаграмма является столбчатой[1074] и указывает текущий входящий и исходящий трафик для 10 самых активных сетевых интерфейсов в системе сетевого управления.

Далее приведена таблица **Исходные данные** для данного графика. Она также была задана привязкой при запуске виджета.

device (Device)	**interface** (Interface)	**interfacesInterfacetrafficIncomingtraffic** (Incoming Traffic, bps)	**interfacesInterfacetrafficOutgoingtraffic** (Outgoing Traffic, bps)
admin.lh (Network Host)	Intel(R) Wireless WiFi Link 4965AGN-Native WiFi Filter Driver-0000	801	747
admin.lh (Network Host)	Intel(R) Wireless WiFi Link 4965AGN-QoS Packet Scheduler-0000	801	747
admin.lh (Network Host)	Intel(R) Wireless WiFi Link 4965AGN	801	747
admin.lh (Network Host)	Microsoft Windows Mobile Remote Adapter	11	64
admin.lh (Network Host)	Teredo Tunneling Pseudo-Interface	0	15
admin.lh (Network Host)	Intel(R) 82566MM Gigabit Platform-QoS Packet Scheduler-0000	NULL	NULL
admin.lh (Network Host)	WAN-QoS Packet Scheduler-0000	NULL	NULL

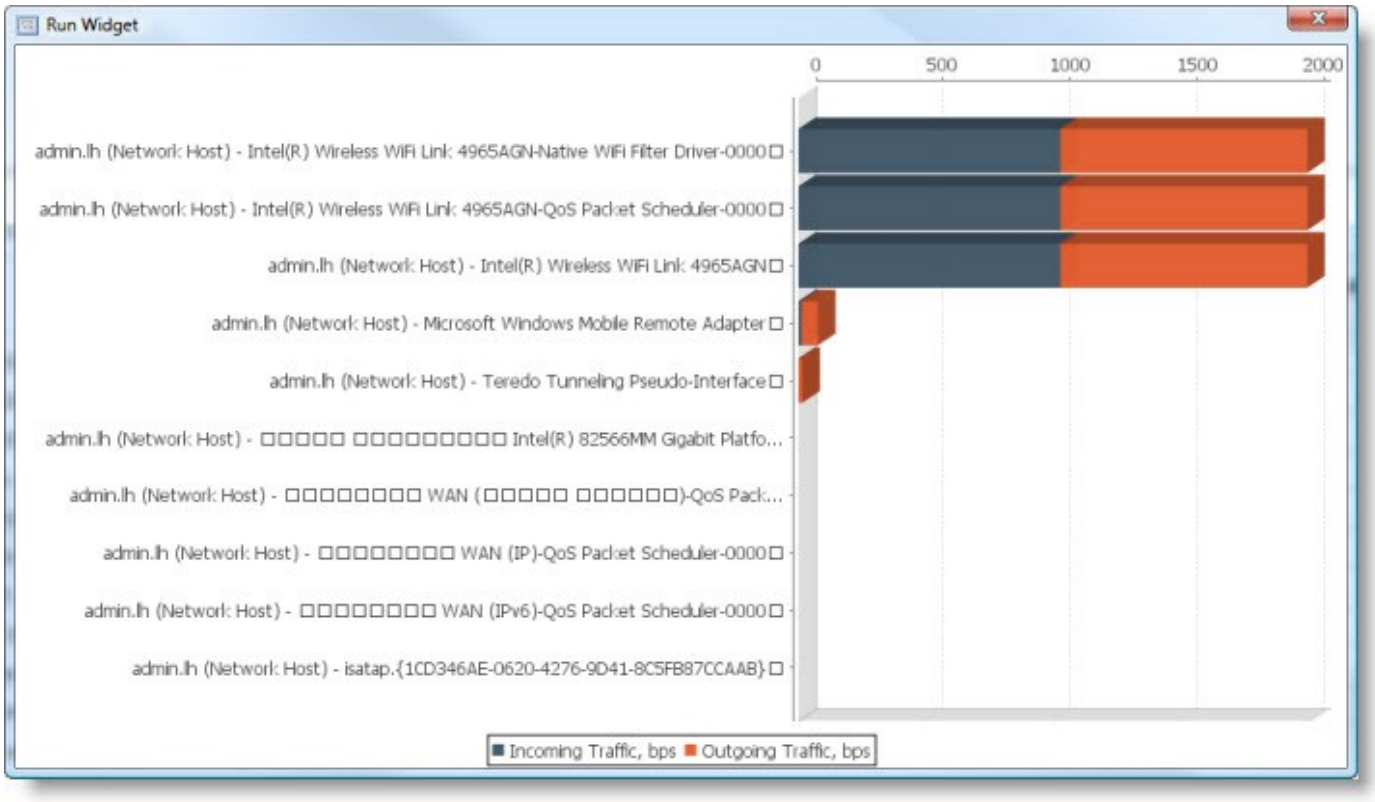
admin.lh (Network Host)	WAN (IP)-QoS Packet Scheduler-0000	NULL	NULL
admin.lh (Network Host)	WAN (IPv6)-QoS Packet Scheduler-0000	NULL	NULL
admin.lh (Network Host)	isatap.{1CD346AE-0620-4276-9D41-8C5FB87CCAAB}	NULL	NULL

Нам нужно взять значение входящего трафика из третьего столбца и значения исходящего трафика. Поэтому нам нужны две записи **привязок исходных данных**.

Легенда серии	**Категория**	**Значение**
'Incoming Traffic, bps'	{device} + ' - ' + {interface}	{interfacesInterfacetrafficIncomingtraffic}
'Outgoing Traffic, bps'	{device} + ' - ' + {interface}	{interfacesInterfacetrafficOutgoingtraffic}

Данная таблица указывает, что в графике будет две серии данных. В процессе обработки первой записи **привязки исходных данных** все добавленные записи массива данных будут принадлежать сериям с фиксированным именем `Incoming Traffic, bps` (оно цитируется как строковой литерал, т.к. все ячейки таблицы **Привязки исходных данных** содержат выражения). Во время обработки второй записи все записи массива данных будут относиться к серии `Outgoing Traffic, bps`.

Полученный график:



ПУЗЫРЬКОВАЯ ДИАГРАММА

Имейте в виду, что каждый тип графика содержит таблицу **Исходные данные** и таблицу **Привязки исходных данных** по умолчанию, что значительно облегчает понимание функциональности графика. Данный пример использует данные и привязки пузырьковой диаграммы[112] по умолчанию, чтобы продемонстрировать создание массива данных.

График **Исходные данные**:

key (Key)	**x** (X)	**y** (Y)	**z** (Z)

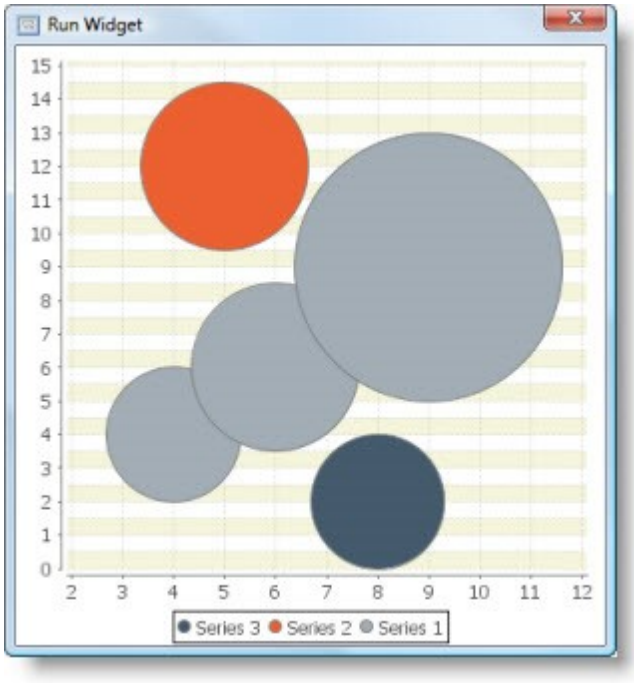
Серия 1	4	4	4
Серия 1	6	6	5
Серия 1	9	9	8
Серия 2	5	12	5
Серия 3	8	2	4

График **Привязки исходных данных**:

Легенда	**X**	**Y**	**Z**
{key}	{x}	{y}	{z}

Данные привязки очень простые: значение каждого элемента массива данных берется из поля таблицы **Исходные данные** с таким же именем. Имейте в виду, что **Z** - размер пузырьков, **X** и **Y** - их координаты.

Полученный график:



13.4.11.1.2 Диаграммы, основанные на переменной

Диаграмма может быть основана на значениях переменной 61 сервера. В зависимости от свойств компонента "Диаграмма", она может отображать историю изменений значения переменной или контролировать их в реальном времени. В данном случае диаграмма будет обновляться динамически.

Основанными на переменной могут быть следующие типы диаграмм:

- XY-диаграмма 1100
- XY-диаграмма с областями 1106
- XY-столбчатая диаграмма 1110

Диаграмма, основанная на значениях переменной, называется **временным графиком**. Он отображает изменения определенного значения с течением времени. Данное значение рассчитывается при помощи выражения 112, которое содержит ссылки на таблицу данных 49, содержащую значение переменной.

Ось X основанного на переменных графика должна быть осью дат.

Свойства серий переменных

Для более подробной информации о свойствах серий данных, основанных на переменной, см. раздел исходные переменные 1055.

13.4.11.1.2.1 Размножение серий переменных

Серии переменных имеют свойство-флажок **Размножать**, которое позволяет создавать отдельные подсерии данных для каждой записи, найденной в исходной переменной. На процесс обработки серий это влияет следующим образом:

Размножение отключено

Является настройкой по умолчанию. Будет создан только один рад данных, даже если переменная имеет несколько строк. По умолчанию **Имя** серии будет указывать на **Ряд** с индексом 0. Это номер ряда по умолчанию 119. Также можно ссылаться на другие ряды, используя ссылку формата `{field[row]}`.

Размножение включено

Создается отдельная подсерия для каждого найденного ряда переменной. **Имя** серии должно быть выражением 112. Данное выражение оценивается для каждого ряда переменной. Текущий ряд также определяется как "активный" для подсерии, чтобы во время оценки **Выражения** для исторических и будущих значений её ряд по умолчанию 119 оставался таким же.

Размножение серии будет работать только в двух случаях:

- серия основана на статистике 1063 ИЛИ
- порядок рядов в значении переменной никогда не меняется

Если выражение **Имени** серии было оценено как NULL, для текущей записи не будет создана серия. Это позволяет пропустить некоторые серии.

Далее приведен пример размноженного выражения **Имени** серии, которое пропускает создание серий для записей, у которых поля `{ifInOctets}` и `{ifOutOcters}` равны 0.

```
({ifInOctets} > 0 && {ifOutOctets} > 0) ? 'Interface ' + {ifDescr} + ' incoming traffic, Kbit/s' : null
```

13.4.11.1.2.2 Типы серий переменных

Настройка серии **Тип** полностью меняет способ, согласно которому "сырые" исходные данные конвертируются в точки графика. Существуют четыре типа серий:

- Индикатор
- Счетчик с переполнением
- Счетчик без переполнения
- Сбрасываемый

Если серия основана на статистике 1063, ее тип должен совпадать с типом статистического канала 718, на котором она основана.

Индикатор

Исходное значение сохраняется как оно есть, без какой-либо конверсии. Является настройкой по умолчанию, что удобно для применения в большинстве случаев.

Данный тип используется для измерения температуры, количества человек в комнате или биржевого курса.

Счетчик с переполнением

Подходит для продолжительных увеличивающихся счетчиков. Источник данных типа "Счетчик с переполнением" предподлагает, что счетчик никогда не уменьшается, за исключением случая переполнения счетчика. Серия учитывает данное переполнение. Счетчик показывает количество единиц изменения данных в секунду. Когда счетчик переполнен, серия проверяет, произошло ли переполнение на 32-битной или 64-битной границе и действует соответственно, добавляя подходящее значение в результат.

Проще говоря, Счетчик рассматривает разницу между предыдущим и текущим значением (дельту). Приведем пример одометра. Значение для точки на графике рассчитывается как разница показаний счетчика, поделенная на разницу по времени.

Счетчик без переполнения

Серии этого типа будут показывать производные из последнего текущего значения источника данных. Данный тип удобнее в применении для индикаторов, например, для измерения количества людей, вошедших в или покидающих комнату. По сути счетчик без переполнения работает по тому же принципу, что и счетчик с переполнением, но без проверок переполнения счетчика. Поэтому, если счетчик не сбрасывается на 32 или 64 битах, вы можете использовать этот тип и сочетать его с **минимальным значением**, равным 0.

Производный тип похож на счетчик, но в данном случае возможен обратный отсчет. Например, при мониторинге реверсивного насоса. Итоговое число может быть как отрицательным, так и положительным.

Сбрасываемый

Сбрасываемые серии используются для счетчиков, которые сбрасываются при чтении. Применяется к быстрым счетчикам, которые склонны к переполнению. Поэтому, вместо их чтения, вы делаете сброс после каждого прочтения, чтобы быть уверенным, что до следующего переполнения осталось максимум времени. Другое применение можно найти, например, для подсчета сообщений после последнего обновления.

Абсолютные серии похожи на одометр, однако, в данном случае счетчик сбрасывается каждый раз при прочтении. Значение для точки на графике рассчитывается как разница показаний счетчика, поделенная на разницу по времени.

13.4.11.1.2.3 Серии, основанные на статистике

В большинстве случаев, серии переменных основаны на "сырых" значениях переменной серии. Данные значения сохраняются в базе данных сервера.

Однако, можно построить график, данные которого основаны на статистическом канале[718].

Чтобы построить такой график:

- убедитесь, что для переменной серии существует статистический канал
- используйте *ссылку на значение статистики* в **Выражении** серии

Ссылка на значение статистики

Ссылка[117] на значение статистики имеет один из следующих форматов: `{statistics/channel}` или `{statistics/channel#key}`.

Когда в **Выражении** серии используется ссылка на значение статистики, отрисовщик берет данные истории из канала с именем `channel`. Данный канал должен быть основан на исходной переменной.

Если задан `key`(ключ), массив данных из статистики, совпадающий с данным ключем[724], будет использован в качестве источника.

На выбор данных статистики влияют следующие параметры:

- **Агрегирование**. Настройка "Агрегирование серии" совпадает с функцией статистического агрегирования[721], результаты которой будут использоваться.
- **Тип**. Тип серии должен совпадать с типом статистического канала[725].

Выражение серии может включать только одну ссылку на значение статистики. Ссылка на множество каналов невозможна.

Ссылка на другие поля из выражения серии, основанной на статистике

Выражение серии, основанной на статистике, может также ссылаться на другие поля исходной переменной. Однако такие ссылки имеют несколько важных нюансов:

- Статистический канал не сохраняет изначальные значения истории переменной. Таким образом, любая ссылка на ячейки исходной переменной преобразуются в **текущее** значение данной переменной.

- Если включено ведение истории в реальном времени для серий, основанных на статистике, отрисовщик графика объединяет выражение статистического канала с выражением серии, чтобы сформировать новое выражение для построения точек графика в реальном времени.

13.4.11.1.3 Диаграммы, основанные на событии

Диаграмма может основываться на данных событий 73 сервера. В зависимости от свойств компонента "Диаграмма", она может использовать события реального времени и/или историю событий 73 в качестве исходных данных. При использовании событий реального времени диаграмма обновляется динамически.

Быть основанными на событии могут следующие типы графиков:

- XY-диаграмма 1100
- XY-диаграмма с областями 1106
- XY-столбчатая диаграмма 1110

Диаграмма, основанная на данных события, называется **временным графиком**. Он отображает изменения определенного значения с течением времени. Данное значение рассчитывается при помощи выражения 112, которое содержит ссылки на данные события 73.

Ось X основанной на событии диаграммы должна быть осью дат.

Более подробную информацию см. в разделе исходные события 1053.

13.4.11.1.3.1 Зависимые графики

Диаграмма может быть основана на наборе данных другого графика. Использование зависимого графика подразумевает наличие как минимум двух диаграмм: исходной (ее данные используются для построения зависимой) и самой зависимой диаграммы. Поэтому зависимый график может быть создан только как часть XY-диаграммы с общей осью параметров 1143 (или XY-Диаграммы с общей осью значений 1146). Следующие диаграммы могут быть зависимыми:

- XY-диаграмма 1100
- XY-диаграмма с областями 1106
- XY-столбчатая диаграмма 1110

Настройка зависимых графиков

Для конфигурации нового зависимого графика выполните следующие действия:

1. Создайте XY-диаграмму с общей осью параметров 1143 (или XY-диаграмму с общей осью значений 1146).
2. Добавьте 1160 к ней простую XY-диаграмму.
3. Сконфигурируйте источник данных для XY-диаграммы и убедитесь, что она отображает правильные значения.
4. Добавьте 1160 другую XY-диаграмму. Вторая XY-диаграмма будет зависеть от исходных данных первой XY-диаграммы.
5. Измените тип источника данных 1053 второй диаграммы на **Зависимый**.
6. Откройте свойства **Зависимого графика** второй диаграммы и измените **Имя исходного графика** на имя компонента виджета, соответствующего исходной XY-диаграмме, например `xyLineChart1`.
7. Выберите **Привязки зависимого графика** и сконфигурируйте выражения для обработки исходных данных.
8. Запустите виджет и проверьте зависимый график.

Массив данных

Построение набора данных зависимого графика выполняется путем обработки набора данных исходного графика с помощью **Привязок зависимого графика**. Ниже дано краткое описание этого процесса:

1. Отрисовщик графиков берет первую серию из массива данных исходного графика.

2. Привязки из таблицы **Привязки зависимого графика** применяются к значениям этой серии. Обратите внимание, что привязка применяется только в том случае, если ее **Индекс серии** согласуется с индексом текущей серии. Каждое применение привязки к серии порождает новую серию в конечном массиве данных.
3. Этот же процесс повторяется для оставшихся серий массива данных исходного графика.

Ниже представлена таблица **Привязок зависимого графика**:

Привязка	Ожидаемый тип значения	Описание
Серия	Строка	Текстовое имя серии данных.
X	Число	Числовое значение, отображаемое вдоль оси определений (X).
Y	Число	Числовое значение, отображаемое вдоль оси измерений (Y).
Индекс серии	Целое	Индекс серии, к которой применятся привязка. Если не указан, привязка применяется к каждой серии.

Свойства зависимых графиков

Свойства зависимых графиков доступны только для XY диаграммы 1100, XY-диаграммы с областями 1106 и XY-столбчатой диаграммы 1110, когда они добавлены в качестве подграфиков к XY-диаграмме с общей осью параметров 1143 (или XY-диаграмме с общей осью значений 1146).

ИМЯ ИСХОДНОГО ГРАФИКА

Имя компонента подграфика, чьи данные будут использованы для построения зависимого графика. Исходный и зависимый графики быть размещены в одной и той же XY-диаграмме с общей осью параметров 1143 (или XY-диаграмме с общей осью значений 1146).

Имя свойства: **sourceDataPlotID**

Тип свойства: **Строка**

ПРИВЯЗКИ ЗАВИСИМОГО ГРАФИКА

Набор выражений, используемых для извлечения данных из исходного графика и создания массива данных зависимого графика. Это свойство действительно только для зависимых графиков.

Имя свойства: **dependentChartBindings**

Тип свойства: **Таблица данных**

13.4.11.1.4 Тренды

Диаграмма может отображать тренды, основанные на данных из массива данных другого графика. Выделяют четыре типа трендов:

- Линейный
- Экспоненциальный
- Скользящее среднее
- Процентиль

Определение тренда включает две диаграммы для представления: исходная диаграмма (его данные используются для расчета тренда) и сам тренд. Таким образом, тренды могут быть размещены на рабочую область следующими графиками, когда они являются частью смешанной XY-диаграммы 1139:

- XY-диаграмма 1100
- XY-диаграмма с областями 1106
- XY-столбчатая диаграмма 1110

Установка трендов

Для конфигурации нового тренда для графика выполните следующие действия:

1. Создайте смешанную XY-диаграмму 1139
2. Добавьте 1160 к ней простую XY-диаграмму.

3. Сконфигурируйте источник данных для XY-диаграммы и убедитесь, что она отображает правильные значения.
4. Добавьте 1160 другую XY-диаграмму. Вторая диаграмма покажет тренд(ы) исходных данных первой диаграммы.
5. Измените тип источника данных 1053 второй диаграммы на **Трендинг**.
6. Так как график тренда и исходный график находятся в одном адресном пространстве и имеют общие оси, вы теперь можете удалить оси диаграммы тренда из смешанного графика-родителя. Данный шаг является опциональным.
7. Откройте вкладку "Трендинг" второго координатного графика и измените **Имя исходного графика** на имя компонента виджета, соответствующего изначальной XY-диаграмме, например, `xyLineChart1`.
8. Выберите тип тренда и сконфигурируйте другие свойства тренда.
9. Запустите виджет и проверьте тренд.

Свойства определений тренда

Свойства определений тренда доступны только для XY-диаграммы 1100, XY-диаграммы с областями 1106, XY-столбчатой диаграммы 1110, когда они добавляются в качестве подграфиков к смешанной XY-диаграмме 1139.

ИМЯ ИСХОДНОГО ГРАФИКА

Имя компонента подграфика, данные которого будут использоваться для построения кривых тренда. Должен являться подграфиком той же смешанной XY-диаграммы.

Имя свойства: **sourceDataPlotID**

Тип свойства: **String**

ВИД ТРЕНДИНГА

Тип тренда(трендов) для построения: **Линейный**, **Экспоненциальный**, **Скользящее среднее** или **Процентиль.**

Имя свойства: **trendingType**

Тип свойства: **Integer**

ФАКТОР ПРОГНОЗА

Длина "будущего" тренда по времени в сравнении с временной рамкой исходных данных, например:

- Значение 0.0 означает, что тренд построен специально для временной рамки, покрываемой исходными данными.
- Значение 0.5 означает, что тренд уходит в будущее на половину временной рамки исходных данных. Например, если исходные данные покрываю весь 2012 год, тренд будет покрывать 2012 и две четверти 2013.
- Значение 1.0 означает, что тренд уходит в будущее на полную временную рамку исходных данных. Например, если исходные данные покрываю весь 2012 год, тренд будет покрывать весь 2012 и 2013.
- Значение 3.0 означает, что тренд уходит в будущее на три временных рамки источника и т.д.

Имя свойства: **forecastFactor**

Тип свойства: **Float**

Линейная и экспоненциальная опции тредов

КОЛИЧЕСТВО ТОЧЕК

Количество элементов данных в кривой тренда.

Нулевое значение означает, что количество точек будет совпадать с количеством элементов данных в исходной серии.

Данная опция пригодна только для линейного и экспоненциального трендов.

Имя свойства: **pointCount**

Тип свойства: **Float**

Опции тренда "Скользящее среднее"

ПЕРИОД

Период осреднения для расчета скользящего среднего (в диапазоне данных диаграммы).

Имя свойства: **period**

Тип свойства: **Float**

НАЧАЛЬНАЯ ТОЧКА

Точка начального периода для расчета скользящего среднего (в диапазоне данных диаграммы).

Имя свойства: **skipPeriod**

Тип свойства: **Float**

Опции тренда "Процентиль"

ПРОЦЕНТИЛЬ

Основа тренда "Процентиль" в процентах. Значение по умолчанию - 95%, тренд будет отображен так, что 95% исходных значений останутся ниже линии тренда.

Имя свойства: **percentile**

Тип свойства: **Float**

13.4.11.2 Отладка диаграмм

Когда новая диаграмма создается в редакторе виджетов 423, она содержит определенные ранее таблицу Исходные данные 1057 и Привязки исходных данных 1057, которые ссылаются на эти исходные данные по умолчанию. Вместе они образуют нечто наподобие "демо графика", представляя его общий вид.

Как только вы начали настраивать свой график, вы изменяете его привязки исходных данных так, чтобы они ссылались на ваши данные. Во время нормальной работы виджета и в режиме предварительного просмотра редактора таблица исходных данных изменяется привязкой, и затем по этим данным строится массив данных графика.

Однако во время работы в редакторе система не получает данные от сервера. Таким образом, как только вы изменили привязки исходных данных, система больше не в состоянии построить график, т.к. таблица "Исходные данные" по умолчанию не совпадает с "настоящими" привязками.

Чтобы устранит эту проблему, контекстное меню графика в редакторе виджетов содержит пункт **Загрузить реальные исходные данные**. Данная операция приводит к тому, что система находит привязку виджета 1295, указывающую на текущие исходные данные графика, пересчитывает ее выражение, получая данные сервера, и записывает эти данные в таблицу "Исходные данные" в "Режиме редактора". После этого вы сможете настроить график, который использует текущие данные.

Чтобы вернуть изначальное демо-значение исходных данных, воспользуйтесь пунктом контекстного меню **Возврат исходных данных**.

13.4.11.3 Графики категорий

Основной чертой всех графиков категорий является их ось параметров, которая отображает разные текстовые *категории*, в то время как ось значений представляет числовые значения.

Все графики категорий основаны на:

- области построения 1187
- отрисовщике категорий 1228

13.4.11.3.1 Диаграмма

Диаграмма просто соединяет каждый элемент данных (категорию, значение) при помощи прямых линий.

Диаграмма основана на области построения категорий 1187 и отрисовщике категорий 1228 и наследует все их свойства.

Диаграмма выглядит следующим образом:

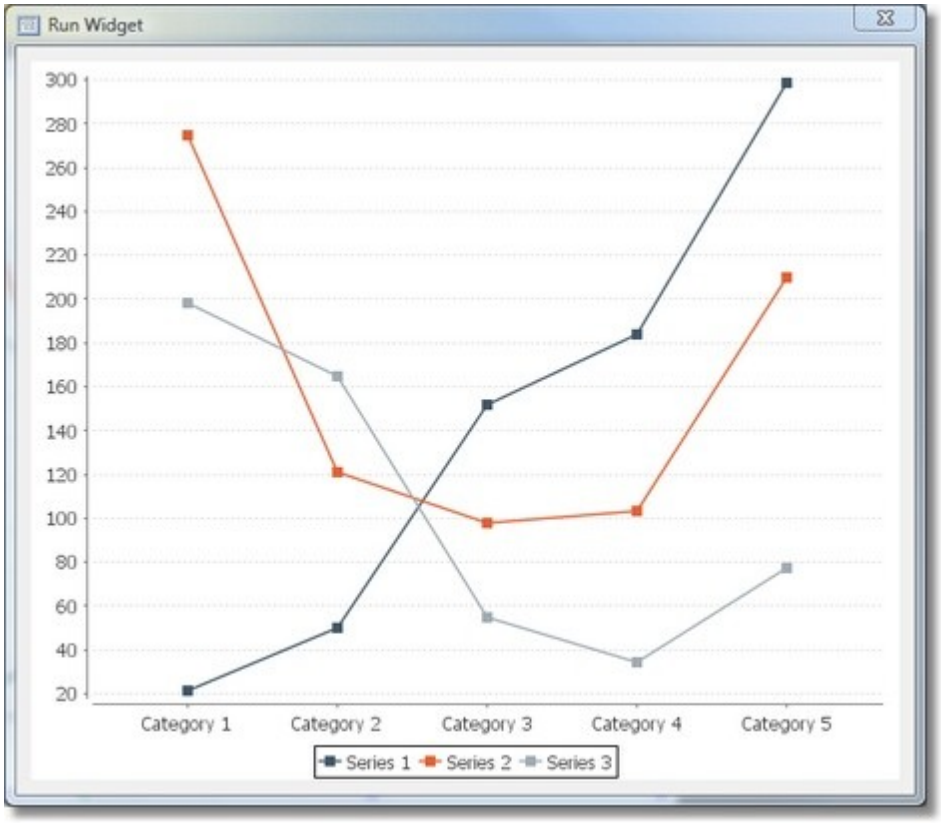
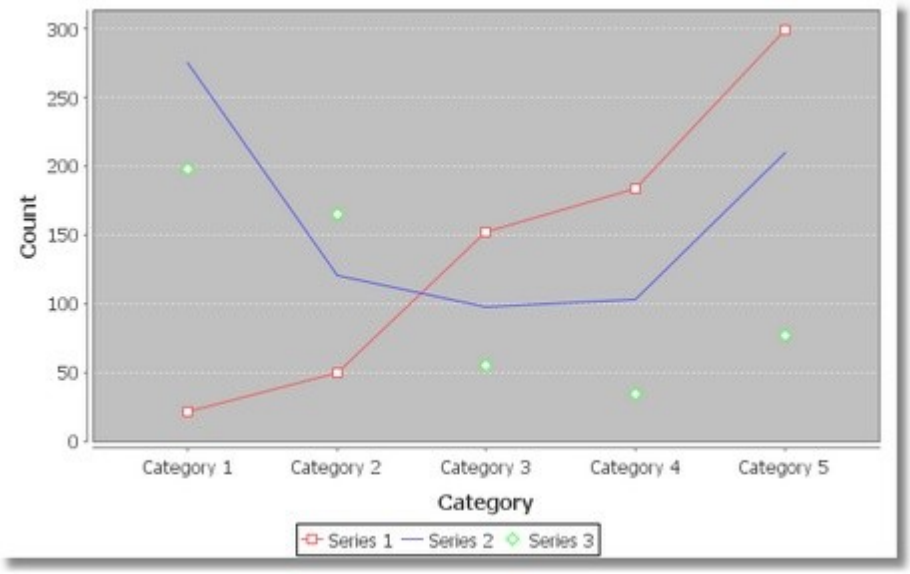


Диаграмма поддерживает четыре вида отрисовщика.

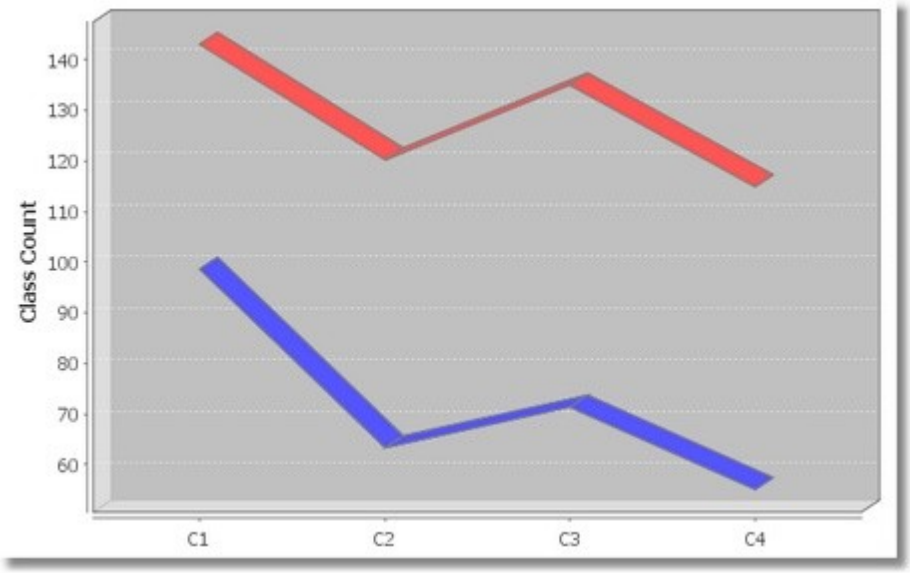
ЛИНЕЙНЫЙ ОТРИСОВЩИК

Отрисовщик, который отображает элементы данных путем соединения точек графика с прямыми линиями.



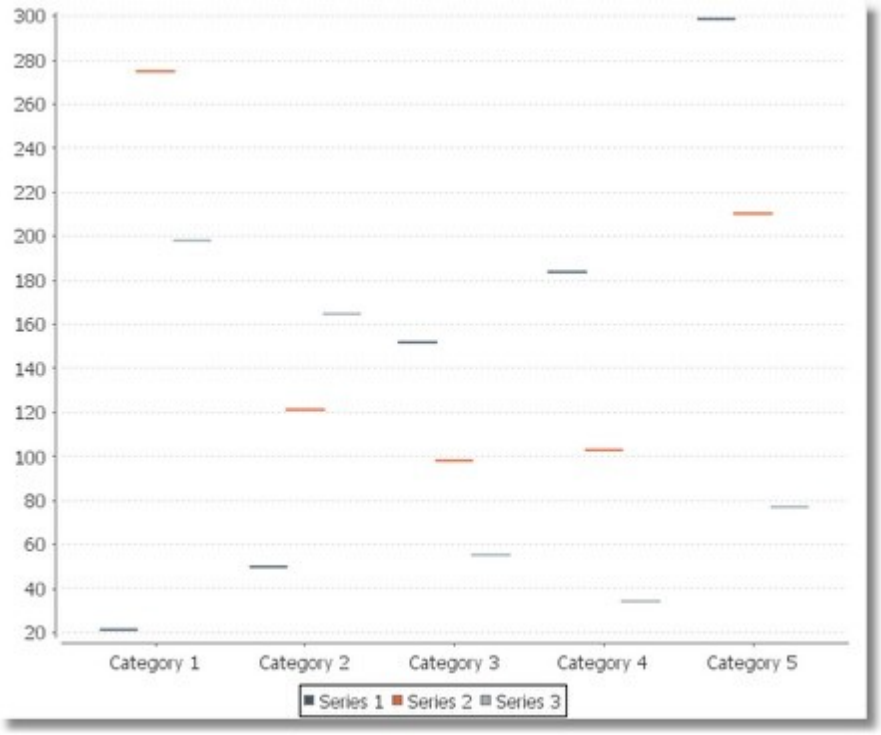
ЛИНЕНЫЙ С 3D

Линейный отрисовщик, который использует "псевдо-3D" эффект для отображения линий графика.



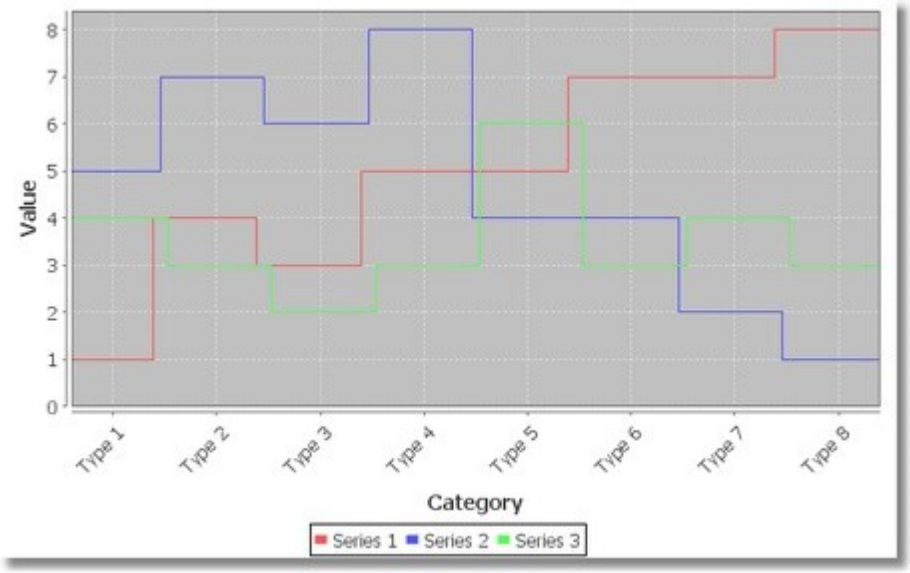
УРОВНЕВЫЙ ОТРИСОВЩИК

Отрисовщик, который чертит горизонтальные линии для отображения элементов.



СТУПЕНЧАТЫЙ ОТРИСОВЩИК

Отрисовщик, который соединяет элементы данных при помощи ступенчатой линии.



Массив данных

Линейный график поддерживает только модель Пользовательские данные 1057.

Он имеет следующие привязки исходных данных:

Привязка	Тип ожидаемого значения	Описание
Серия	строка	Текстовое имя серии данных. Серия данных представлена одной линией (или набором фигур) на графике.
Категория	строка	Имя категории. Категории отображаются на оси определений.
Значение	число	Числовое значение, отображаемое для вышеуказанных серии/категории. Значения показаны вдоль оси измерений.

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Непрозрачный 1275, Фон 1275, Рамка 1275

Все общие свойства графика 1160.

Свойства Исходные данные 1057 и Привязки исходных данных 1057.

Все свойства области построения категорий 1187.

Все свойства отрисовщика 1228.

Все свойства, определяемые напрямую отрисовщиком категорий 1232 актуальны для Диаграмм и 3D Диаграмм. Эти свойства не имею эффекта на уровневые и ступенчатые отрисовщики.

Пользовательские свойства

ГРАНИЦА ЭЛЕМЕНТОВ

Граница элементов в виде процентного отношения ко всей длине оси категорий (значение по умолчанию - 0.20 или двадцать процентов). Контролирует пространство, предназначенное для промежутков между элементами внутри одной категорий.

Данное свойство пригодно для отрисовщика уровня.

Имя свойства: **itemMargin**

Тип свойства: **Плавающее**

МАКСИМАЛЬНАЯ ШИРИНА ЭЛЕМЕНТОВ

Максимальная ширина элемента в виде процентного отношения от длины оси. Например, установив значение на 0.05, вы можете быть уверены, что элемент не превысит длину линии оси более, чем на пять процентов. Это способствует улучшению вида графиков, в которых, возможно, будет отображены один или два элемента.

Данное свойство пригодно для отрисовки уровня.

Имя свойства: **maximumItemWidth**

Тип свойства:**Плавающее**

3D X-СМЕЩЕНИЕ

Смещение по оси X для эффекта 3D.

Данное свойство пригодно для отрисовки 3D диаграммы.

Имя свойства: **XOffset**

Тип свойства: **Плавающее**

3D Y-СМЕЩЕНИЕ

Смещение по оси Y для эффекта 3D.

Данное свойство пригодно для отрисовки 3D диаграммы.

Имя свойства: **YOffset**

Тип свойства: **Плавающее**

ОКРАСКА СТЕН

Цвет[1279], используемый для заливки фона "сторон" (или "стен") области построения графика.

Данное свойство пригодно для 3D диаграммы.

Имя свойства: **wallPaint**

Тип свойства: **Таблица данных**

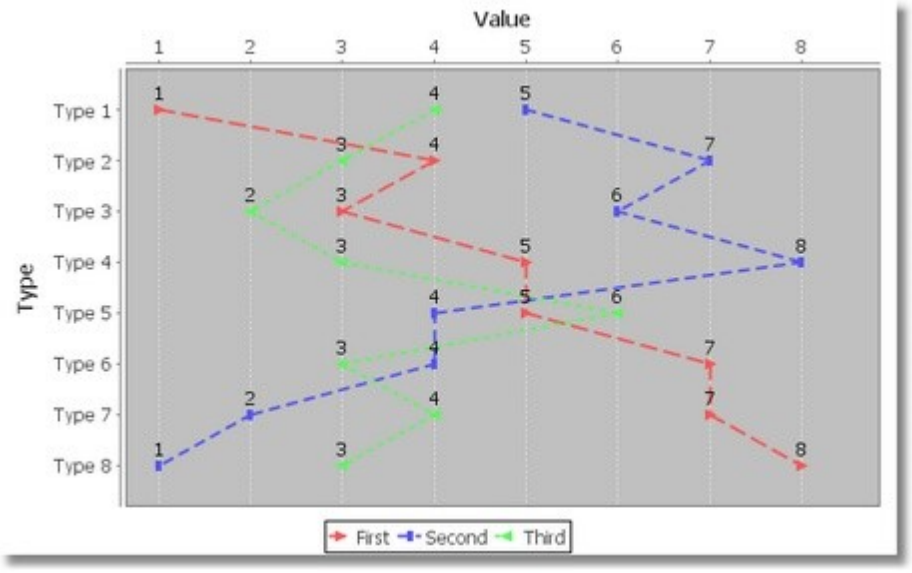
Общие события

Скрытие[1285], Показ[1285], Перемещение[1285], Изменение размеров[1285], Клик мыши[1286], Нажатие кнопки мыши[1286], Отпускание кнопки мыши[1286], Вход мыши[1286], Выход мыши[1286], Перемещение мыши[1286], Вращение колесика мыши[1286], Печать клавиши[1287], Нажатие клавиши[1287], Отпускание клавиши[1287], Получение фокуса[1287], Потеря фокуса[1287]

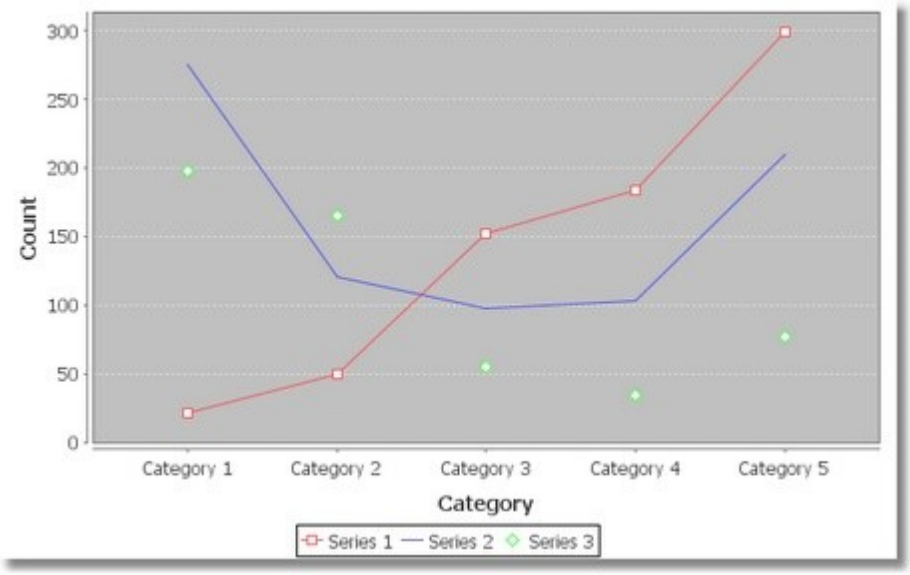
Все соответствующие события графика[1182].

Дополнительные примеры

Диаграмма с метками элементов. Ориентация области построения была изменена на горизонтальную:



Диаграмма, на которой каждая серия отображается через комбинацию линий и/или фигур:



13.4.11.3.2 Диаграмма с областями

Диаграмма с областями представляет заполненную область между осью определений и точками данных.

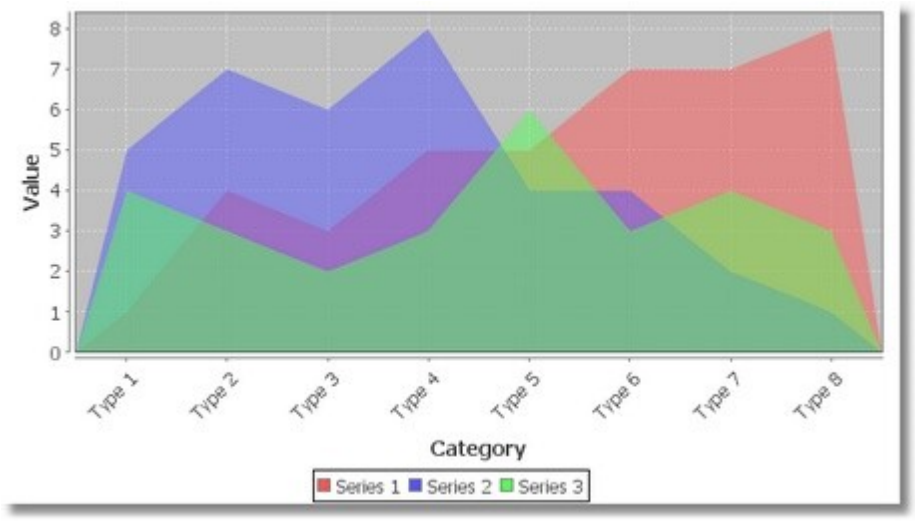
Диаграмма с областями основана на области построения категорий[1187] и отрисовщике категорий[1228]. Она наследует все их свойства.

Диаграмма с областями выглядит следующим образом:

Диаграмма с областями поддерживает два отрисовщика.

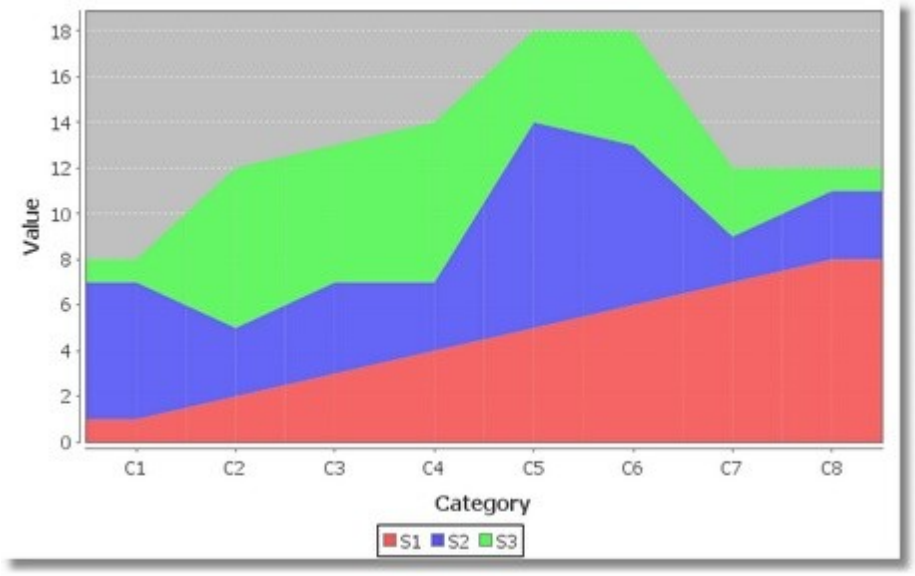
ОТРИСОВЩИК С ОБЛАСТЬЮ

Отрисовщик, который размещает на графике различные серии данных одну поверх другой. Поскольку серии перекрывают друг друга, они отображаются полупрозрачно. Прозрачность контролируется свойством "Прозрачность" графика[1187].



СОСТАВНОЙ ОТРИСОВЩИК

Отрисовщик, который размещает серии данных друг над другом.



Массив данных

График с областями поддерживает только модель Пользовательские данные[1057].

Данный график имеет следующие привязки исходных данных:

Привязка	Тип ожидаемого значения	Описание
Серия	строка	Текстовое имя серии данных.
Категория	строка	Имя категории. Категории отображаются на оси определений.
Значение	число	Числовое значение, отображаемое для этой серии/категории. Значения показаны вдоль оси измерений.

Общие свойства

Ширина[1274], Высота[1274], Привязки[1275], Видимый[1275], Непрозрачный[1275], Фон[1275], Рамка[1275]

Все общие свойства графика[1160].

Свойства Исходные данные 1057 и Привязки исходных данных 1057.

Все свойства области построения категорий 1187.

Все свойства отрисовщика 1228.

Пользовательские свойства

ТИП КРАЕВ

Контролирует изображение конечных точек на графике. Существуют три варианта:

- **Скошенный**: сужение до нуля.
- **Обрезанный**: срез около первых и последних значений.
- **Уровневый**: заполняет края уровня графика первыми и последними значениями данных.

Имя свойства: **endType**

Тип свойства: **строка**

ОТОБРАЖАТЬ В ВИДЕ ПРОЦЕНТОВ

Свойство-флажок, контролирующее отображение отрисовщиком каждого значения элемента в процентах (так, чтобы добавление стеков областей доходило до 100 %).

Данное свойство пригодно для составного отрисовщика.

Имя свойства: **renderAsPercentages**

Тип свойства: **логическое**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

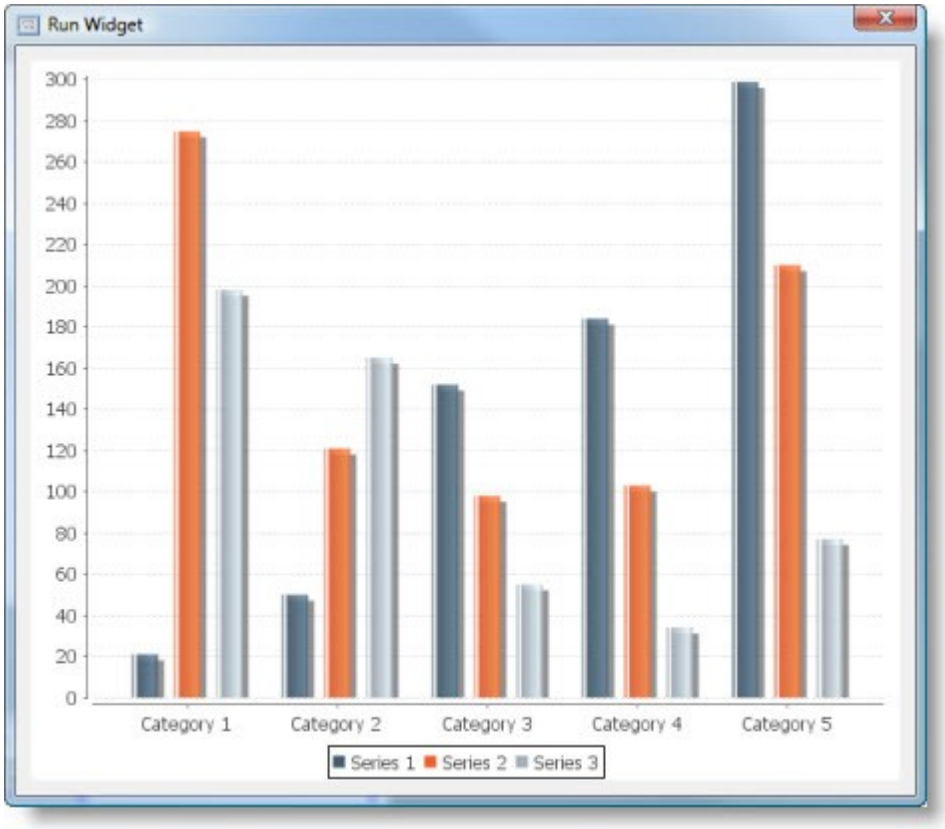
Все соответствующие события графика 1182.

13.4.11.3.3 Столбчатая диаграмма

Столбчатая диаграмма представляет элементы данных в виде прямоугольных столбцов.

Столбчатая диаграмма основана на области построения категорий 1187 и отрисовщике столбцов категорий 1229. Он наследует все их свойства.

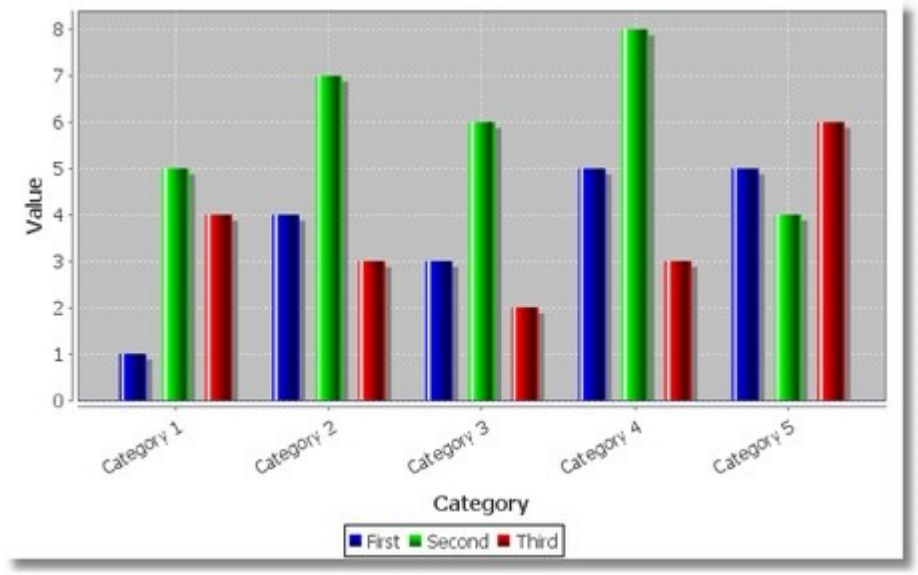
Столбчатая диаграмма выглядит следующим образом:



Столбчатая диаграмма поддерживает шесть отрисовщиков.

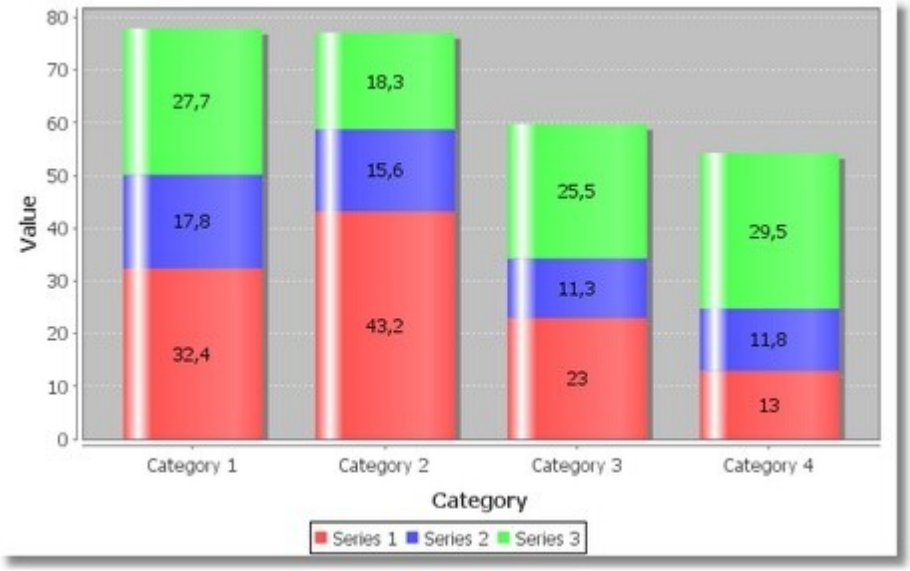
СТОЛБЧАТЫЙ ОТРИСОВЩИК

Классический отрисовщик столбцов, который рисует столбцы для каждой пары серии/категории в виде отдельной "группы столбцов".



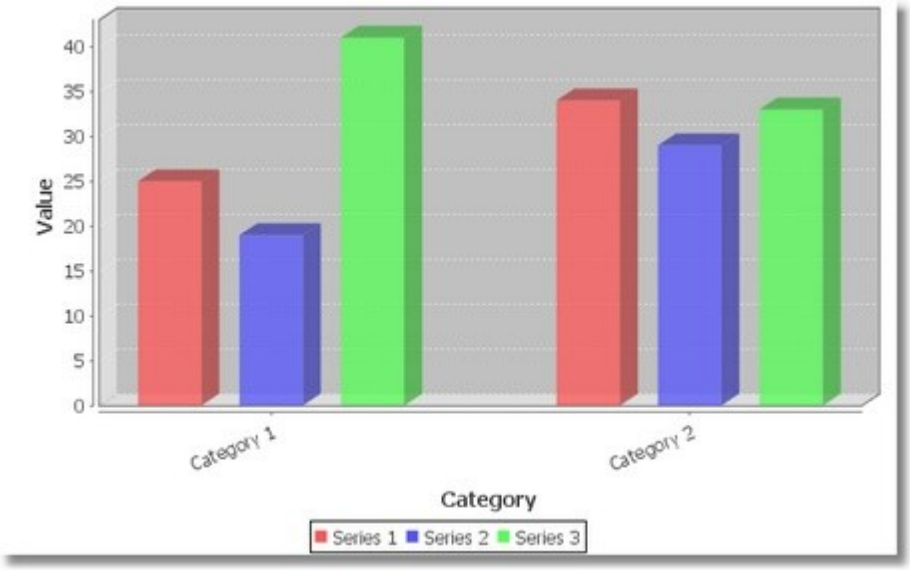
СОСТАВНОЙ ОТРИСОВЩИК

Отрисовщик, который отображает столбцы различных серий друг над другом (стопкой) в каждой категории.



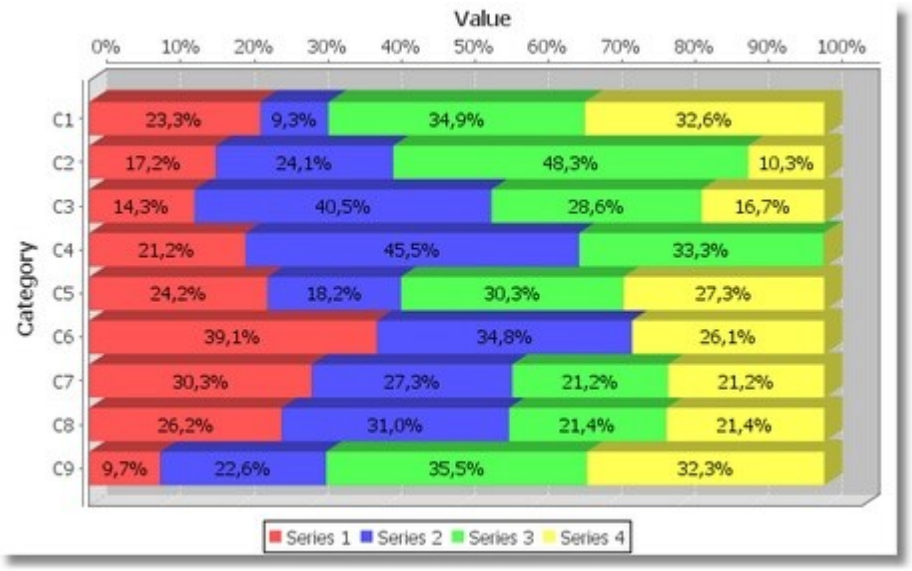
СТОЛБЧАТЫЙ 3D ОТРИСОВЩИК

Отрисовщик, отображающий элементы в 3D эффекте.



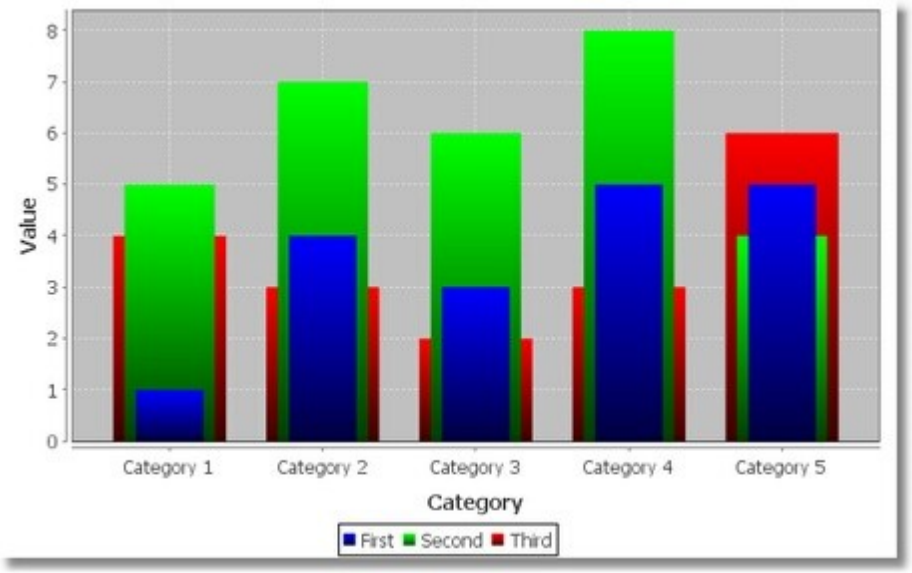
СОСТАВНОЙ 3D ОТРИСОВЩИК

Отрисовщик, отображающий столбцы стопкой с эффектом 3D.



МНОГОСЛОЙНЫЙ СТОЛБЧАТЫЙ ОТРИСОВЩИК

Отрисовщик, который отображает столбцы для каждой пары серий/категорий один за другим.

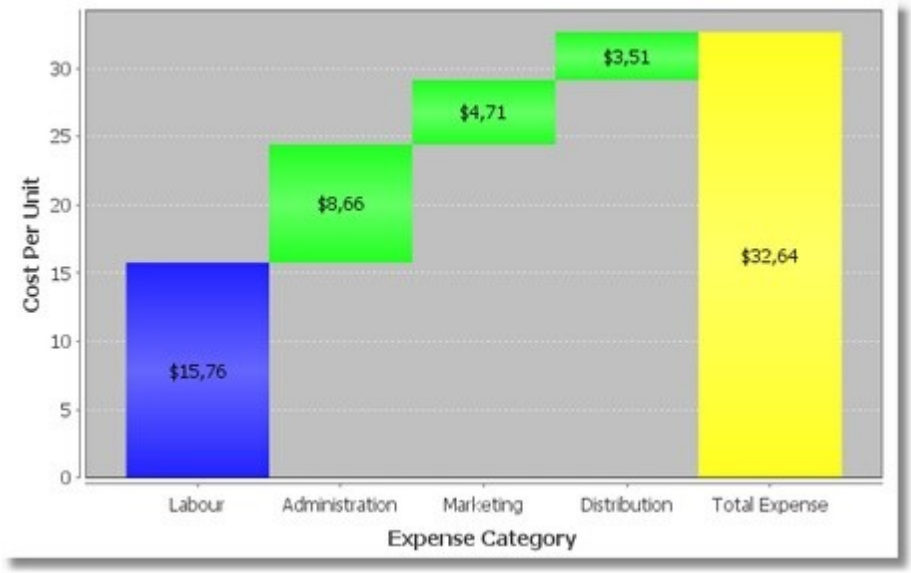


ВОДОПАДНЫЙ СТОЛБЧАТЫЙ ОТРИСОВЩИК

Отрисовщик для изображения графиков "водопадом". График "водопад" выделяет различие между двумя значениями и компонентами, которые и создают это различие.

Значение в последней категории массива данных должно быть определено в виде суммы элементов в предыдущих категориях - иначе это приведет к тому, что последний столбец будет отображен некорректно в области построения.

Цвета столбцов для данного отрисовщика определяются при помощи специальных свойств. Относительные свойства, наследованные из основного отрисовщика, игнорируются.



Массив данных

Столбчатый график поддерживает только модель Пользовательские данные 1057.

Он имеет следующие привязки исходных данных:

Привязка	Тип ожидаемого значения	Описание
Серия	строка	Текстовое имя серии данных.
Категория	строка	Имя категории. Категории отображаются вдоль оси определения.
Значение	число	Отображение числового значения для вышеуказанных серии/категории. Значения отображаются вдоль оси измерения.

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Непрозрачный 1275, Фон 1275, Граница 1275

Все общие свойства графика 1160.

Свойства Исходные данные 1057 и Привязки исходных данных 1057.

Все свойства области построения категорий 1187.

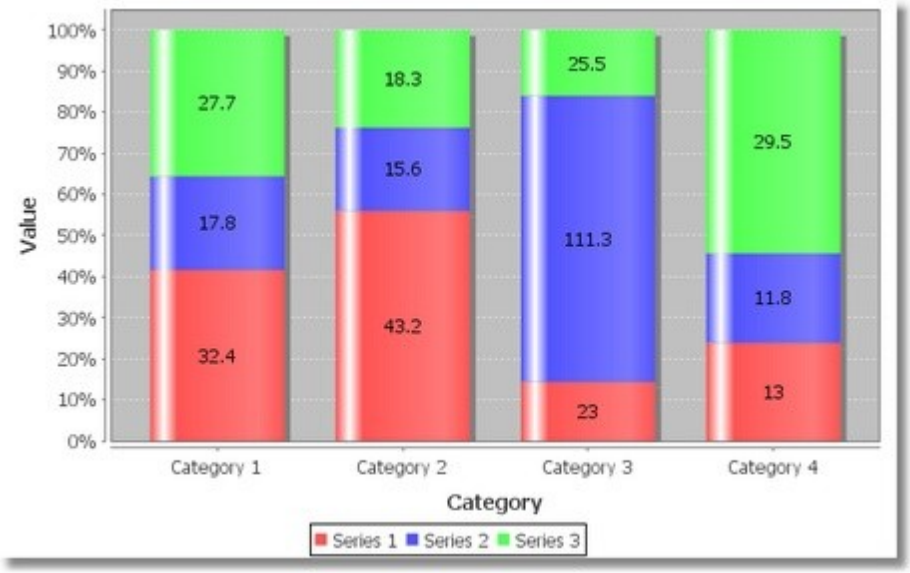
Все свойства отрисовщика категорий столбцов 1229.

Пользовательские свойства

ОТОБРАЖАТЬ В ВИДЕ ПРОЦЕНТОВ

Флажок, контролирующий отображение каждого значения элемента в виде процентов (так, чтобы столбцы доходили до 100 %).

Пример столбчатого составного графика, отображающего значения в процентах:



Данное свойство пригодно для столбчатой и составной (3D) диаграмм.

Имя свойства: **renderAsPercentages**

Тип свойства: **логическое**

3D X-СМЕЩЕНИЕ

Смещение X для эффекта 3D.

Данное свойство пригодно для столбчатой и составной (3D) диаграмм.

Имя свойства: **XOffset**

Тип свойства: **плавающее**

3D Y-СМЕЩЕНИЕ

Смещение Y для эффекта 3D.

Данное свойство пригодно для столбчатой и составной (3D) диаграмм.

Имя свойства: **YOffset**

Тип свойства: **плавающее**

ОКРАСКА СТЕН

Цвет[1279], используемый для заливки "сторон" (или "стен") фона области построения графика.

Данное свойство пригодно для столбчатой и составной (3D) диаграмм.

Имя свойства: **wallPaint**

Тип свойства: **таблица данных**

ОКРАСКА ПЕРВОГО СТОЛБЦА

Цвет заливки[1279] первого столбца (т.е. первой категории).

Имя свойства: **firstBarPaint**

Тип свойства: **таблица данных**

ОКРАСКА ПОСЛЕДНЕГО СТОЛБЦА

Цвет заливки[1279] последнего столбца (т.е. последней категории).

Имя свойства: **lastBarPaint**

Тип свойства: **таблица данных**

ОКРАСКА ПОЛОЖИТЕЛЬНЫХ СТОЛБЦОВ

Цвет заливки[1279] промежуточных столбцов, представляющих положительные значения.

Имя свойства: **positiveBarPaint**

Тип свойства: **таблица данных**

ОКРАСКА ОТРИЦАТЕЛЬНЫХ СТОЛБЦОВ

Цвет заливки 1279 промежуточных столбцов, представляющих отрицательные значения.

Имя свойства: **negativeBarPaint**

Тип свойства: **таблица данных**

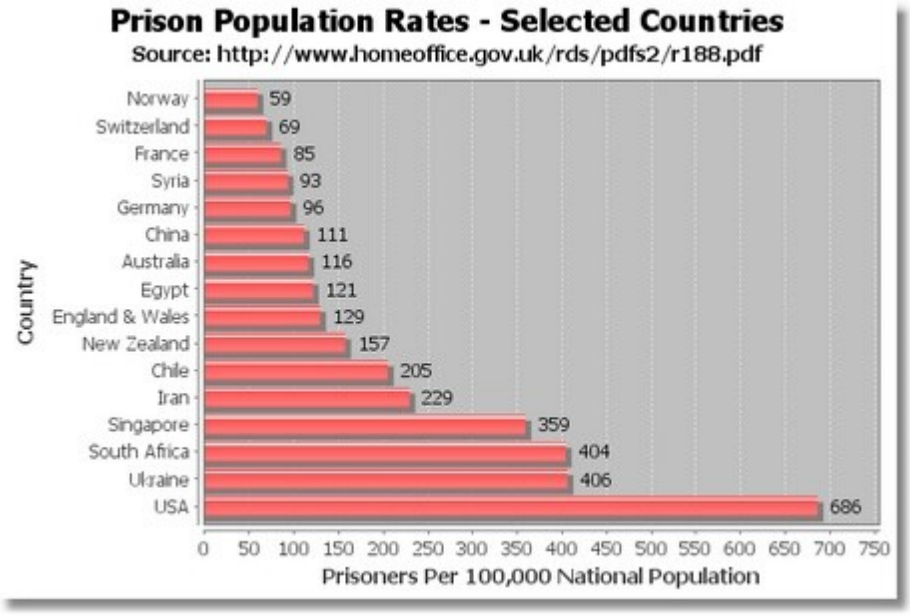
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

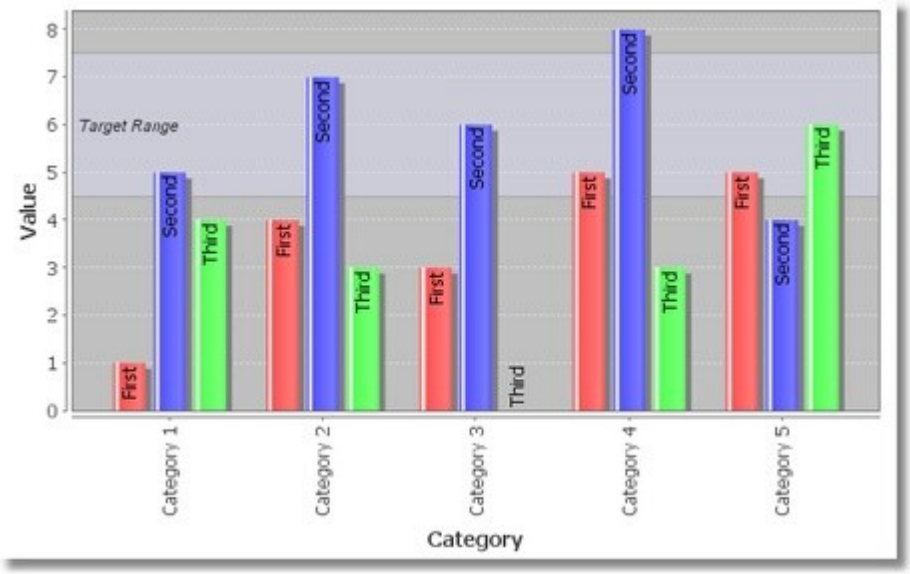
Все соответствующие события графика 1182.

Дополнительные примеры

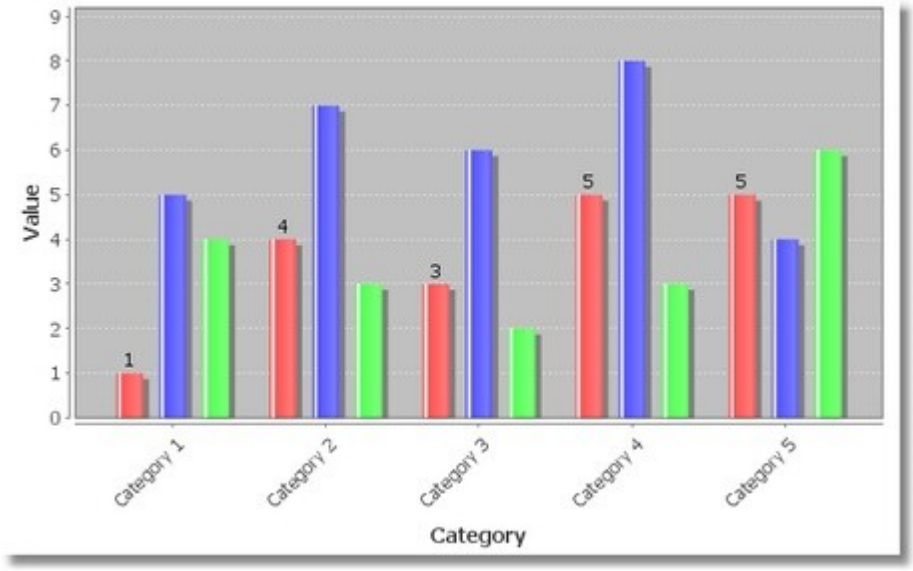
Столбчатая диаграмма с горизонтальной ориентацией:



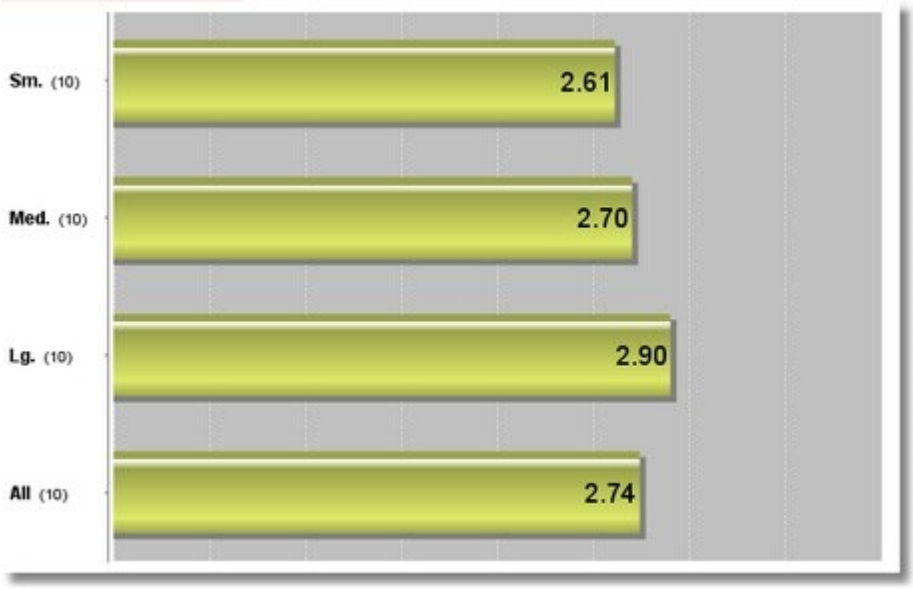
Столбчатая диаграмма с созданной пользователем **позицией метки элемента по умолчанию**. В график была добавлена метка измерения для указания целевого диапазона значений данных:



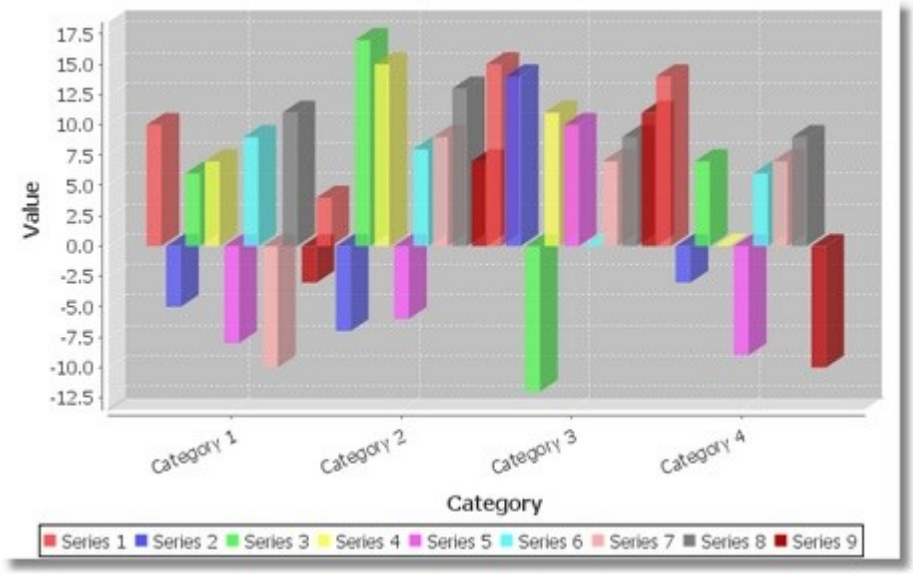
Столбчатая диаграмма с метками элементов только для одной серии:



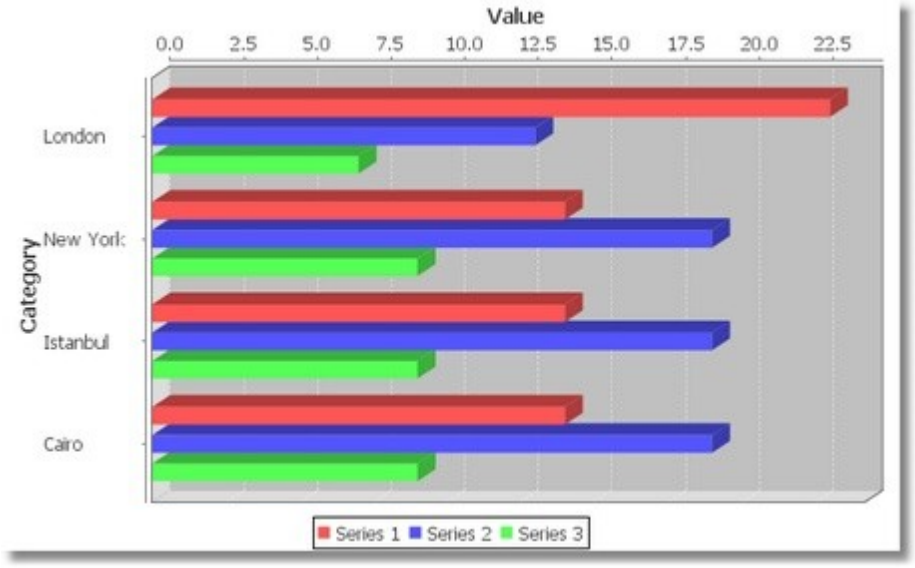
Другая пользовательская настройка меток:



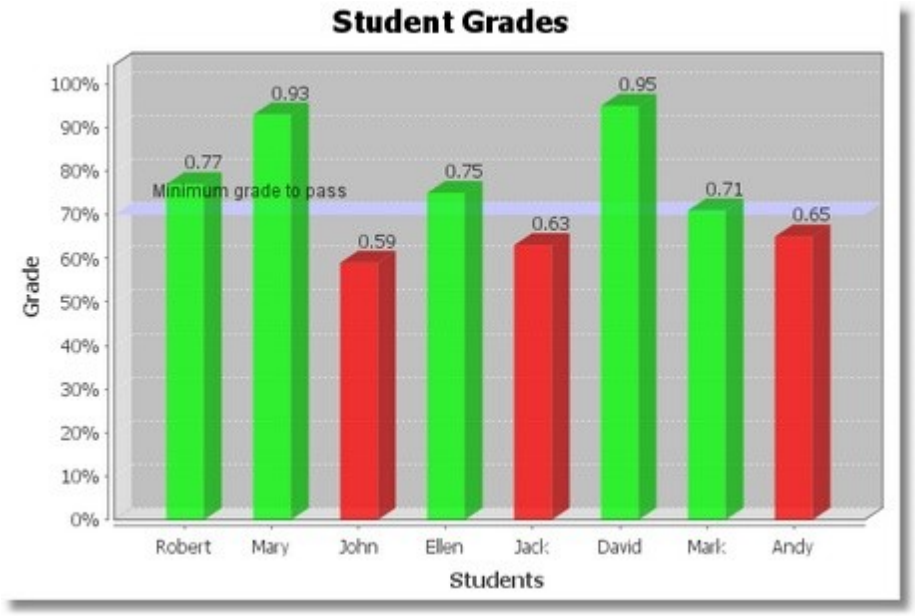
Столбчатая 3D диаграмма с множеством серий:



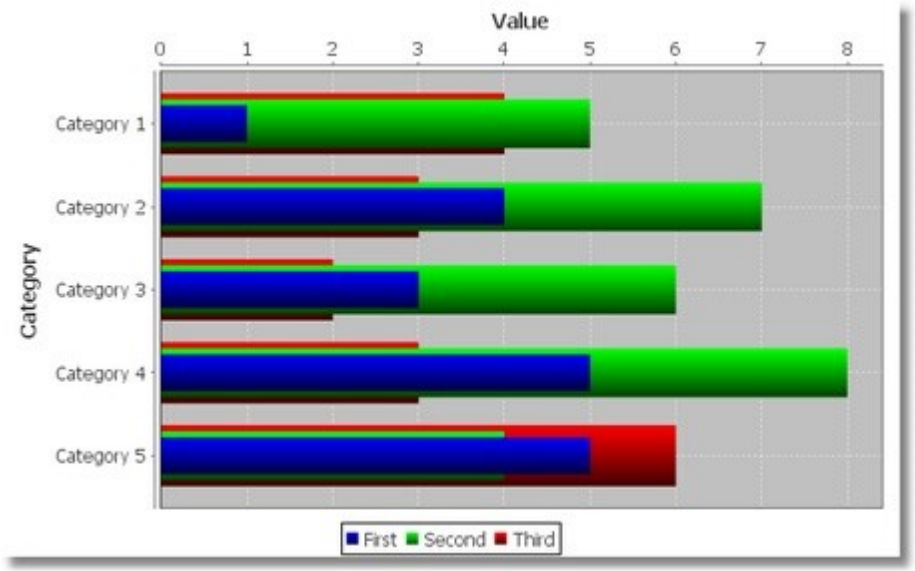
Столбчатая 3D диаграмма с горизонтальной ориентацией:



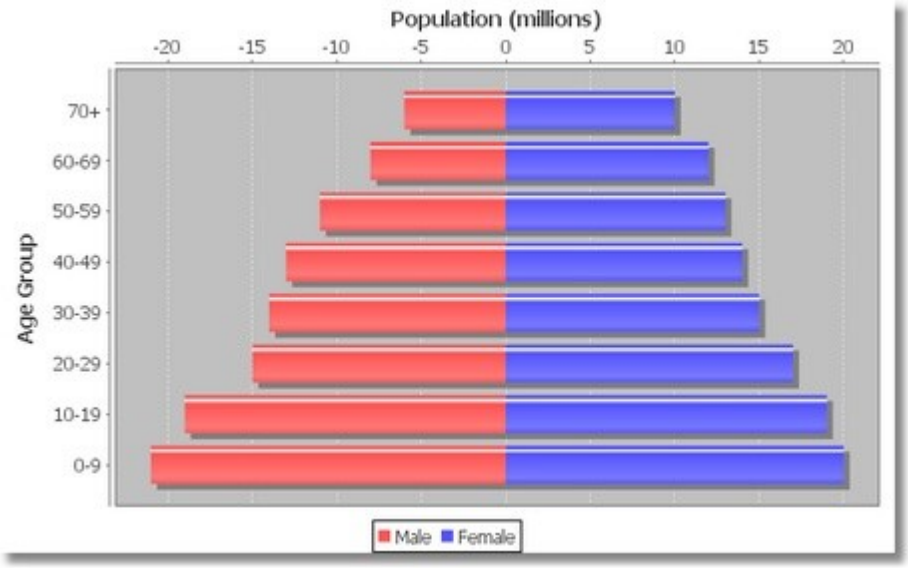
Столбчатая диаграмма с определенными пользователем **цветами окраски серий**:



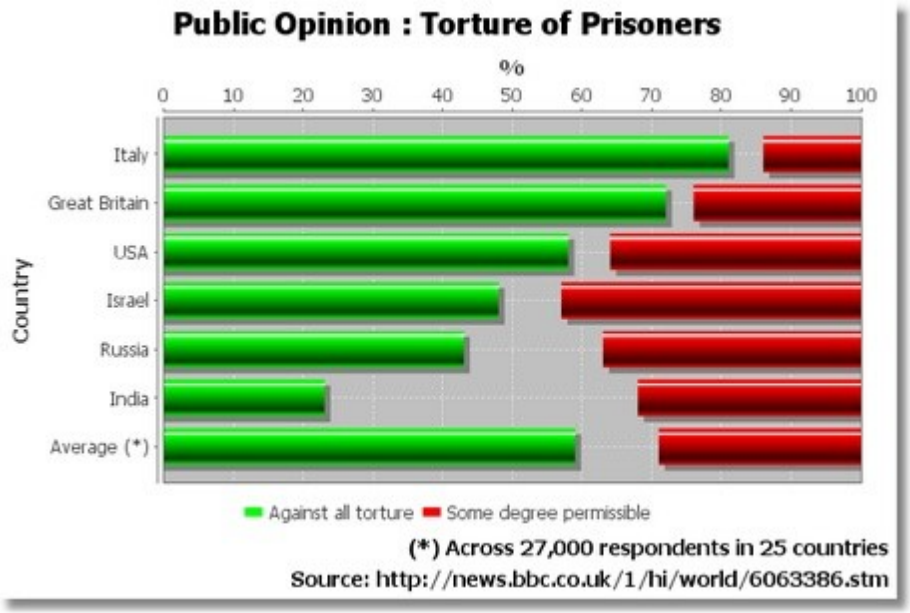
Многослойный столбчатая диаграмма с горизонтальной ориентацией:



Многослойный столбчатая диаграмма с отрицательными значениями:

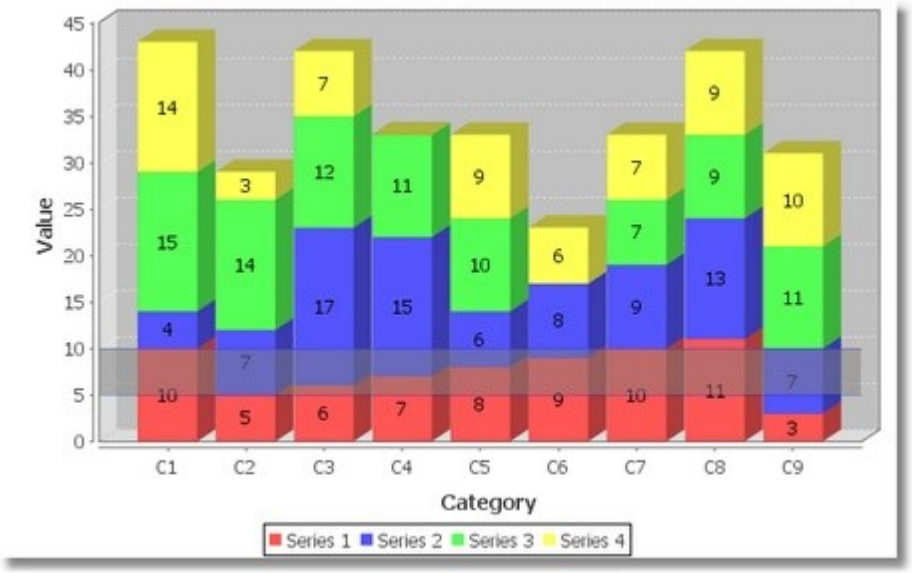


Составная столбчатая диаграмма с горизонтальной ориентацией и включенным свойством **Проценты**. Полностью прозрачный цвет применяется для средних серий, чтобы оставался промежуток на графике:

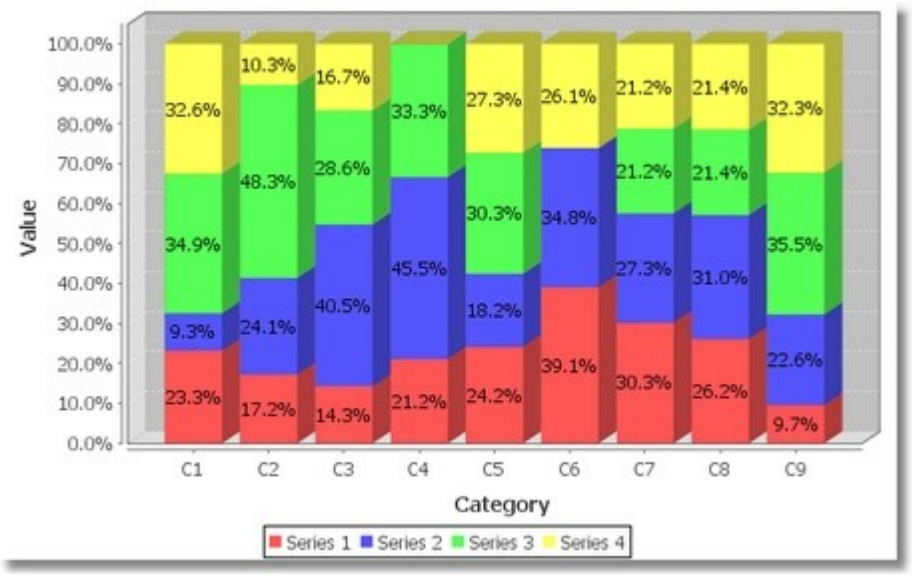


Столбчатая составная диаграмма, который использует **ось дат** в качестве оси измерений:

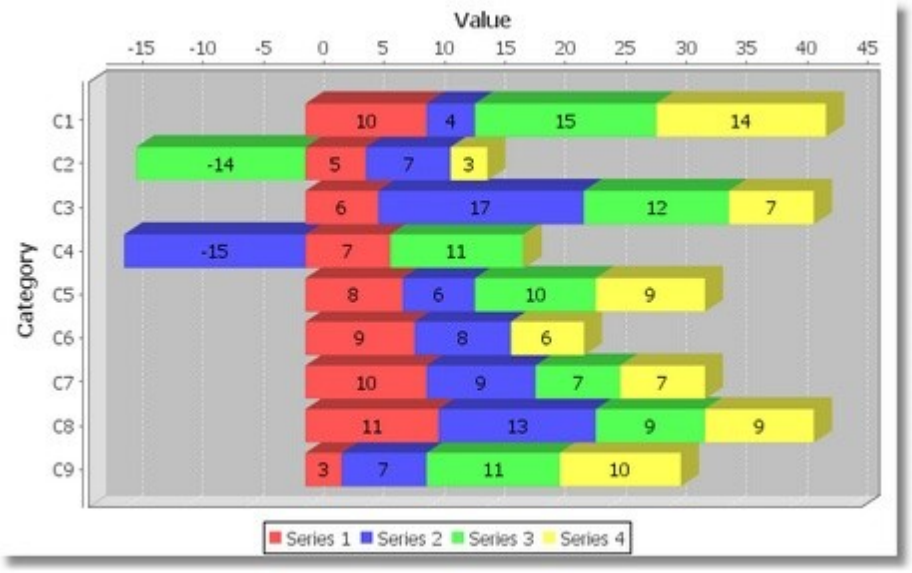
Столбчатая составная диаграмма с эффектом 3D и отметкой интервала:



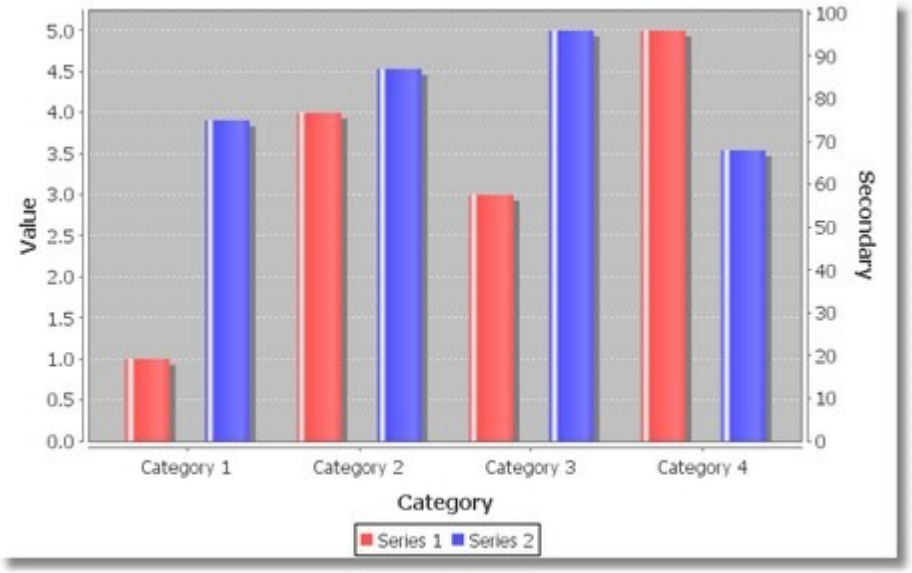
Столбчатая составная диаграмма с эффектом 3D и вертикальной ориентацией:



Столбчатая составная диаграмма с эффектом 3D и отрицательными значениями:



Столбчатая диаграмма с двумя осями измерений:

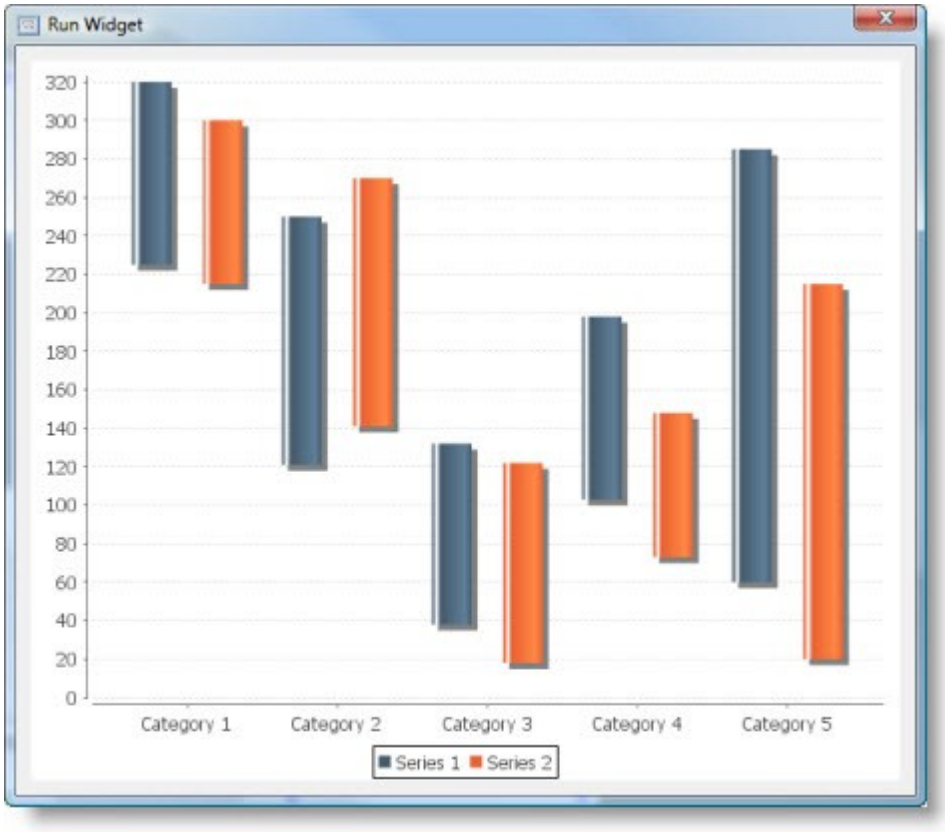


13.4.11.3.4 Столбчатая диаграмма интервалов

Столбчатая диаграмма, которая отображает интервалы (т.е. начальное значение и конечное значение) для каждой пары серии/категории в массиве данных.

Столбчатая диаграмма интервалов основана на области построения категорий[1187] и отрисовщике столбцов категорий[1229]. Он наследует все их свойства.

Столбчатый график с интервалами выглядит следующим образом:



Массив данных

Столбчатая диаграмма интервалов поддерживает только модель Пользовательские данные 1057.

Она имеет следующие привязки исходных данных:

Привязка	Тип ожидаемого значения	Описание
Серия	строка	Текстовое имя серии данных.
Категория	строка	Имя категории. Категории отображаются вдоль оси определений.
Начало	число	Значение начала интервала для вышеуказанных серий/категорий. Значения отображаются вдоль оси измерений.
Конец	число	Значение окончания интервала для вышеуказанных серии/категории.

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Непрозрачный 1275, Фон 1275, Рамка 1275

Все общие свойства графика 1160.

Свойства Исходные данные 1057 и Привязки исходных данных 1057.

Все свойства области построения категорий 1187.

Все свойства отрисовщика категорий столбцов 1229.

Пользовательские свойства

Не определены.

Общие события

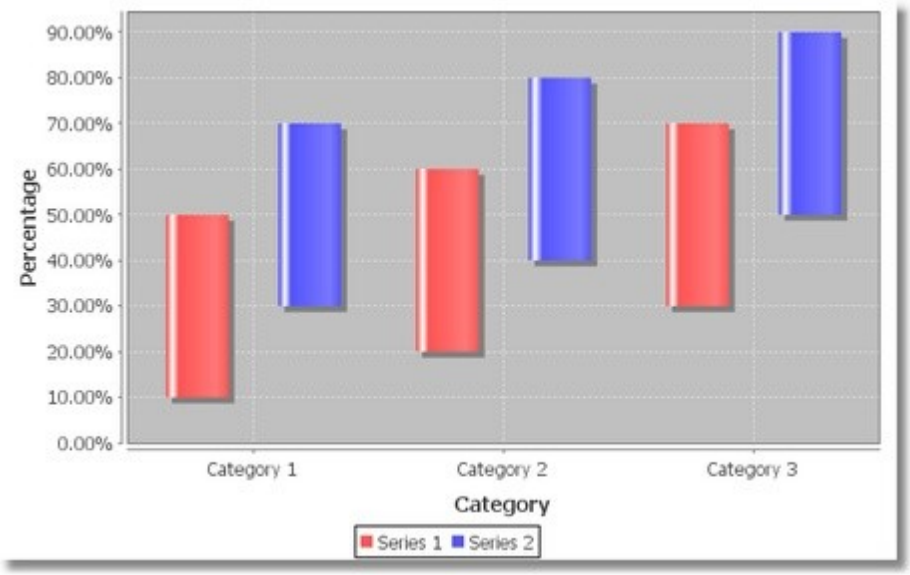
Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика

мыши, Печать клавиши, Нажатие клавиши, Отпускание клавиши, Получение фокуса, Потеря фокуса

Все соответствующие события графика.

Дополнительный пример

Другой пример столбчатой диаграммы интервалов:



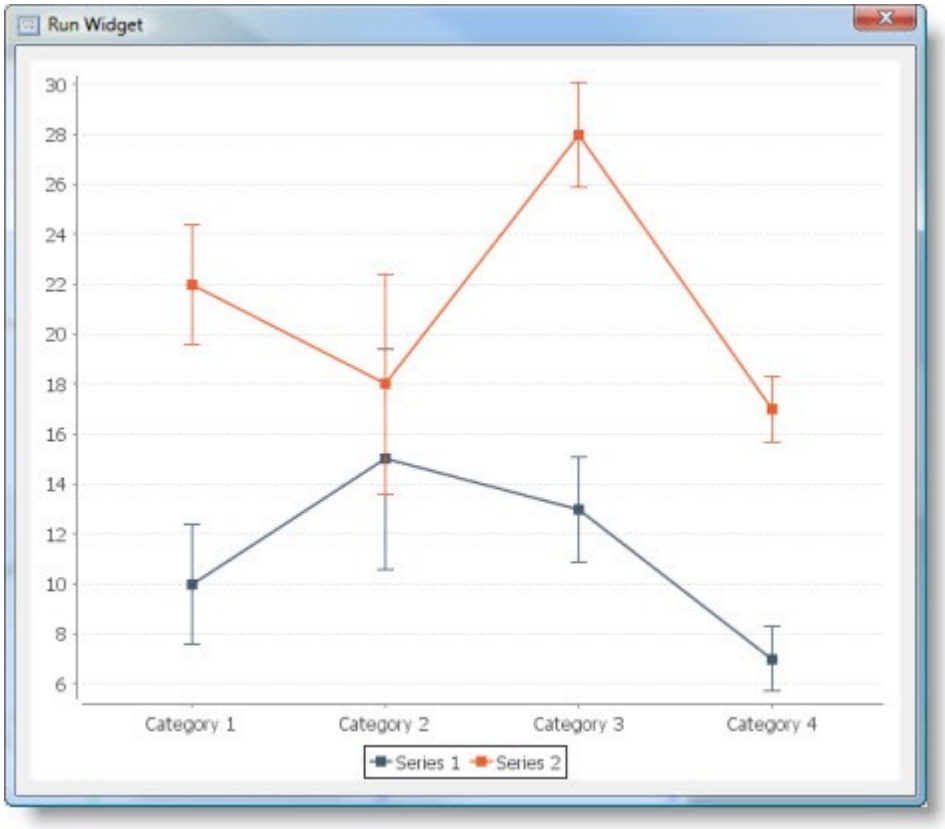
13.4.11.3.5 Статистическая диаграмма

Статистическая диаграмма показывает среднее и стандартное отклонение значения пар для каждой пары серии/категории в массиве данных.

Статистическая диаграмма основана на области построения категорий.

Ее отрисовщики основаны на отрисовщике линейных категорий и отрисовщике столбцов категорий.

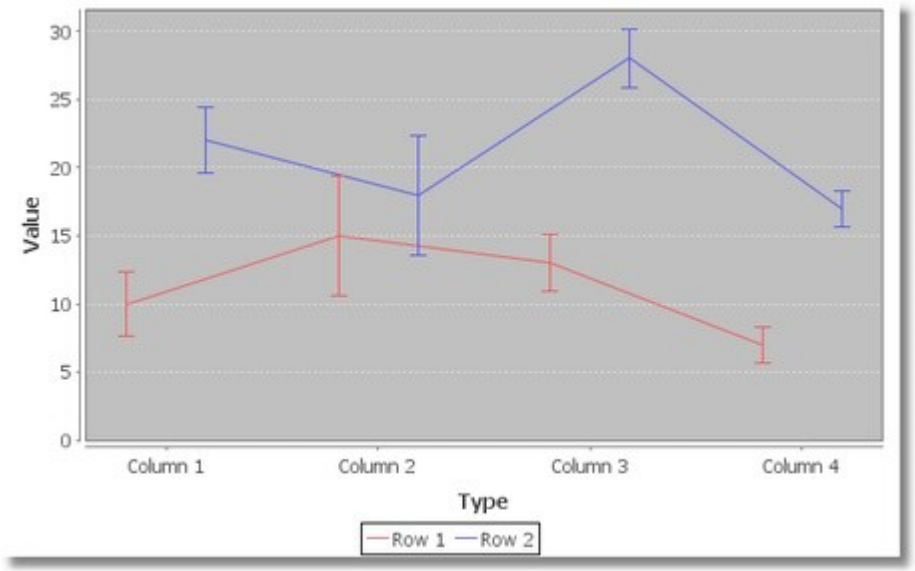
Статистическая диаграмма выглядит следующим образом:



Статистическая диаграмма поддерживает два отрисовщика.

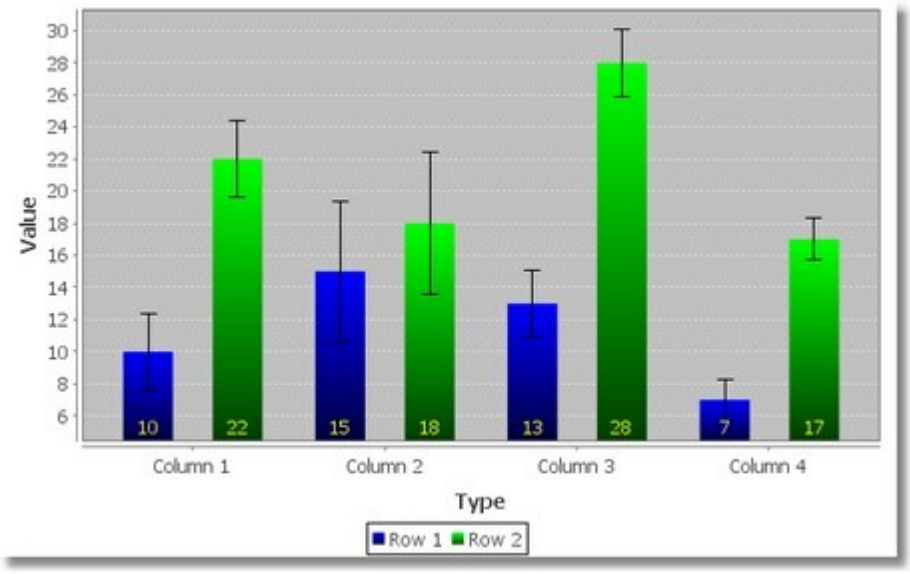
ЛИНЕЙНЫЙ ОТРИСОВЩИК

Отрисовщик, который отображает линии и/или фигуры для каждого значения данных и затем накладывает индикатор стандартного отклонения.



СТОЛБЧАТЫЙ ОТРИСОВЩИК

Отрисовщик, который отображает столбцы для каждого значения данных и затем накладывает индикатор стандартного отклонения.



Массив данных

Статистическая диаграмма поддерживает только Пользовательские данные[1057].

Она имеет следующие привязки исходных данных:

Привязка	Тип ожидаемого значения	Описание
Серия	строка	Текстовое имя серии данных.
Категория	строка	Имя категории. Категории отображаются вдоль оси определений.
Среднее	число	Среднее значение для вышеуказанных серии/категории. Значения отображаются вдоль оси измерений.
Отклонение	число	Значение стандартного отклонения вышеуказанных серии/категории.

Основные свойства

Ширина[1274], Высота[1274], Привязки[1275], Видимый[1275], Непрозрачный[1275], Фон[1275], Рамка[1275]

Все общие свойства графика[1160].

Свойства Исходные данные[1057] и Привязки исходных данных[1057].

Все свойства отрисовщика линейных категорий[1232] (если тип отрисовщика - Линейный)

Все свойства отрисовщике столбцов категорий[1229] (если тип отрисовщика - Столбчатый)

Пользовательские свойства

ЦВЕТ ИНДИКАТОРА ОШИБКИ

Цвет заливки[1279] для отображения индикатора отклонения (ошибки) для каждого элемента. Если является нулевым значением, используется цвет элемента (т.е. индикатор ошибки будет того же цвета, что и Линия/Фигура/Столбец/Контур столбца элемента).

Имя свойства: **errorIndicatorPaint**

Тип свойства: **Таблица данных**

ШТРИХ ИНДИКАТОРА ОШИБКИ

Штрих[1282] отображения индикатора отклонения (ошибки) для каждого элемента. Если является нулевым значением, используется цвет штриха контура элемента.

Имя свойства: **errorIndicatorStroke**

Тип свойства: **Таблица данных**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1288, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

Все соответствующие события графика 1182.

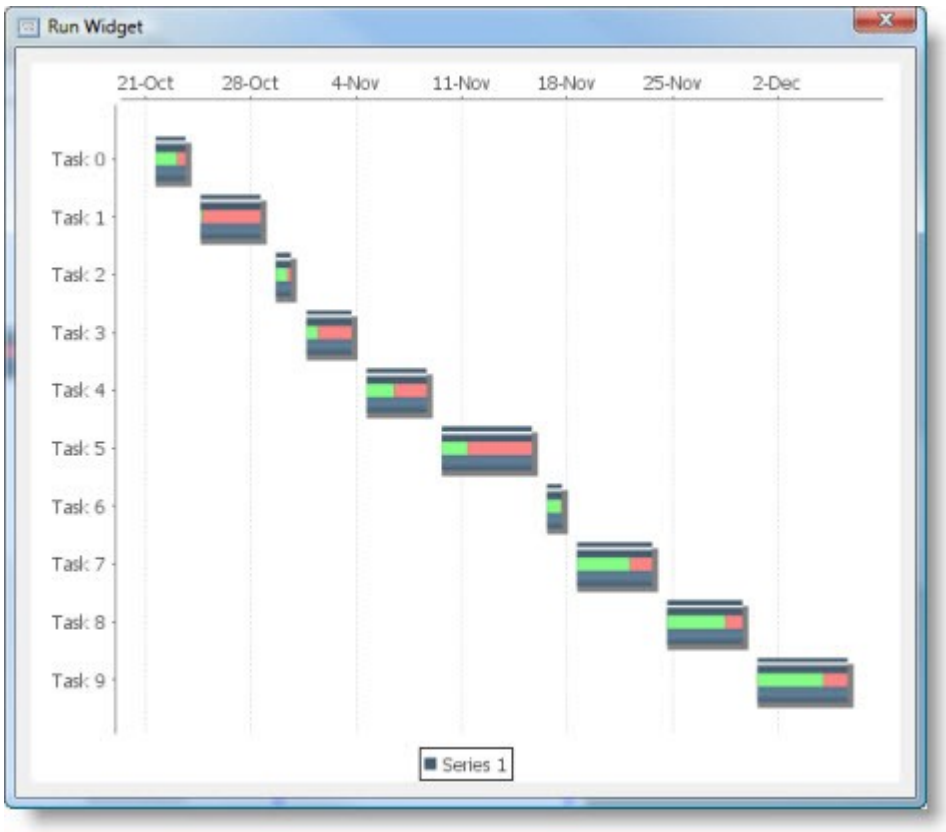
13.4.11.3.6 Диаграмма Ганта

Диаграмма Ганта используется для отображения простых диаграмм Ганта.

Диаграмма Ганта является столбчатым графиком работы. Диаграммы Ганта отображают даты начала и конца отдельных задач проекта.

Диаграмма Ганта основана на области построения категорий 1187 и отрисовщике столбцов категорий 1229. Она наследует все их свойства.

Диаграмма Ганта выглядит следующим образом:



Массив данных

Диаграмма Ганта поддерживает только модель Пользовательские данные 1057.

Она имеет следующие привязки исходных данных:

Привязка	Тип ожидаемого значения	Описание
Серия	строка	Текстовое имя серии данных. Серия обычно используется для указания проектных имен в мультипроектных графиках.
Описание	строка	Описание задачи. Задачи отображаются вдоль оси параметров. Если одна серия содержит несколько задач с одинаковым описанием, они будут отображаться на

		графике в виде нескольких столбцов (подзадач), соответствующих одной задаче.
Дата начала	дата или число	Временная метка начала задачи. Числовые значения конвертируются в даты, они рассматриваются в виде количества миллисекунд, прошедших с 1 января 1970.
Дата конца	дата или число	Временная метка окончания задачи.
Процент выполнения	число	Процент выполнения задачи.

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Непрозрачный 1275, Фон 1275, Рамка 1275

Все общие свойства графиков 1160.

Свойства Исходные данные 1057 и Привязки исходных данных 1057.

Все свойства области построения категорий 1187.

Все свойства отрисовщика столбцов категорий 1229.

Пользовательские свойства

ОКРАСКА ЗАВЕРШЕННОЙ ЧАСТИ

Цвет 1279 заливки выполненной части задачи.

Имя свойства: **completePaint**

Тип свойства: **Таблица данных**

ОКРАСКА НЕЗАВЕРШЕННОЙ ЧАСТИ

Цвет 1279 заливки еще невыполненной части задачи.

Имя свойства: **incompletePaint**

Тип свойства: **Таблица данных**

ПРОЦЕНТ НАЧАЛА ОБЛАСТИ ЗАВЕРШЕННОСТИ

Начальная позиция индикатора "процент выполнения" в виде процентного отношения к **ширине** столбца задачи (например, 0.30 равно 30 процентам).

Имя свойства: **startPercent**

Тип свойства: **Плавающее**

ПРОЦЕНТ ОКОНЧАНИЯ ОБЛАСТИ ЗАВЕРШЕННОСТИ

Конечная позиция индикатора "процент выполнения" в виде процентного отношения к **ширине** столбца задачи (например, 0.30 равно 30 процентам).

Имя свойства: **endPercent**

Тип свойства: **Плавающее**

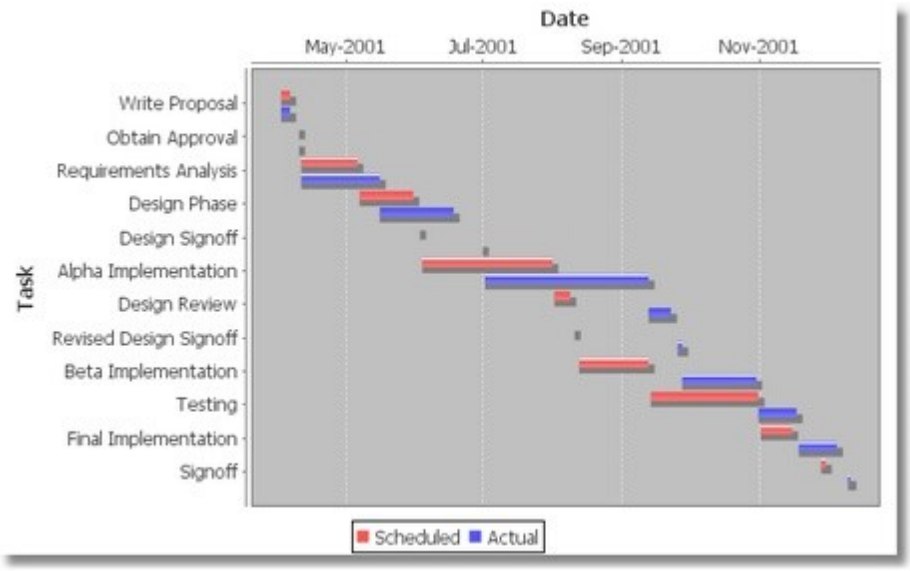
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

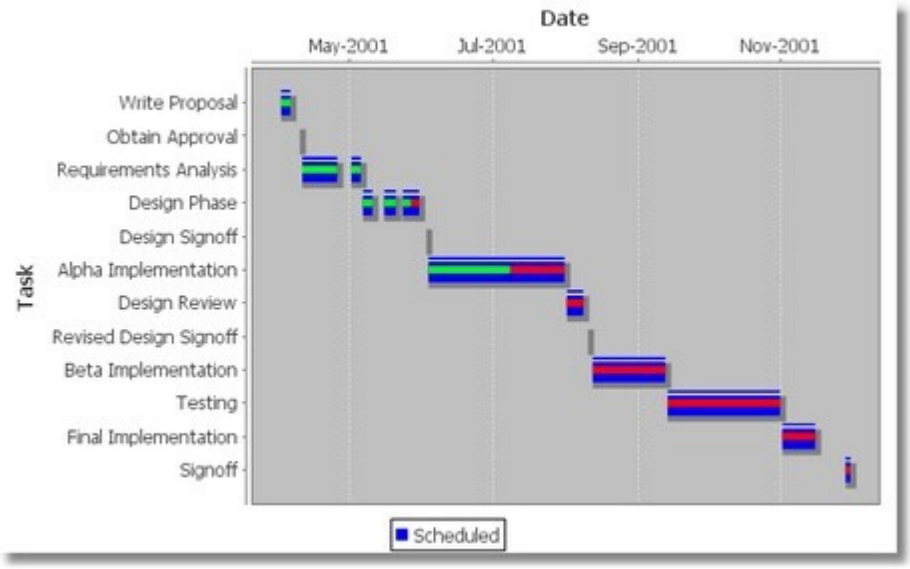
Все соответствующие события графика 1182.

Дополнительные примеры

Простая Диаграмма Ганта, отображающая процент текущего выполнения в отношении запланированного:



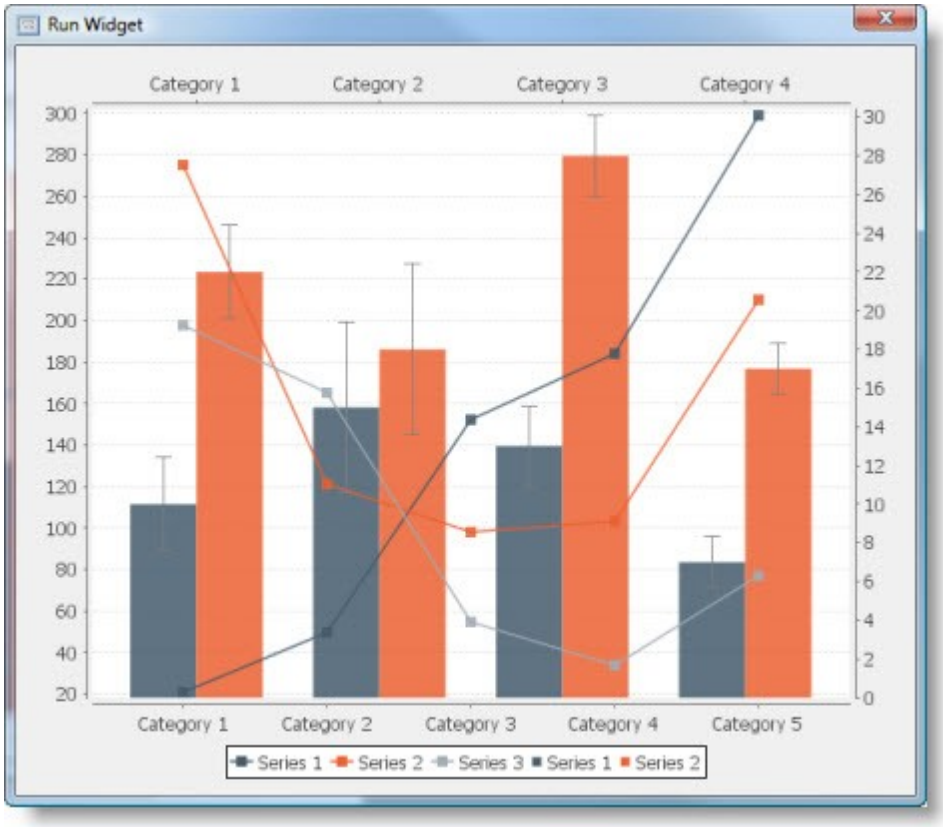
Простая Диаграмма Ганта, отображающая завершенность процесса определенных задач на данное число:



13.4.11.3.7 Смешанная диаграмма

Смешанная диаграмма представляет собой составную диаграмму 1160, включающую в себя два и более "простых" графика категорий 1067 в одной области построения. Данные подграфики используют одну и ту же область построения и оси, однако, каждый из них имеет собственный массив данных и отрисовщик.

Смешанная диаграмма выглядит следующим образом:



Когда к смешанной диаграмме категорий добавляется "простой" график в виде подграфика, он немного отличается от своей обычной версии:

- Оси подгдрафика добавляются в список осей графика-контейнера. Подграфик не имеет собственных осей. Вместо них он обладает ссылками на оси графика-родителя, которые используются для отображения данных. По умолчанию ссылка осуществляется на оси, переданные от подграфика.
- Подграфик не имеет свойств, касающихся области построения 1185, т.к. его данные будут отображены на области построения графика-родителя.
- Подграфик не имеет общих свойств графика 1160, график-родитель имеет свои общие свойства.
- Подграфик не имеет свойств компонента виджета по умолчанию 1274, кроме свойства **Привязки**, т.к. оно не отображается отдельно.

Пользовательские свойства подграфика

Каждый график, добавленный к смешанному графику, обладает несколькими дополнительными свойствами:

ИНДЕКС ОСИ ПАРАМЕТРОВ

Индекс используемой оси параметров смешанного графика-родителя. Первая ось в таблице осей параметров имеет индекс **1.**

Имя свойства: **domainAxisIndex**

Тип свойства: **Целое**

ИНДЕКС ОСИ ЗНАЧЕНИЙ

Индекс используемой оси значений смешанного графика-родителя. Первая ось в таблице осей значений графика-родителя имеет индекс **1.**

Имя свойства: **rangeAxisIndex**

Тип свойства: **Целое**

ВЫСОТА (Z-ORDER)

Высота подграфика в области построения графика-родителя. Графики с меньшими порядковыми номерами перекрывают графики с большими номерами.

Имя свойства: **index**

Тип свойства:**Целое**

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Непрозрачный 1275, Фон 1275, Рамка 1275

Все общие свойства графиков 1160.

Все свойства области построения категорий 1187.

Пользовательские свойства

Не определены.

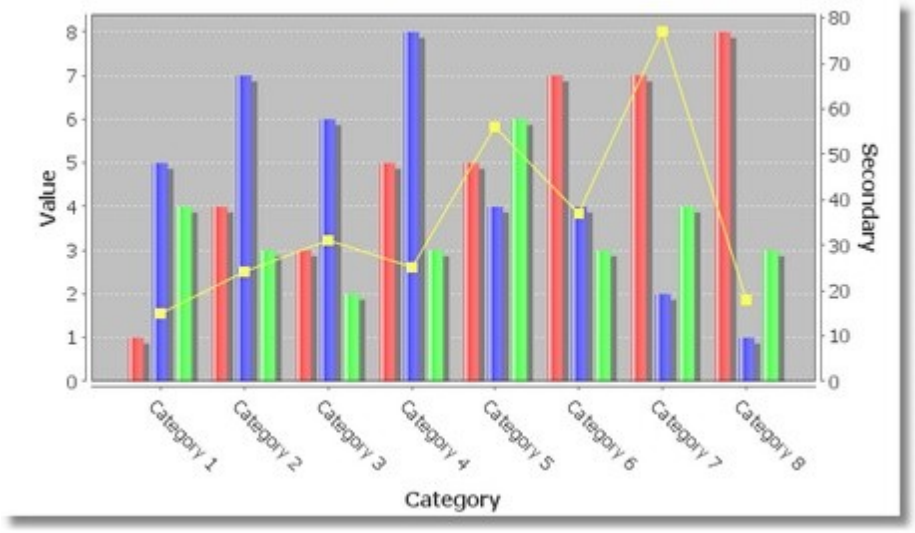
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

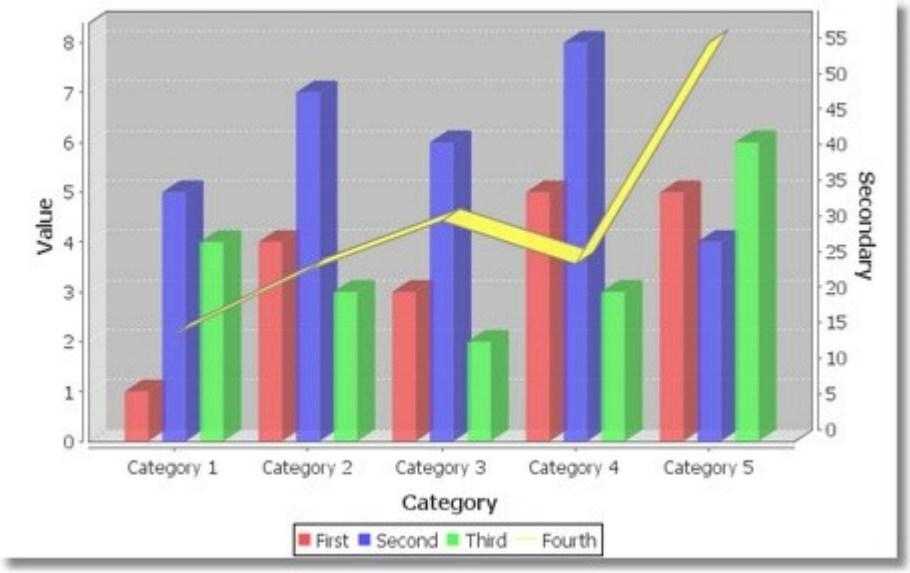
Все соответствующие события графика 1182.

Дополнительные примеры

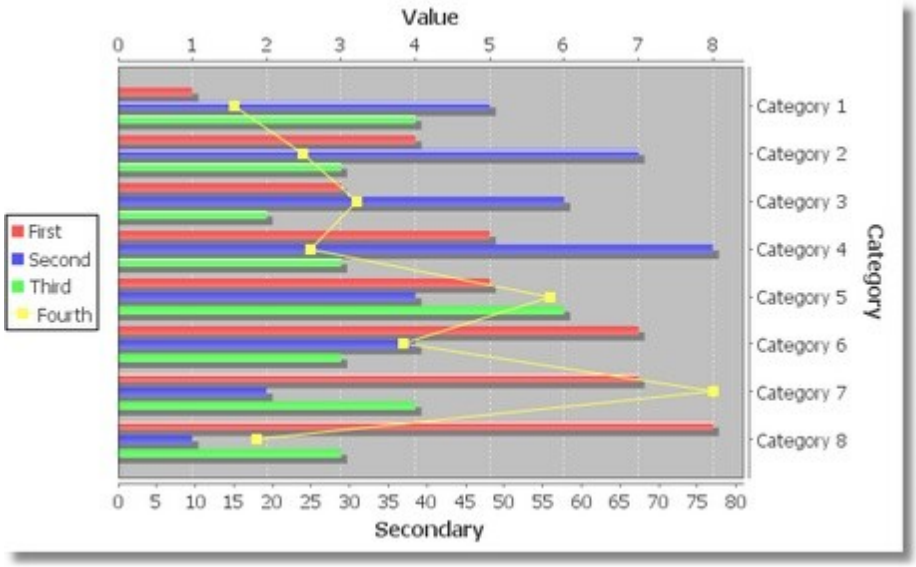
Смешанная диаграмма с линейным 1067 и столбчатым 1074 подграфиками с различными осями измерений:



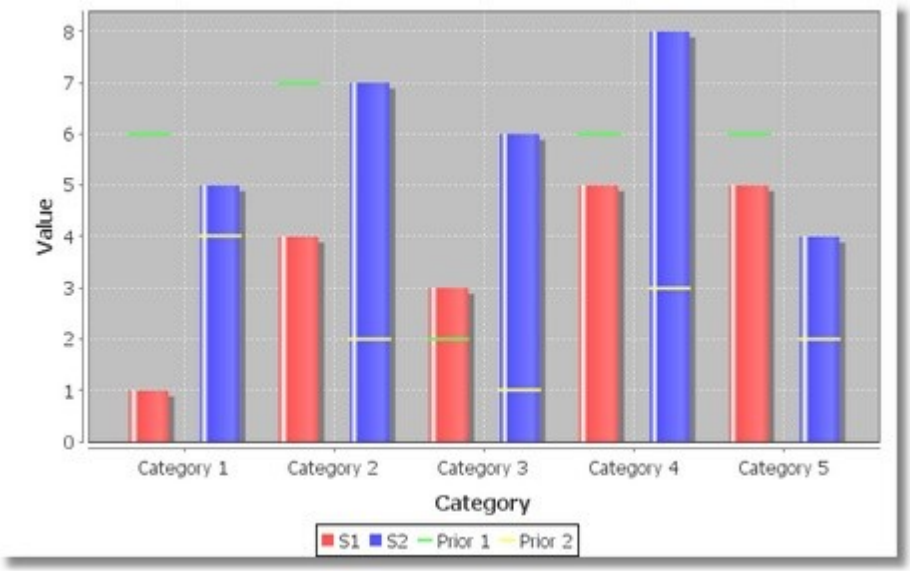
То же, что и в предыдущем примере, только с 3D:



То же, что и в предыдущем примере, только с горизонтальной ориентацией:



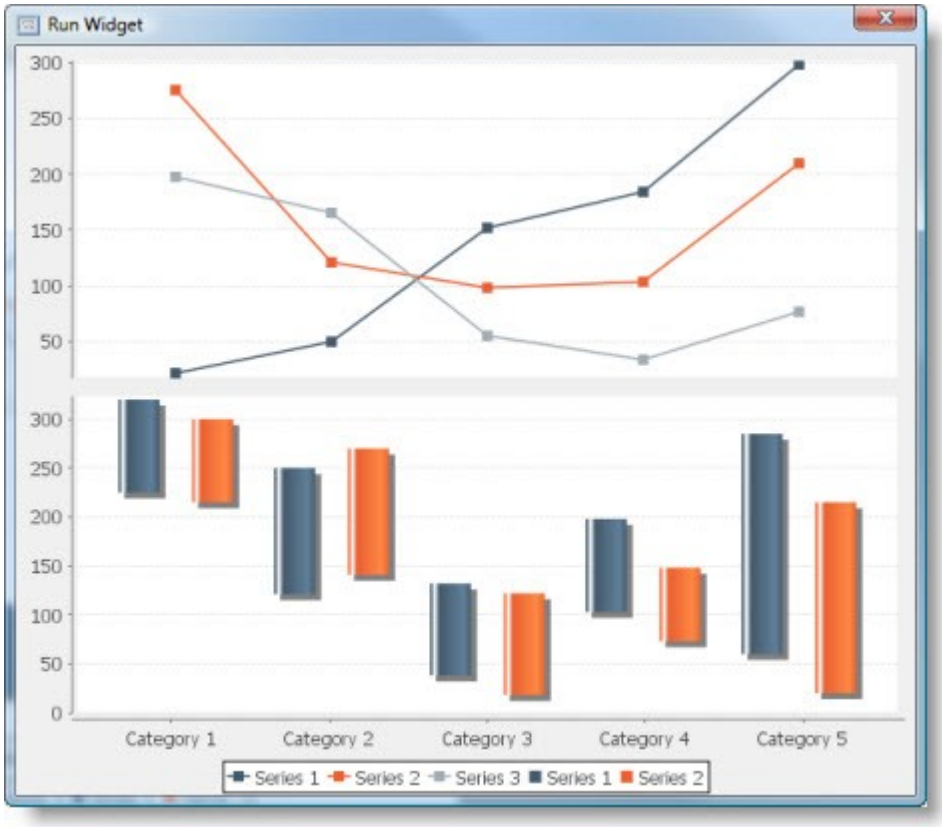
Смешанная диаграмма со столбчатым подграфиком и линейным подграфиком, использующим уровневый отрисовщик:



13.4.11.3.8 Диаграмма с общей осью параметров

Диаграмма с общей осью параметров является составной диаграммой[1160], в которой два и более графика категорий[1067] используют одну ось параметров.

Диаграмма с общей осью параметров выглядит следующим образом:



Когда "простой" график категорий добавляется в качестве подграфика к графику с общей осью параметров, он имеет некоторые отличия от своей обычной версии:

- Не имеет свойств Оси параметров[1189] и Фиксированное пространство оси параметров[1190], т.к. ось параметров является частью графика-родителя.
- Не имеет свойства Ориентация[1193], т.к. унаследует ориентацию от графика-родителя.
- Не имеет общих свойств графика[1160], т.к. график-родитель обладает своим набором основных свойств графика.
- Не имеет свойств компонента виджета по умолчанию[1274], за исключением свойства **Привязки**, т.к. оно не отображается отдельно.

Пользовательские свойства подграфиков

Каждый график, добавленный в качестве подграфика к графику с общей областью параметров, имеет несколько дополнительных свойств:

ВЕС

Вес определяет, какое пространство области построения графика-родителя предназначено для данного подграфика. Например, если родитель имеет три подграфика, вес которых равен 1, 2 и 4, пространство, предназначенное для каждого подграфика, будет составлять 1/7, 2/7 и 4/7 соответственно (где 7 - сумма весов всех графиков).

Имя свойства: **weight**

Тип свойства: **Целое**

ВЫСОТА (Z-ORDER)

Подграфики с большим порядковым номером размещаются ближе к общей оси.

Имя свойства: **index**

Тип свойства: **Целое**

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Непрозрачный 1275, Фон 1275, Рамка 1275

Все общие свойства графиков 1160.

Свойство Оси определений 1189.

Свойство Фиксированное пространство оси параметров 1190.

Пользовательские свойства

ОРИЕНТАЦИЯ

Ориентация графика (Вертикальная или Горизонтальная). Данная ориентация применяется ко всем подграфикам.

Имя свойства: **orientation**

Тип свойства: **Строка**

ПРОМЕЖУТОК

Интервал между подграфиками.

Имя свойства: **gap**

Тип свойства: **Плавающее**

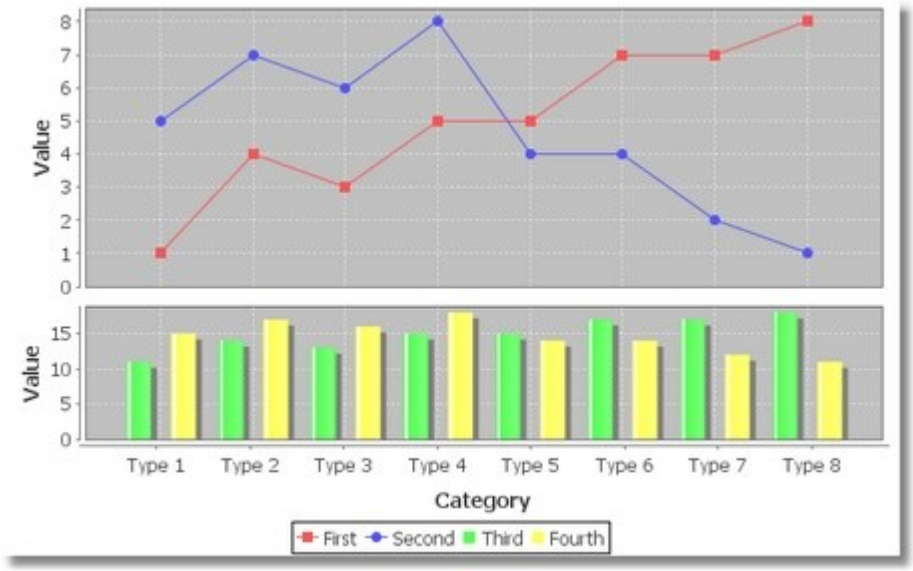
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

Все соответствующие события графика 1182.

Дополнительные примеры

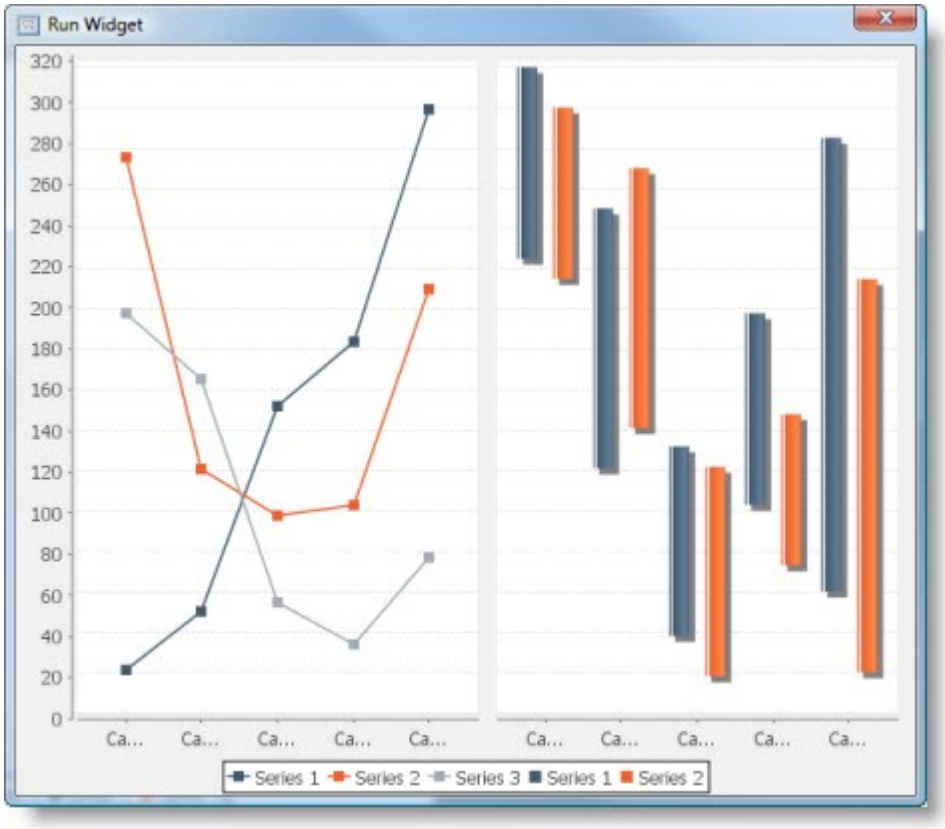
Два подграфика на одной оси параметров:



13.4.11.3.9 Диаграмма с общей осью значений

Диаграмма с общей осью значений является составным графиком 1160, в котором два и более графика категорий 1067 используют одну ось значений.

График с общей областью значений выглядит следующим образом:



Когда график категорий добавляется в качестве подграфика к графику с общей областью значений, он имеет некоторые отличия от своей обычной версии:

- Не имеет свойств Оси измерений[1189] и Фиксированное пространство оси измерений[1190], т.к. ось измерений является частью графика-родителя.
- Не имеет свойства Ориентация[1193], т.к. унаследует ориентацию от графика-родителя.
- Не имеет общих свойств графика[1160], т.к. график-родитель обладает своим набором основных свойств графика.
- Не имеет свойств компонента виджета по умолчанию[1274], за исключением свойства **Привязки**, т.к. оно не отображается отдельно.

Пользовательские свойства подграфиков

Каждый график, добавленный в качестве подграфика к графику с общей областью значений, имеет несколько дополнительных свойств:

ВЕС

Вес определяет, какое пространство области построения графика-родителя предназначено для данного подграфика. Например, если родитель имеет три подграфика, вес которых равен 1, 2 и 4, пространство, предназначенное для каждого подграфика, будет составлять 1/7, 2/7 и 4/7 соответственно (где 7 - сумма индивидуального веса).

Имя свойства: **weight**

Тип свойства: **Целое**

ВЫСОТА (Z-ORDER)

Подграфики с большим порядковым номером размещаются ближе к общей оси.

Имя свойства: **index**

Тип свойства: **Целое**

Общие свойства

Ширина[1274], Высота[1274], Привязки[1275], Видимый[1275], Непрозрачный[1275], Фон[1275], Рамка[1275]

Все общие свойства графиков[1160].

Свойство Оси измерений 1189.

Свойство Фиксированное пространство оси измерений 1190.

Пользовательские свойства

ОРИЕНТАЦИЯ

Ориентация графика (Вертикальная или Горизонтальная). Данная ориентация применяется ко всем подграфикам.

Имя свойства: **orientation**

Тип свойства: **Строка**

ПРОМЕЖУТОК

Интервал между подграфиками.

Имя свойства: **gap**

Тип свойства: **Плавающее**

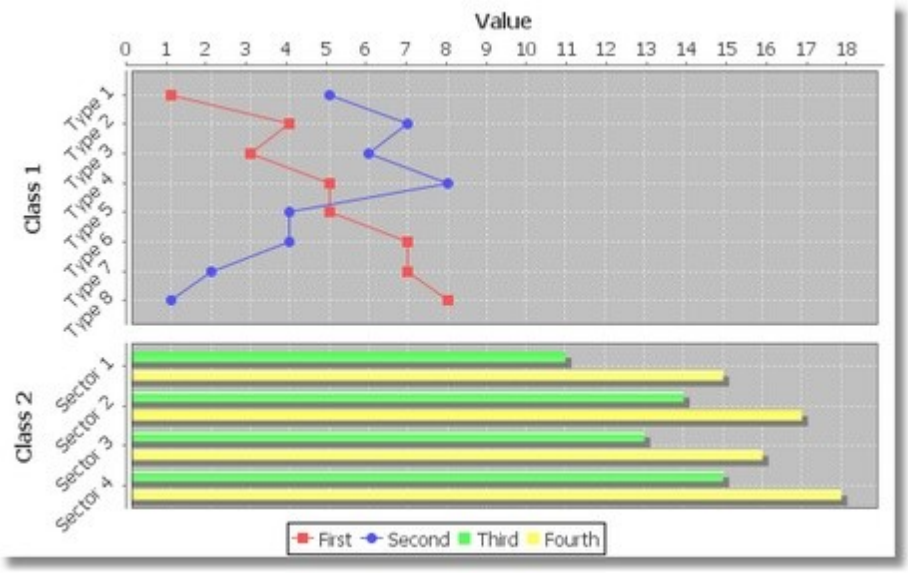
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

Все соответствующие события графика 1182.

Дополнительные примеры

Два подграфика с горизонтальной ориентацией с общей осью значений:



13.4.11.4 Координатные (X,Y) графики

Координатные графики отображают значения в двумерном пространстве. Оси параметров и значений аналогичны.

Все координатные графики основаны на:

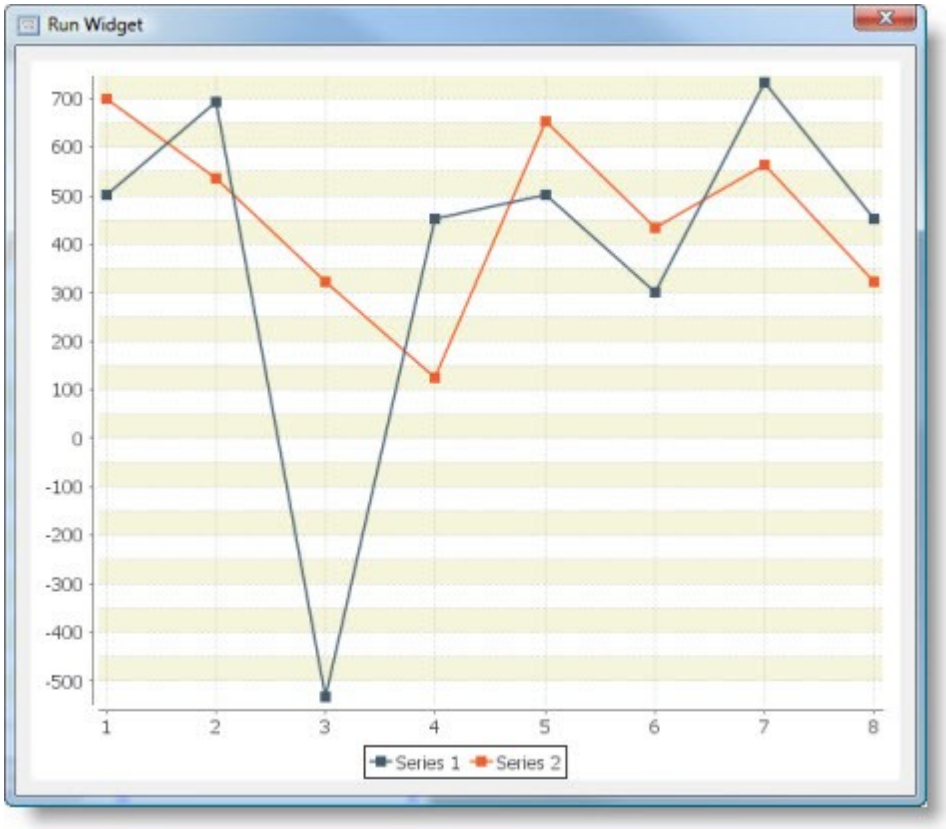
- Координатной области построения 1197
- Координатном отрисовщике 1234

13.4.11.4.1 XY-Диаграмма

XY-Диаграмма просто соединяет все точки данных (X, Y) прямыми линиями.

Диаграмма основана на координатной области построения 1197 и координатном отрисовщике 1234. Она наследует все их свойства.

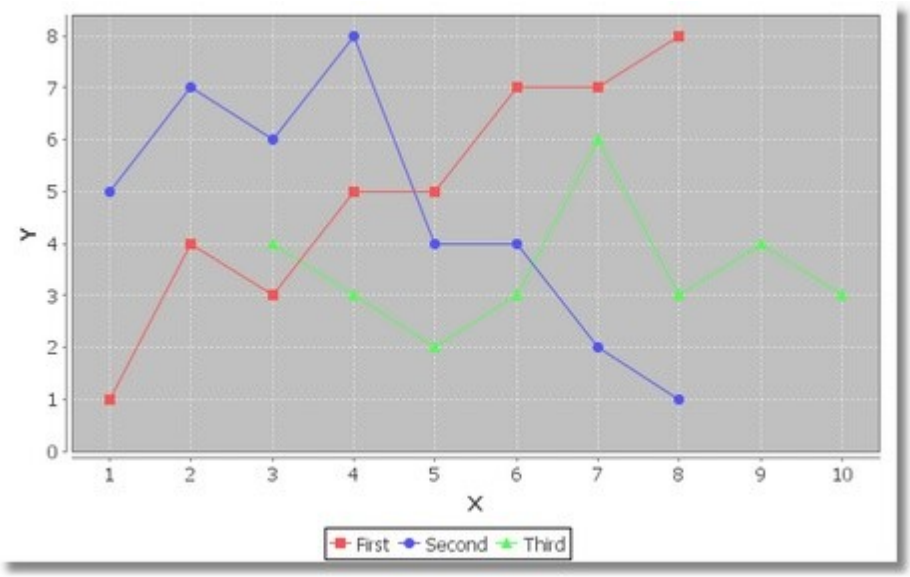
XY-Диаграмма выглядит следующим образом:



XY-Диаграмма поддерживает четыре вида отрисовщика:

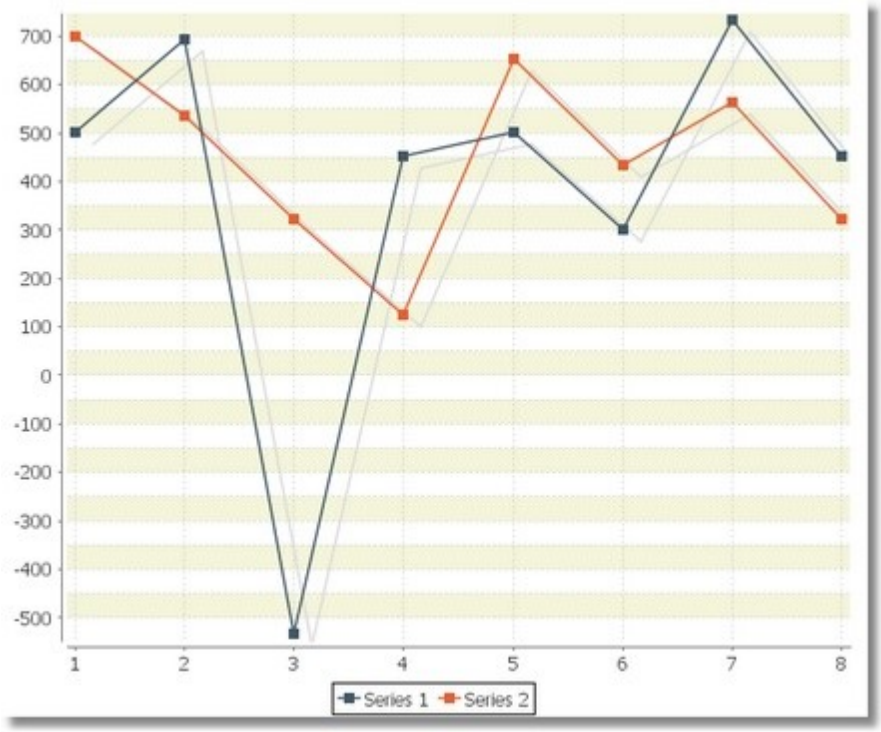
ЛИНЕЙНЫЙ ОТРИСОВЩИК

Отрисовщик, отображающий элементы при помощи линии между каждой точкой (x, y) и фигуры в каждой точке (x, y).



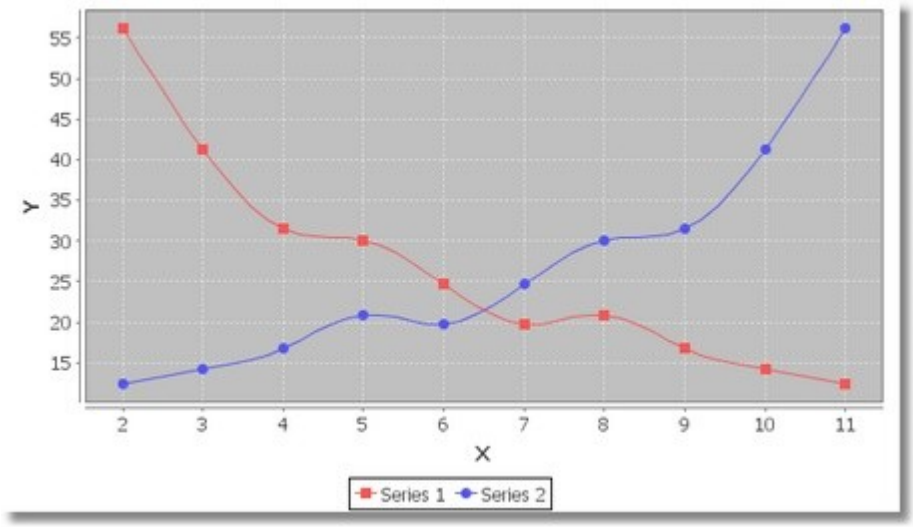
ЛИНЕЙНЫЙ 3D ОТРИСОВЩИК

Отрисовщик, использующий "псевдо-3D" эффект для начертания линий.



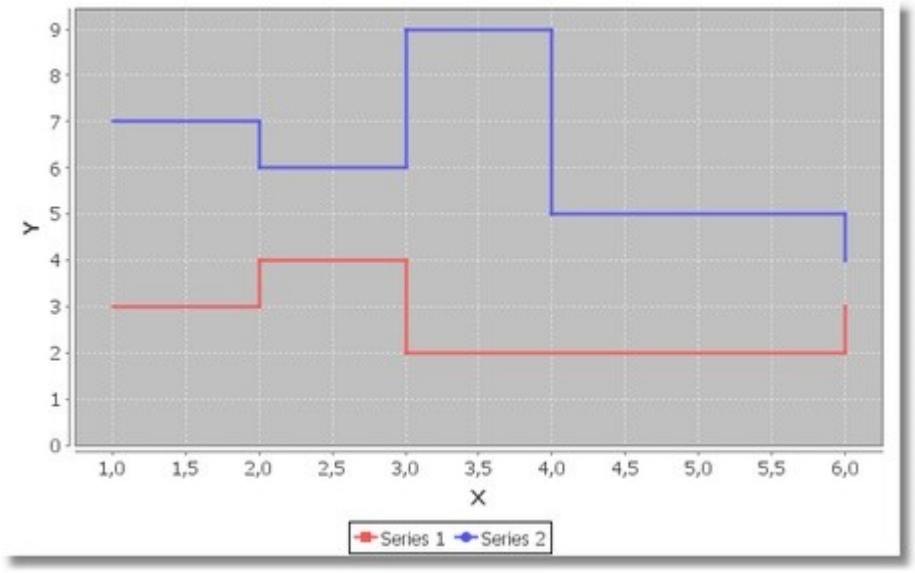
СПЛАЙНОВЫЙ ОТРИСОВЩИК

Отрисовщик, использующий сглаженные кривые для соединения точек. В результате получается плавная линия, проходящая через все точки на графике.



СТУПЕНЧАТЫЙ ОТРИСОВЩИК

Отрисовщик отображает элементы "ступенчатыми" линиями для соединения каждой точки (x, y).



Массив данных

XY-Диаграмма поддерживает несколько моделей данных:

- Пользовательские данные 1057
- Данные, основанные на событии 1064
- Данные, основанные на переменной 1061

Имеет следующие привязки исходных данных:

Привязка	Ожидаемый тип значения	Описание
Серия	строка	Текстовое имя серии данных. Серия данных на графике представлена в виде линии (или набора фигур).
X	число	Числовое значение, отображаемое вдоль оси определений (X).
Y	число	Числовое значение, отображаемое вдоль оси измерений (Y).

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Непрозрачный 1275, Фон 1275, Рамка 1275

Все общие свойства графиков 1160.

Все свойства, относящиеся к данным 1053.

Все свойства координатной области построения 1197.

Все свойства координатного линейного отрисовщика 1236.

Пользовательские свойства

3D X-СМЕЩЕНИЕ

Смещение X для 3D эффекта.

Данное свойство пригодно для 3D рендера линии.

Имя свойства: **XOffset**

Тип свойства: **Плавающее**

3D Y-СМЕЩЕНИЕ

Смещение Y для 3D эффекта.

Данное свойство пригодно для 3D рендера линии.

Имя свойства: **YOffset**

Тип свойства: **Плавающее**

ОКРАСКА СТЕН

Цвет[1279], используемый для заливки "сторон" (или "стен") фона области построения графика.

Данное свойство пригодно для 3D отрисовщика.

Имя свойства: **wallPaint**

Тип свойства:**Таблица данных**

ТОЧНОСТЬ

Число сегментов линии, используемое для отображения кривой между точками графика.

Данное свойство пригодно для сплайнового отрисовщика.

Имя свойства: **precision**

Тип свойства: **Целое**

ПОЛОЖЕНИЕ СТУПЕНЬКИ

Дробная часть позиции определений между двумя точками, по которым рисуется ступень. Значение по умолчанию является 1.0 и означает, что ступень строится в позиции определений второй точки. Если "Точка ступени" - 0.5, ступень строится по середине между двумя точками.

Данное свойство пригодно для ступенчатого отрисовщика.

Имя свойства: **stepPoint**

Тип свойства: **Плавающее**

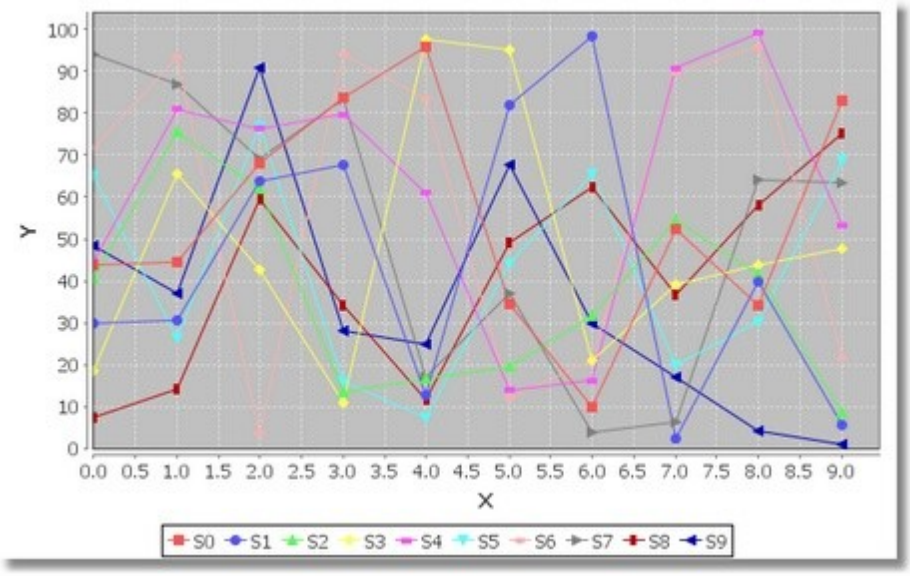
Общие события

Скрытие[1285], Показ[1285], Перемещение[1285], Изменение размеров[1285], Клик мыши[1286], Нажатие кнопки мыши[1286], Отпускание кнопки мыши[1286], Вход мыши[1286], Выход мыши[1286], Перемещение мыши[1286], Вращение колесика мыши[1286], Печать клавиши[1287], Нажатие клавиши[1287], Отпускание клавиши[1287], Получение фокуса[1287], Потеря фокуса[1287]

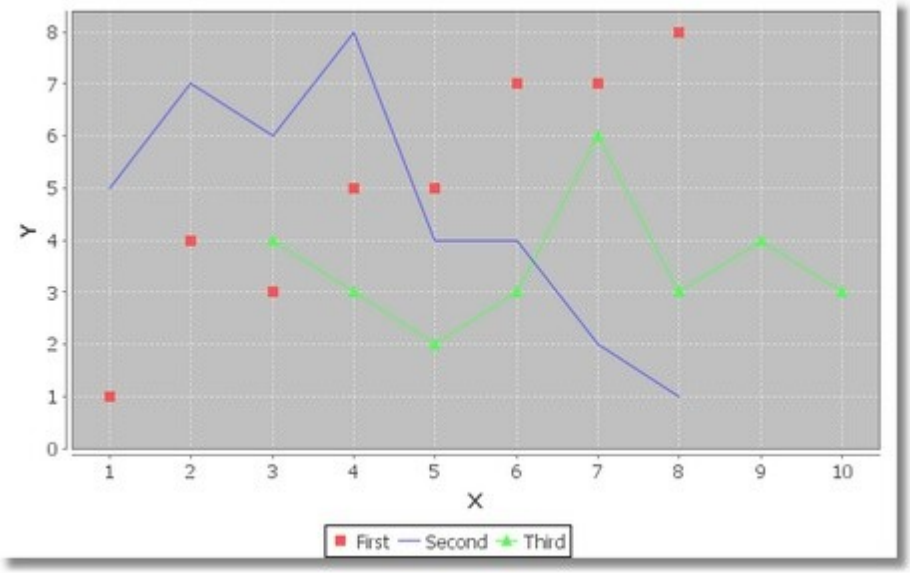
Все соответствующие события графика[1182].

Дополнительные примеры

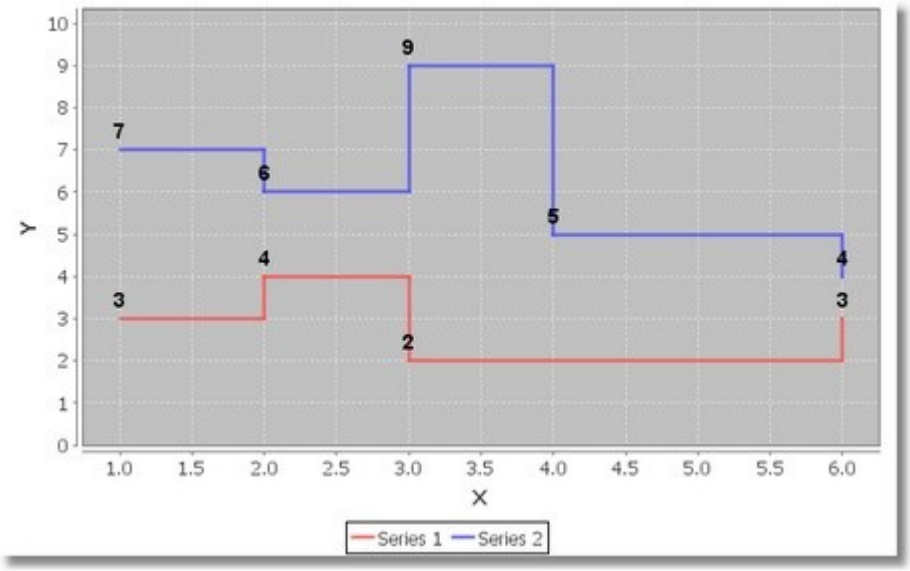
XY-Диаграмма с множеством серий данных. В каждой точке отображается фигура для различия между сериями:



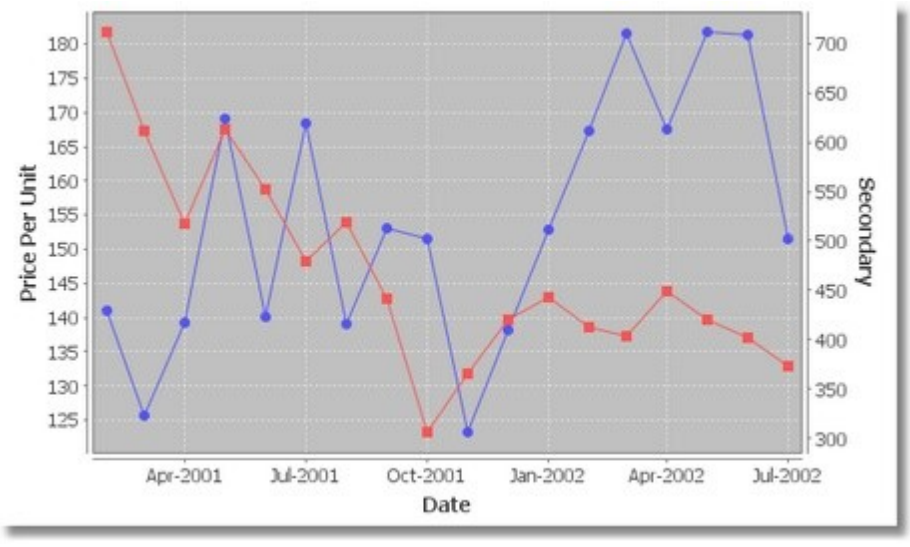
XY-Диаграмма, где каждая серия отображена в виде различных комбинаций линий и/или фигур:



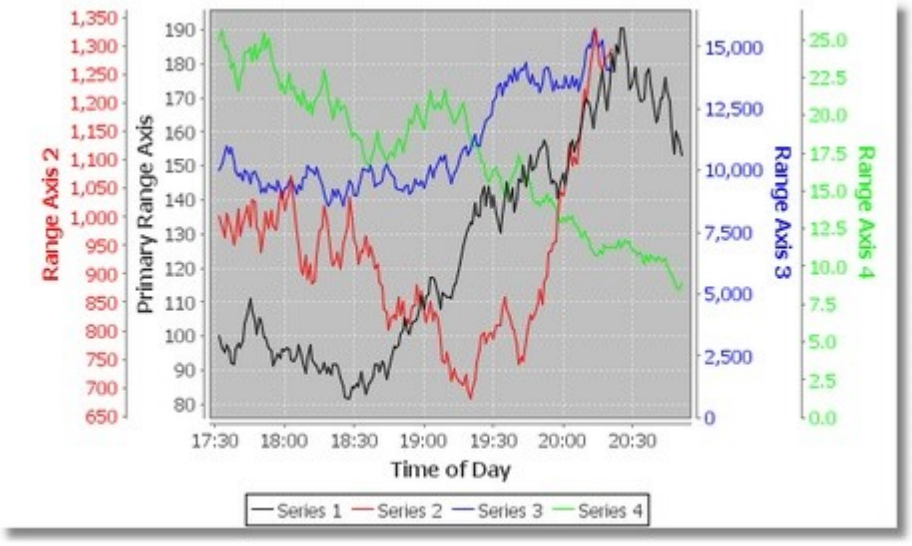
Ступенчатая XY-Диаграмма с метками элементов:



XY-Диаграмма с двумя осями измерений:



XY-Диаграмма с множеством осей измерений:



XY-Диаграмма с четырьмя сериями данных и указателями[1209]:

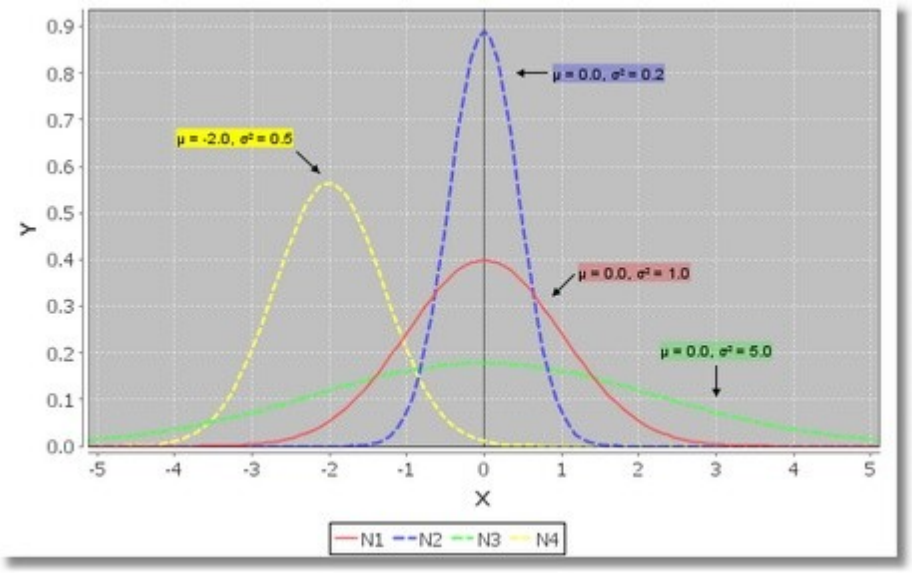
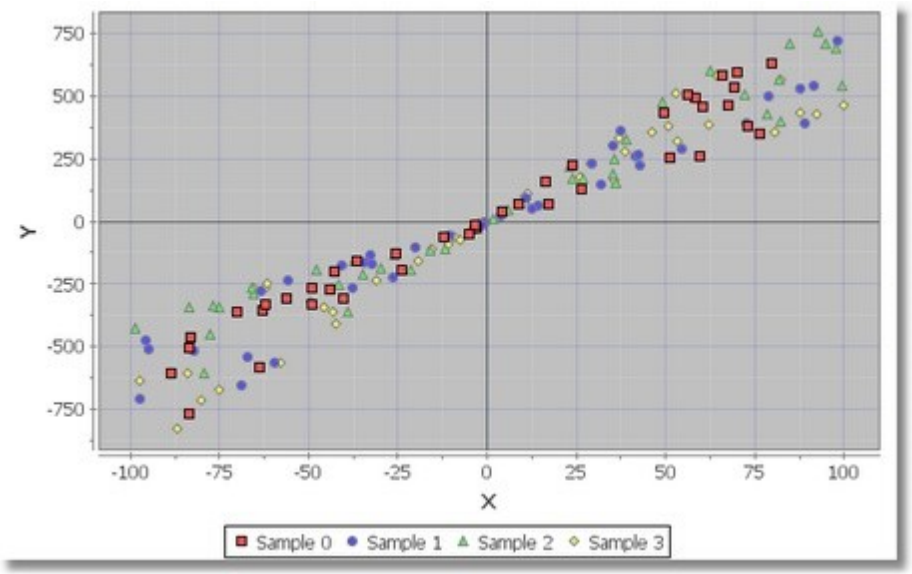


График рассеяния, отображающий множество серий данных. Использует рендер линии при отключенном свойстве **Видимость линий по умолчанию**:

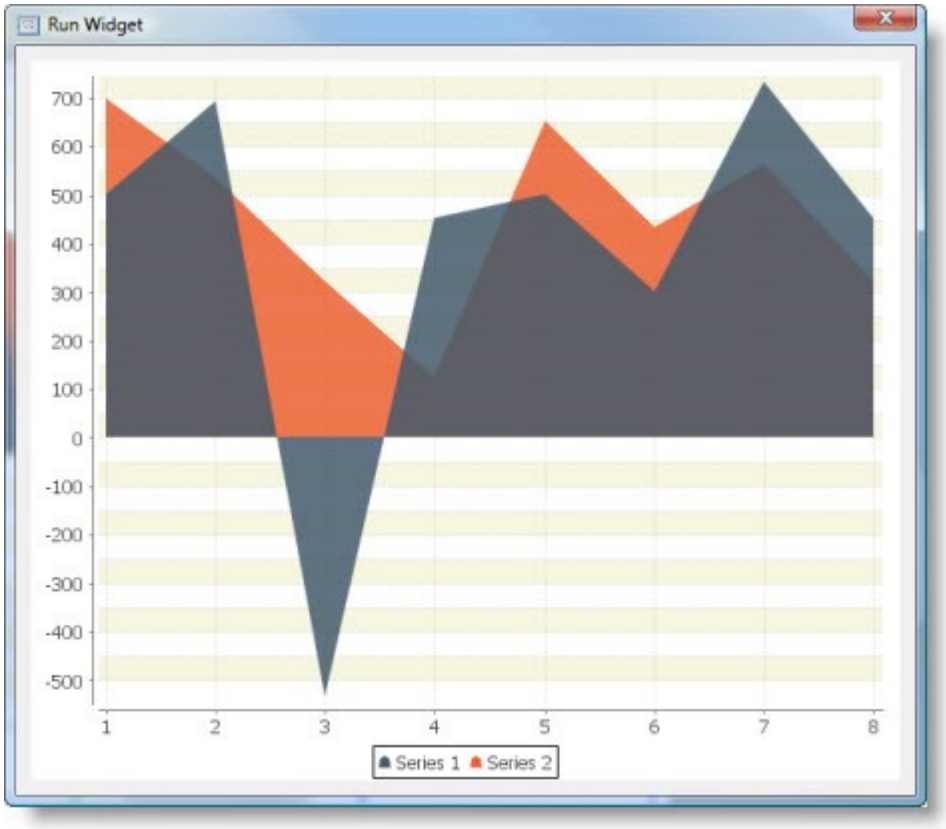


13.4.11.4.2 XY-Диаграмма с областями

XY-Диаграмма с областями представляет элементы при помощи многоугольников, заполняющих пространство между осью параметров и точками графика (X, Y).

XY-Диаграмма с областями основана на координатной области построения 1197 и координатном отрисовщике 1234. Она наследует все их свойства.

XY-Диаграмма с областями выглядит следующим образом:



XY-Диаграмма с областями поддерживает четыре вида отрисовщика:

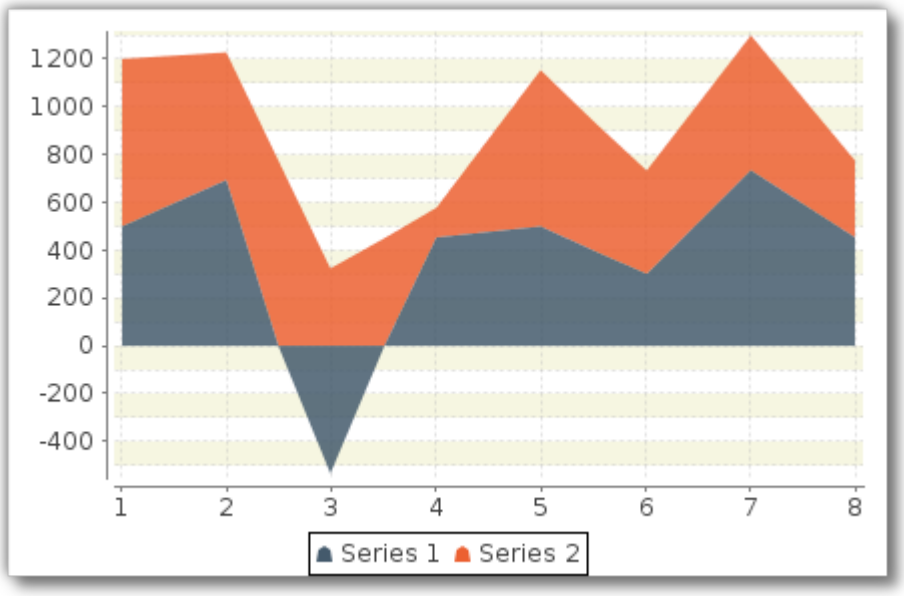
ОТРИСОВЩИК С ОБЛАСТЬЮ

Отображает каждый элемент в массиве данных (X, Y) при помощи многоугольника в пространстве между осью x и точкой графика.

Представленный ниже график использует ось дат 1168 в качестве оси x.

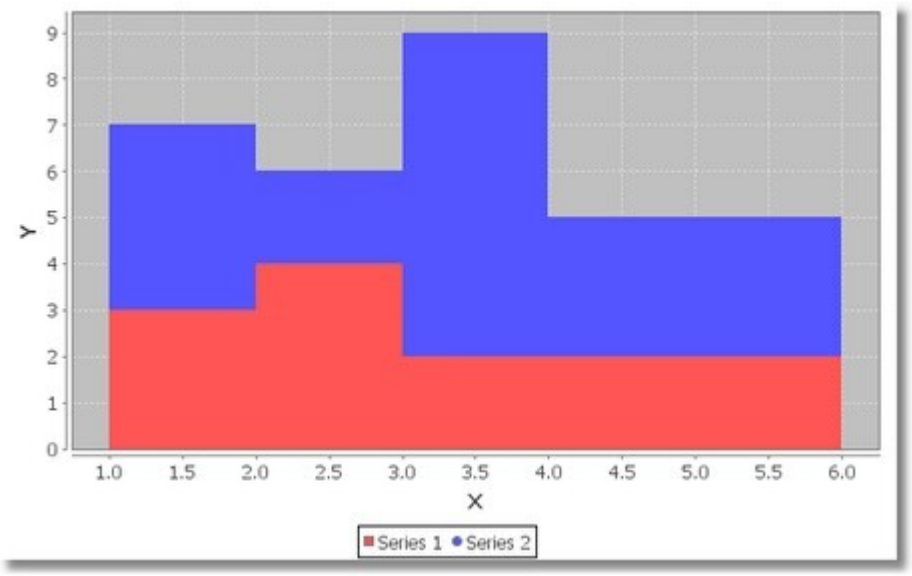
СОСТАВНОЙ ОТРИСОВЩИК

Заполняет область под серией данных и размещает каждую серию друг над другом.



СТУПЕНЧАТЫЙ ОТРИСОВЩИК

Отображает данные из массива (X, Y) в виде заполненной области под "ступенями".



РАЗНОСТНЫЙ ОТРИСОВЩИК

Выделяет различие между элементами двух серий, заполняя области между линиями каждой серии. Цвет заливки меняется между "положительным" (когда серия 1 больше серии 2) и "отрицательным" (когда серия 1 меньше серии 2).

На графике ниже ось дат 1168 используется в качестве оси x.

Массив данных

XY-Диаграмма с областями поддерживает несколько моделей данных:

- Пользовательские данные 1057
- Данные, основанные на событии 1064
- Данные, основанные на переменной 1061

Имеет следующие привязки исходных данных:

Привязка	Ожидаемый тип значения	Описание
Серия	строка	Текстовое имя серии данных.
X	число	Числовое значение, отображаемое вдоль оси определения (X).
Y	число	Числовое значение, отображаемое вдоль оси измерений (Y).

Дополнительные свойства массива данных:

АВТОСОРТИРОВКА

Флажок, контролирующий автоматическое упорядочивание элементов в серии данных по возрастанию относительно значения x.

Имя свойства: **autoSort**

Тип свойства: **логическое**

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Непрозрачный 1275, Фон 1275, Рамка 1275

Все общие свойства графиков 1160.

Все свойства, относящиеся к данным 1053.

Все свойства координатной области построения 1197.

Все свойства координатного отрисовщика 1234.

Пользовательские свойства

ОКАНТОВКА

Флажок, контролирующий начертание контура.

Данное свойство пригодно для отрисовщиков областей и ступеней.

Имя свойства: **outline**

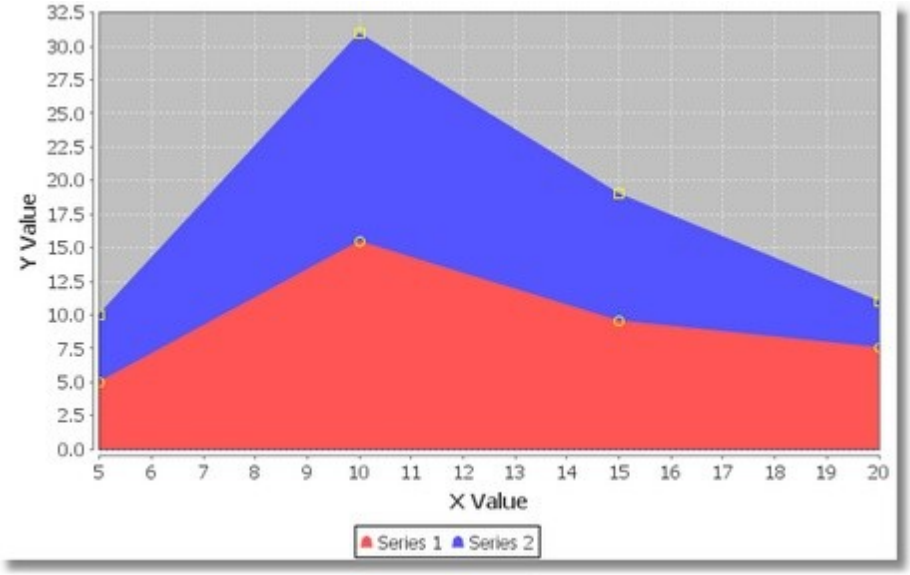
Тип свойства: **логическое**

РЕЖИМ ОТРИСОВЩИКА ОБЛАСТЕЙ

Отрисовщик областей может работать в нескольких режимах:

- область (обычный режим)
- область и формы (отображаются фигуры элемента данных)
- формы и линии (отображаются только формы элементов данных и линии, соединяющие элементы, области не заполняются; график похож на координатный линейный график 1100)
- линии (отображаются только линии, соединяющие элементы, области не заполняются; график похож на координатный линейный график 1100)
- формы (отображаются только формы элементов данных, области не заполняются; график похож на координатный линейный график 1100)

Пример режима "Область и формы":



Данное свойство пригодно для отрисовщика областей.

Имя свойства: **XYAreaRendererMode**

Тип свойства: **целочисленное**

ОКРУГЛЯТЬ X-КООРДИНАТЫ

Флажок, контролирующий, будут ли округлены координаты x (в пространстве отображения).

Данное свойство пригодно для отрисовщика стеков и различия.

Имя свойства: **roundXCoordinates**

Тип свойства: **логическое**

ОБЛАСТЬ ПОСТРОЕНИЯ

Флажок, контролирующий, будет ли заполнена область под ступенями.

Данное свойство пригодно для отрисовщика ступеней.

Имя свойства: **plotArea**

Тип свойства: **логическое**

ВИДИМОСТЬ ФОРМ

Флажок, контролирующий отображение форм в каждой точке графика.

Данное свойство пригодно для отрисовщика ступеней.

Имя свойства: **shapesVisible**

Тип свойства: **логическое**

ЗАЛИВКА ФОРМ

Флажок, контролирующий заполнение (видимых) форм.

Данное свойство пригодно для отрисовщика ступеней.

Имя свойства: **shapesFilled**

Тип свойства: **логическое**

ОСНОВАНИЕ

Основное значение для отрисовщика. Значение по умолчанию равно 0.0.

Данное свойство пригодно для отрисовщика ступеней.

Имя свойства: **rangeBase**

Тип свойства: **плавающее**

ОКРАСКА ПОЛОЖИТЕЛЬНОЙ РАЗНОСТИ

Цвет, используемый для заливки области между сериями данных 1 и 2 при положительном различии.

Данное свойство пригодно для отрисовщика различия.

Имя свойства: **positivePaint**

Тип свойства: **таблица данных**

ОКРАСКА ОТРИЦАТЕЛЬНОЙ РАЗНОСТИ

Цвет, используемый для заливки области между сериями данных 1 и 2 при отрицательном различии.

Данное свойство пригодно для отрисовщика различия.

Имя свойства: **negativePaint**

Тип свойства: **таблица данных**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

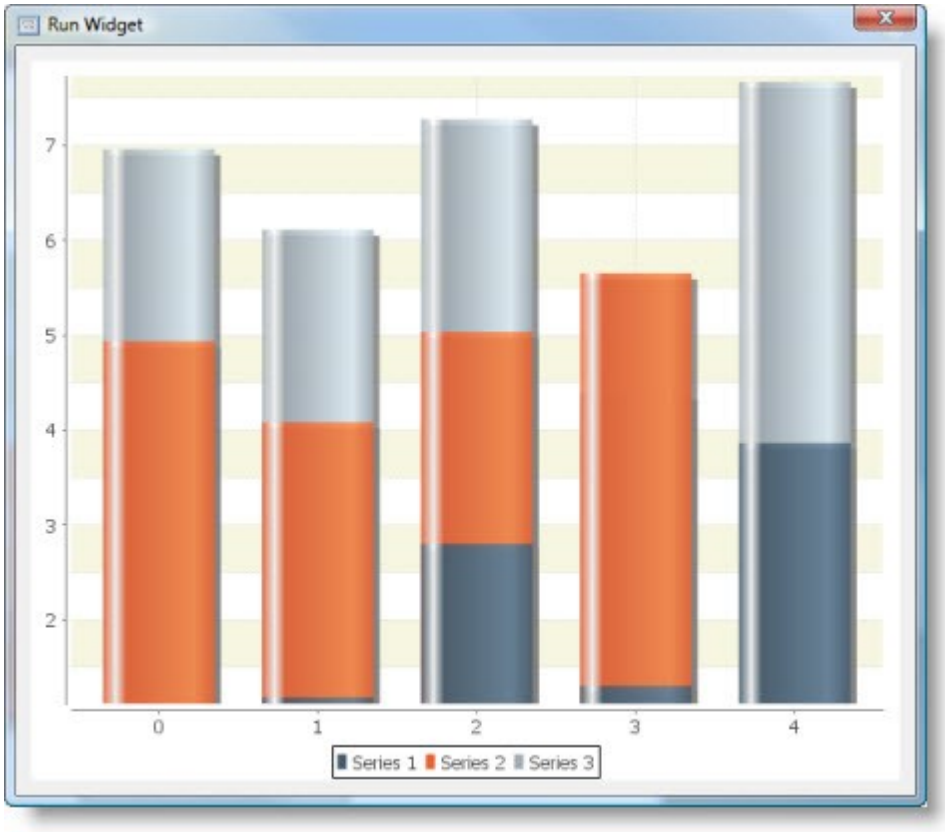
Все соответствующие события графика 1182.

13.4.11.4.3 XY-Столбчатая диаграмма

XY-Столбчатая диаграмма представляет элементы данных (X, Y) в виде прямоугольных столбцов.

XY-Столбчатая диаграмма основана на координатной области построения 1197 и координатном отрисовщике 1234. Она наследует все их свойства.

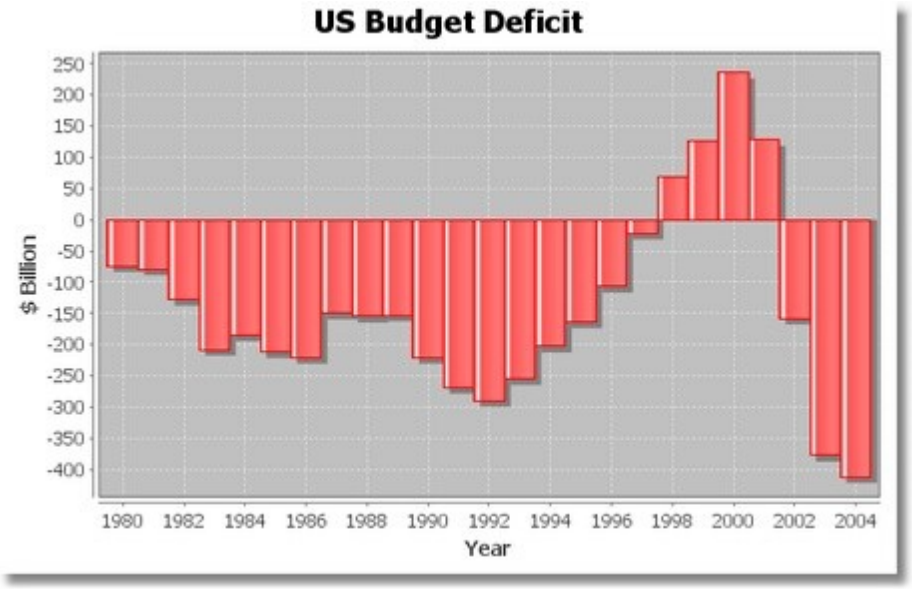
XY-Столбчатая диаграмма выглядит следующим образом:



XY-Столбчатая диаграмма поддерживает три отрисовщика:

ПРОСТОЙ

"Классический" отрисовщик столбцов.

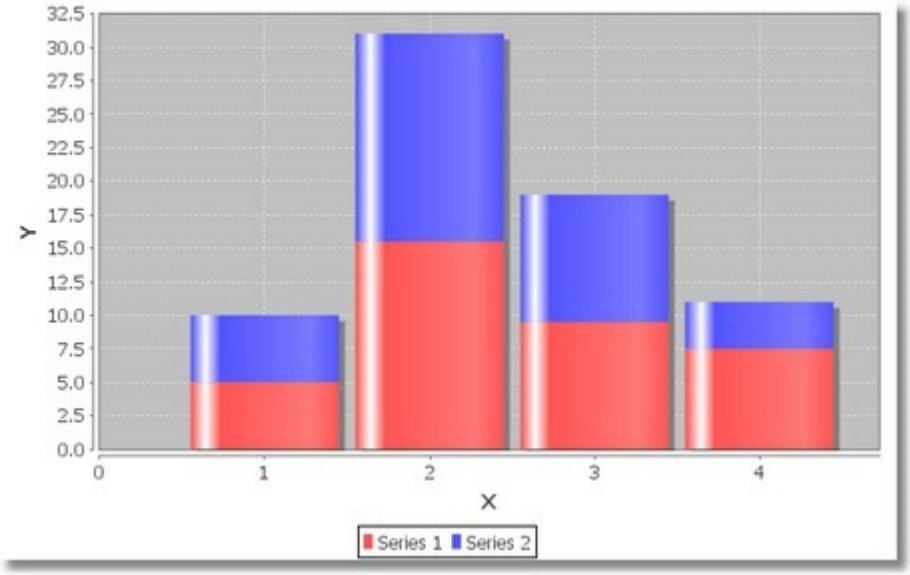


Приведенный ниже пример использует ось дат[1168] в качестве оси x.

СОСТАВНОЙ ОТРИСОВЩИК

Отрисовщик для отображения столбцов графика стеками на координатной области построения.

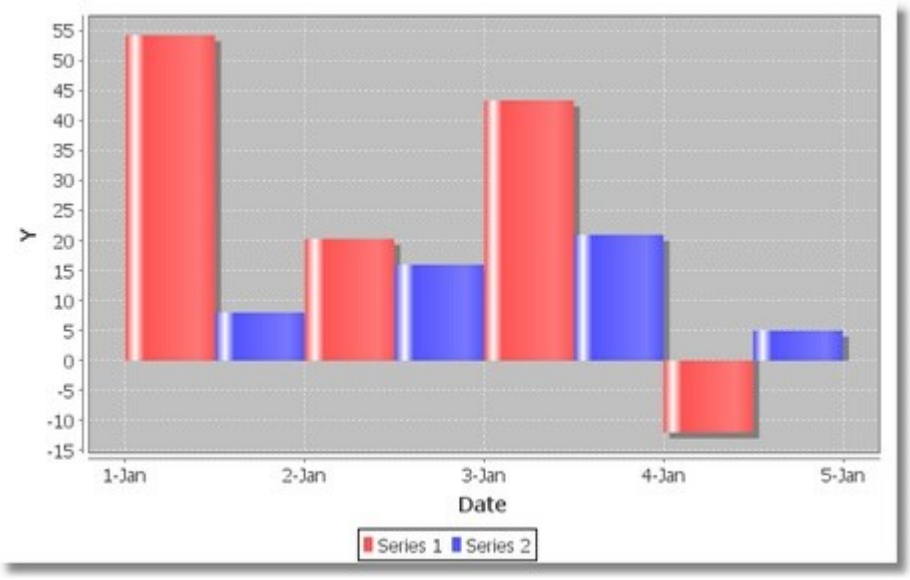
Все серии данных должны использовать одни и те же значения x, чтобы значения отображались стеками.



КЛАСТЕРНЫЙ ОТРИСОВЩИК

Данный отрисовщик немного отличается от простого тем, что он настраивает положение и ширину каждого столбца, помещая столбцы всех серий данных в пределах одного интервала x.

Приведенный ниже пример использует ось дат[1168] в качестве оси x.



Массив данных

XY-Столбчатая диаграмма поддерживает несколько моделей данных:

- Пользовательские данные[1057]
- Данные, основанные на событии[1064]
- Данные, основанные на переменной[1061]

Имеет следующие привязки исходных данных:

Привязка	Тип ожидаемого значения	Описание
Серия	строка	Текстовое имя серии данных, представленной на графике в виде линии (или набора фигур).
X	число	Точка привязки столбца вдоль оси определений (X). **Не используется**, если Фактор выравнивания столбца является отрицательным. Для более подробной информации

		обратитесь к разделу Фактор выравнивания столбцов 1115.
Нижнее X значение	число	Нижняя граница столбца вдоль оси определений (X) **или** часть ширины столбца. Для более подробной информации обратитесь к разделу Фактор выравнивания столбцов 1115.
Верхнее X значение	число	Верхняя граница столбца вдоль оси определений (X) **или** часть ширины столбца. Для более подробной информации обратитесь к разделу Фактор выравнивания столбцов 1115.
Нижнее Y значение	число	Начало столбца на оси измерений (Y). **Не используется**, если отключено свойство Использовать интервал Y 1115.
Верхнее Y значение	число	Окончание столбца на оси измерений (Y).

Дополнительные свойства массива данных:

АВТОСОРТИРОВКА

Флажок, контролирующий автоматическое упорядочивание элементов в серии данных в восходящем порядке по x-значению.

Имя свойства: **autoSort**

Тип свойства: **логическое**

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Непрозрачный 1275, Фон 1275, Рамка 1275

Все общие свойства графиков 1160.

Все свойства, относящиеся к данным 1053.

Все свойства координатной области построения 1197.

Все свойства координатного 1234 рендера 1234.

Тени столбцов

ВИДИМОСТЬ ТЕНИ

Флажок, контролирующий, будут ли отображаться тени столбцов.

Имя свойства: **shadowsVisible**

Тип свойства:**логическое**

ОКРАСКА ТЕНИ

Цвет 1279 заливки тени столбца.

Имя свойства: **shadowPaint**

Тип свойства: **таблица данных**

СМЕЩЕНИЕ ТЕНИ ПО X

X-смещение тени столбца.

Имя свойства: **shadowXOffset**

Тип свойства: **плавающее**

СМЕЩЕНИЕ ТЕНИ ПО Y

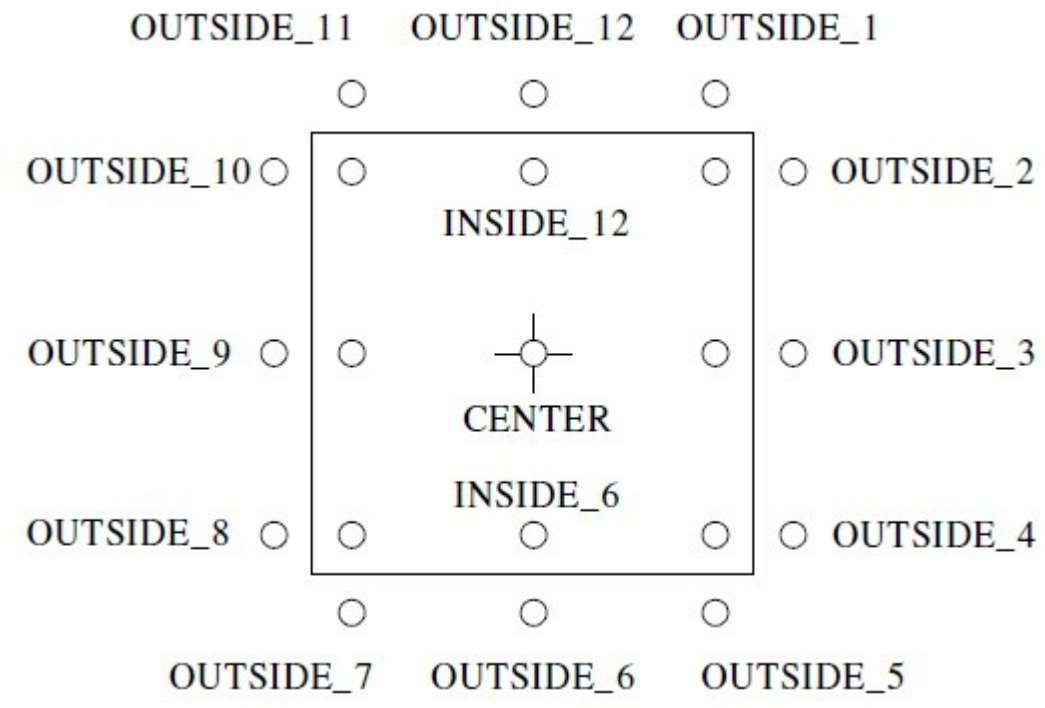
Y-смещение тени столбца.

Имя свойства: **shadowYOffset**

Тип свойства: **плавающее**

Метки элементов

Так как столбцы имеют прямоугольную форму, отрисовщик рассчитывает точки привязки, располагаемые, как указано далее на рисунке. Имейте в виду, что числа занимают (приблизительно) позицию часов на циферблате.



Если метка элемента располагается внутри столбца, рендер рассчитывает, будет ли достаточным размер столбца для вмещения текста. Если нет, рендер проверяет, определена ли для метки "альтернативная" позиция. Если таковая назначена, метка располагается согласно указанной позиции, в противном случае, метка не отображается. Могут быть определены две "альтернативные" позиции, одна для положительных значений, другая для отрицательных (примером для данного случая может послужить ситуация, когда метки положительных значений не помещаются внутри столбца и отображаются сверху столбца, а метки неподходящих по размеру отрицательных значений - снизу).

АЛЬТЕРНАТИВНОЕ ПОЛОЖЕНИЕ МЕТОК ЭЛЕМЕНТОВ

Данное свойство включает в себя два значения:

- **Положительная позиция метки элемента** [1172]: альтернативная позиция для положительных меток элементов. Установите пустое значение, если вы не хотите, чтобы метки не отображались, если они не помещаются внутри столбца.
- **Отрицательная позиция метки элемента**: альтернативная позиция для отрицательных меток элементов. Установите пустое значение, если вы не хотите, чтобы метки не отображались, если они не помещаются внутри столбца.

Имя свойства: **itemLabelPositionFallback**

Тип свойства: **таблица данных**

Другие свойства

ГРАНИЦА

Граница рендера. Граница определяется в виде процентного отношения к ширине столбца (например, 0.10 - 10 процентов) и является величиной, которая убавляется от ширины столбца перед отображением столбца на графике.

Имя свойства: **margin**

Тип свойства: **плавающее**

ОКАНТОВКА СТОЛБЦОВ

Флажок, контролирующий отображение контура вокруг каждого столбца. Цвет и тип штриха контура определяется при помощи свойств основного рендера столбцов.

Имя свойства: **drawBarOutline**

Тип свойства: **логическое**

ОСНОВАНИЕ

Основное значение для столбцов.

По умолчанию рендер отображает столбец между нулевым (основное значение) и значением данных отображаемого элемента. Некоторые определенные рендеры солбцов требуют ненулевое основное значение.

Имя свойства: **base**

Тип свойства: **плавающее**

ИСПОЛЬЗОВАТЬ Y-ИНТЕРВАЛ

Флажок, контролирующий определение длины столбца. Если значение является истинным, используется y-интервал из массива данных. Если является ложным, y-значение из массива данных определяет один конец столбца, а свойство "Основа" -- другой.

Имя свойства: **useYInterval**

Тип свойства: **логический**

КОЭФФИЦИЕНТ ВЫРАВНИВАНИЯ СТОЛБЦОВ

Коэффициент выравнивания столбца. Если коэффициент выравнивания столбца находится в пределах от 0.0 до 1.0, ширина столбца определяется x-интервалом из массива данных, в то время как позиция столбца настраивается так, что x-значение из массива данных приходится на заданную относительную позицию внутри столбца (например, 0.0 располагает столбец так, что x-значение находится слева от столбца, 0.5 -- x-значение находится по центру, 1.0 - x-значение расположено справа). Если коэффициент выравнивания столбца находится вне предела от 0.0 до 1.0, отрисовщик не будет осуществлять выравнивание (в данном случае позиция столбца определяется только x-интервалом в массиве данных, а x-значение игнорируется).

Имя свойства: **barAlignmentFactor**

Тип свойства: **плавающее**

РЕДАКТОР СТОЛБЦОВ

Редактор столбцов отвечает за актуальное отображение отдельных столбцов и имеет следующие свойства:

Свойство	Имя	Тип	Описание
Тип	type	целое	Тип редактора: **Стандартный** или **Градиентный**: • Стандартный редактор столбцов использует один цвет[1279] (текущий цвет серии данных) для заливки столбцов. • Градиентный редактор столбцов применяет эффект в нескольких областях, чтобы придать столбцам более интересный вид.
G1	g1	плавающее	Точка деления между первой и второй градиентными областями (больше 0.0).
G2	g2	плавающее	Точка деления между второй и третьей градиентными областями (больше, чем G1).
G3	g3	плавающее	Точка деления между третьей и четвертой градиентными областями (больше, чем G2 и меньше 1.0).

Имя свойства: **barPainter**

Тип свойства: **таблица данных**

СТОЛБЕЦ ЛЕГЕНДЫ

Фигура[1283], используемая для представления столбца в каждом элементе легенды.

Имя свойства: **legendBar**

Тип свойства: **таблица данных**

Другие свойства

ОТОБРАЖАТЬ В ВИДЕ ПРОЦЕНТОВ

Флажок, контролирующий отображение значений в виде процентного отношения к целому по всем сериям данных.

Пример графика, который отображает значения в процентах:

Данное свойство пригодно для составного отрисовщика.

Имя свойства: **renderAsPercentages**

Тип свойства: **логическое**

ЦЕНТРИРОВАТЬ СТОЛБЕЦ ПО НАЧАЛЬНОМУ ЗНАЧЕНИЮ

Флажок, контролирующий смещение группы столбцов таким образом, что её центр приходится на начальное x-значение, полученное из массива данных.

Имя свойства: **centerBarAtStartValue**

Тип свойства: **логическое**

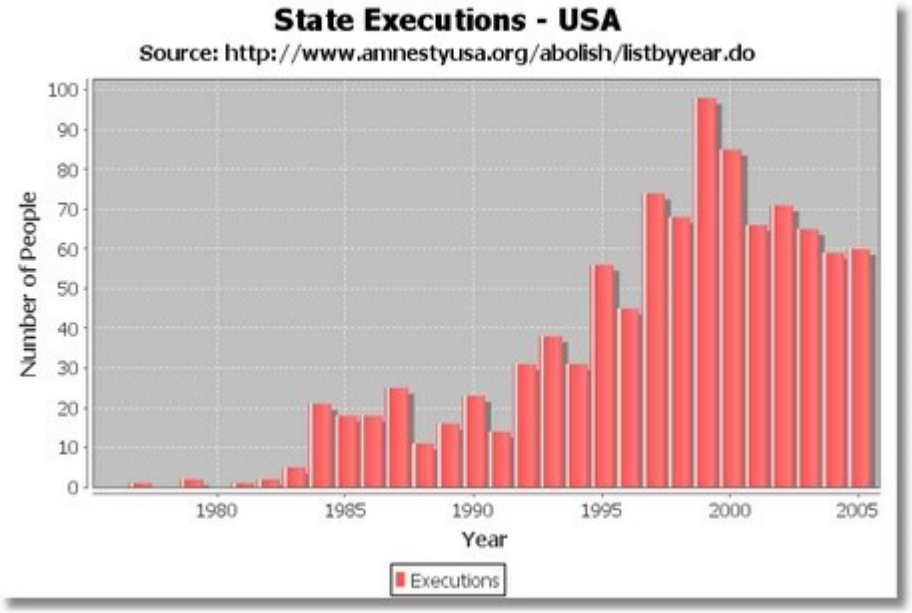
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

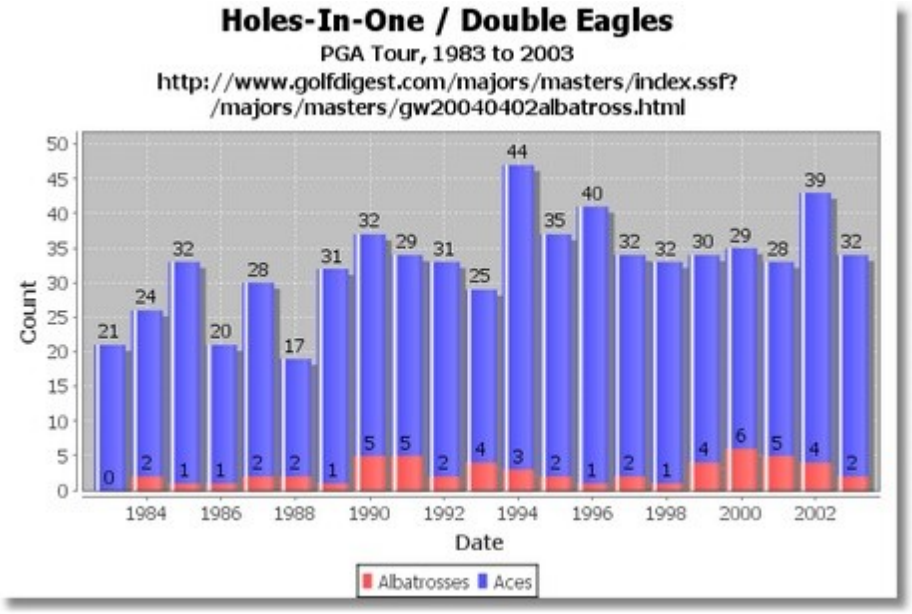
Все соответствующие события графика 1182.

Дополнительные примеры

XY-Столбчатая диаграмма, созданная при помощи оси измерений типа "Ось дат". Диаграмма использует субтитры для указания источника данных:



XY-Столбчатая диаграмма, основанная на оси определений дат:



График, отображающий меняющиеся ширины столбцов. **Коэффициент выравнивания столбцов** является отрицательным, а X координаты левой и правой сторон столбца основаны на значениях **X нижней** и **X верхней** из массива данных:

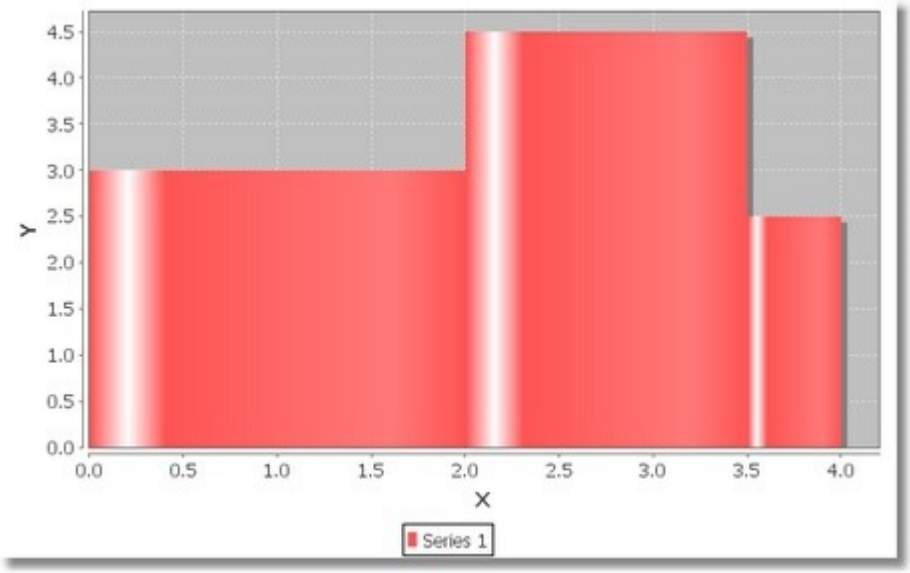
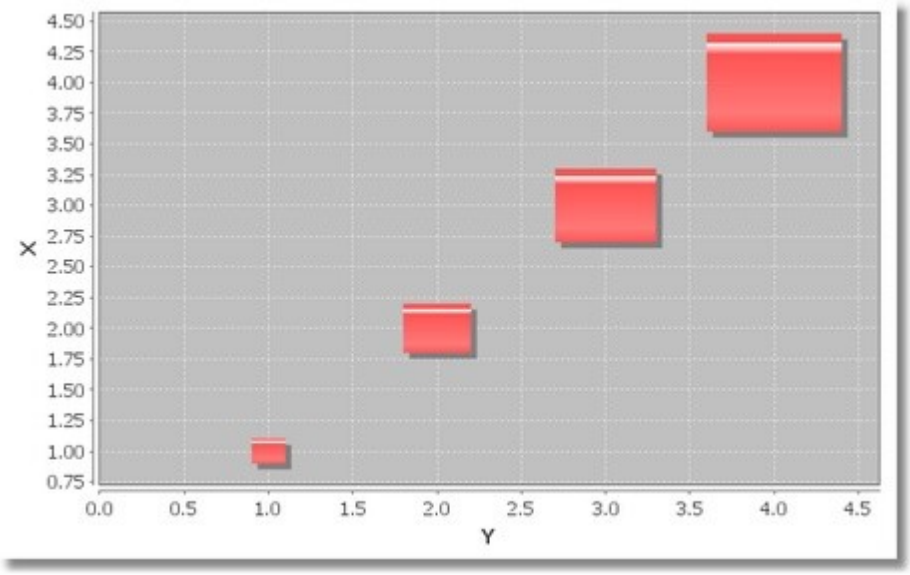
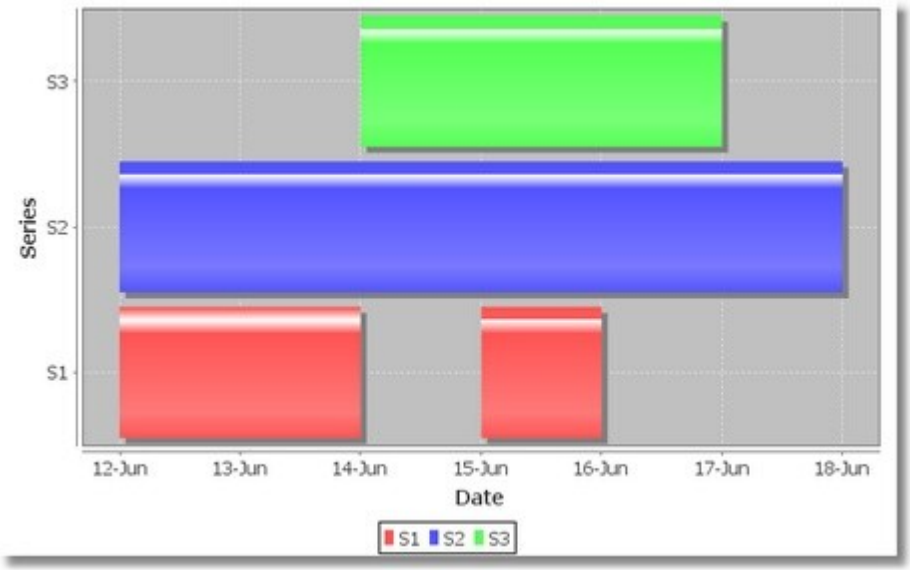


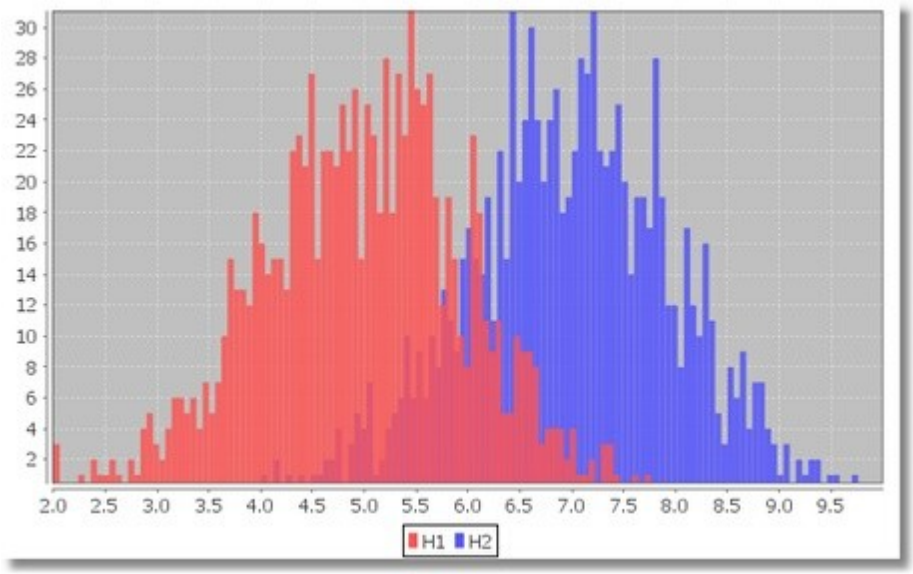
График с "плавающими" столбцами. **Коэффициент выравнивания столбцов** является отрицательным, а X координаты левой и правой сторон столбца основаны на значениях **X нижней** и **X верхней** из массива данных. **Использовать интервал Y** включено, поэтому столбцы отображаются из значений массива данных **Y нижней** и **Y верхней**:



Столбчатая диаграмма, которая отображает различные интервалы вдоль оси дат X:



Простая столбчатая диаграмма с двумя сериями данных, построенная при помощи координатного столбчатого графика:

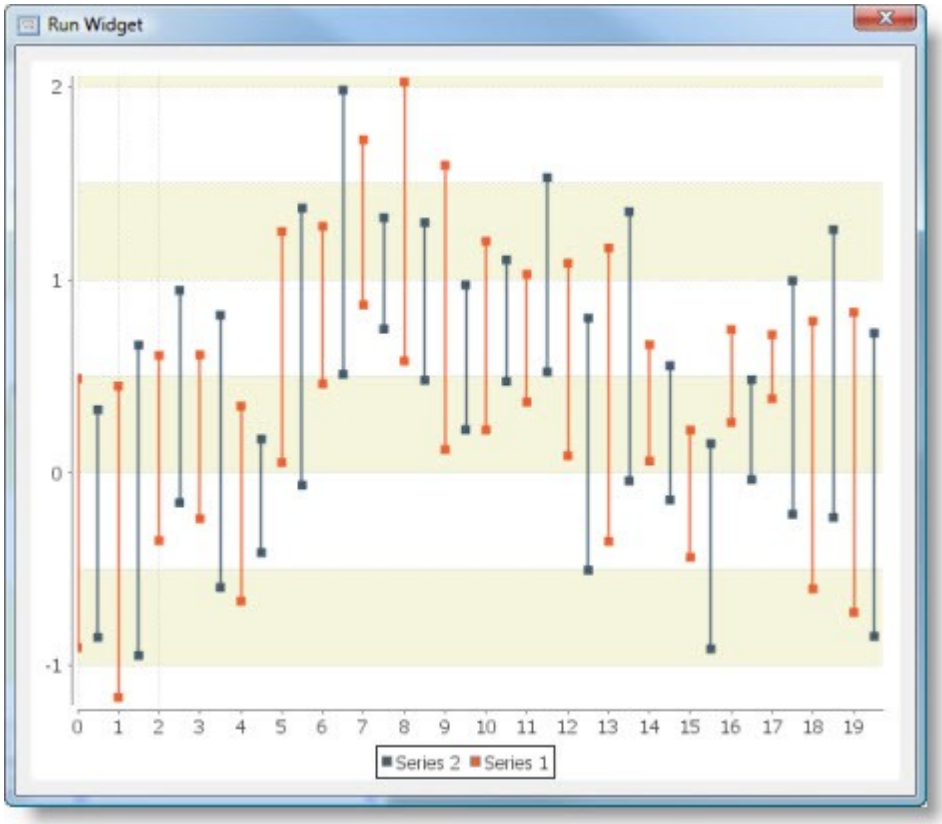


13.4.11.4.4 Диаграмма интервалов

Диаграмма интервалов отображает линии, указывающие y-интервал, соответствующий каждому x-значению.

Диаграмма интервалов основана на координатной области построения 1197 и координатном отрисовщике 1234. Она наследует все их свойства.

Диаграмма интервалов выглядит следующим образом:



Массив данных

Диаграмма интервалов поддерживает только Пользовательские данные 1057.

Она имеет следующие Привязки исходных данных:

Привязка	Тип ожидаемого значения	Описание
Серия данных	строка	Текстовое имя серии данных.
X	число	Числовое значение, отображаемое вдоль оси области определений (X).
нижнее Y значение	число	Нижняя граница Y-интервала.
верхнее Y значение	число	Верхняя граница Y-интервала.

Дополнительные свойства массива данных:

АВТОСОРТИРОВКА

Флажок, контролирующий, будут ли элементы серии данных автоматически отсортированы в порядке возрастания по x-значению.

Имя свойства: **autoSort**

Тип свойства: **Логическое**

РАЗРЕШИТЬ ДУБЛИРОВАНИЕ X-ЗНАЧЕНИЙ

Флажок, контролирующий, могут ли два и более элементов в серии данных иметь одно и то же x-значение.

Имя свойства: **allowDuplicateXValues**

Тип свойства: **Логическое**

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Непрозрачный 1275, Фон 1275, Рамка 1275

Все общие свойства графиков 1160.

Все свойства, относящиеся к данным 1053.

Все свойства координатной области построения 1197.

Все свойства координатного отрисовщика 1234.

Пользовательские свойства

ДОПОЛНИТЕЛЬНЫЙ ГЕНЕРАТОР МЕТОК ЭЛЕМЕНТОВ

Дополнительный генератор метки элемента 1207. Если не пустой, созданная метка элемента будет отображаться около нижнего Y-значения в Отрицательной позиции Метки элемента.

Имя свойства: **lowItemLabelGenerator**

Тип свойства: **Таблица данных**

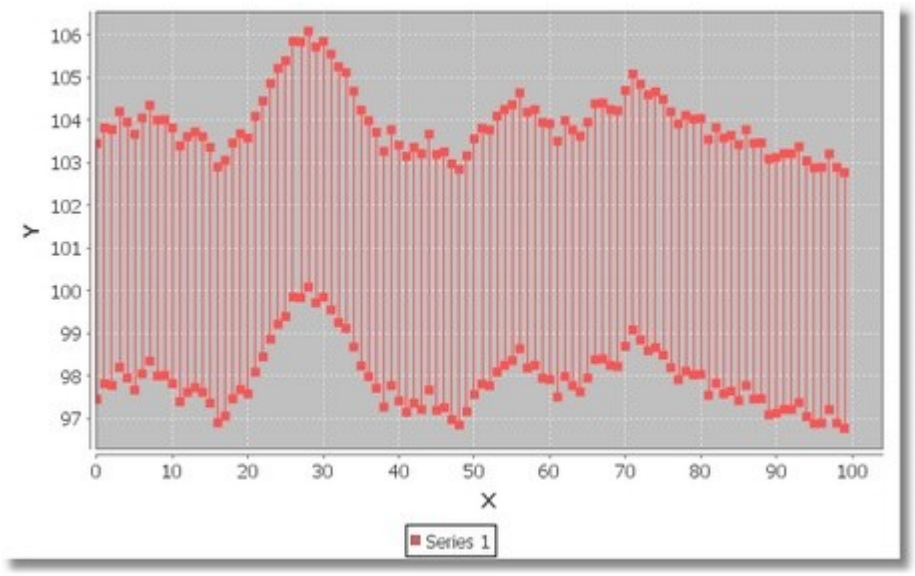
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

Все соответствующие события графика 1182.

Дополнительные примеры

Простая диаграмма интервалов с одной серией данных:

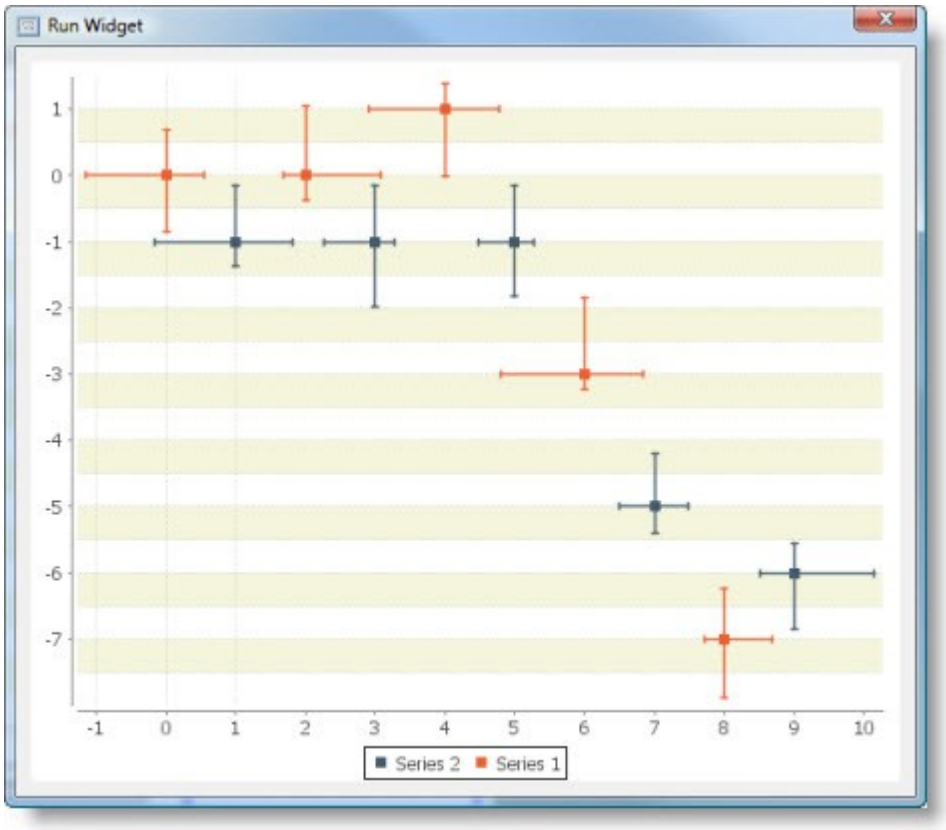


13.4.11.4.5 Диаграмма погрешностей

Диаграмма погрешностей отображает столбцы ошибок рядом с каждым элементом данных.

Диаграмма погрешностей основана на координатной области построения 1119 и координатном отрисовщике 1234. Она наследует все их свойства.

Диаграмма погрешностей выглядит следующим образом:



Массив данных

Диаграмма погрешностей поддерживает только пользовательские данные 1057.

Он имеет следующие привязки исходных данных:

Привязка	Тип ожидаемого значения	Описание
серия данных	строка	Текстовое имя серии данных.
X	число	Числовое значение, отображаемое вдоль оси области определений (X).
нижнее X значение	число	Нижняя граница X-интервала ошибки.
верхнее X значение	число	Верхняя граница X-интервала ошибки.
Y	число	Числовое значение, отображаемое вдоль оси измерений (Y).
нижнее Y значение	число	Нижняя граница Y-интервала ошибки.
верхнее Y значение	число	Верхняя граница Y-интервала ошибки.

Дополнительные свойства массива данных:

АВТОСОРТИРОВКА

Флажок, контролирующий, будут ли элементы серии данных автоматически отсортированы в порядке возрастания по x-значению.

Имя свойства: **autoSort**

Тип свойства: **Логическое**

РАЗРЕШИТЬ ДУБЛИРОВАНИЕ X-ЗНАЧЕНИЙ

Флажок, контролирующий, могут ли два и более элементов в серии данных иметь одно и то же x-значение.

Имя свойства: **allowDuplicateXValues**

Тип свойства: **Логическое**

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Непрозрачный 1275, Фон 1275, Рамка 1275

Все общие свойства графиков 1160.

Все свойства, относящиеся к данным 1053.

Все свойства координатной области построения 1197.

Все свойства координатного отрисовщика 1234.

Пользовательские свойства

РИСОВАТЬ X-ПОГРЕШНОСТЬ

Флажок, контролирующий, будут ли отображаться столбцы ошибки для x-значений в массиве данных.

Имя свойства: **drawXError**

Тип свойства: **Логическое**

РИСОВАТЬ Y-ПОГРЕШНОСТЬ

Флажок, контролирующий, будут ли отображаться столбцы ошибки для y-значений в массиве данных.

Имя свойства: **Логическое**

ДЛИНА КРЫШЕК

Длина горизонтальных линий-ограничителей на каждом конце столбцов ошибок для каждого значения данных.

Имя свойства: **capLength**

Тип свойства: **Плавающее**

ОКРАСКА ПОГРЕШНОСТИ

Цвет заливки 1279, используемый для отображения столбцов ошибки. Если отключено, ренедерер будет использовать цвет заливки серии данных.

Имя свойства: **errorPaint**

Тип свойства: **Таблица данных**

ШТРИХ ПОГРЕШНОСТИ

Штрих 1282, используемый для отображения столбцов ошибки. Если отключено, отрисовщик будет использовать штрих серии данных.

Имя свойства: **errorStroke**

Тип свойства: **Таблица данных**

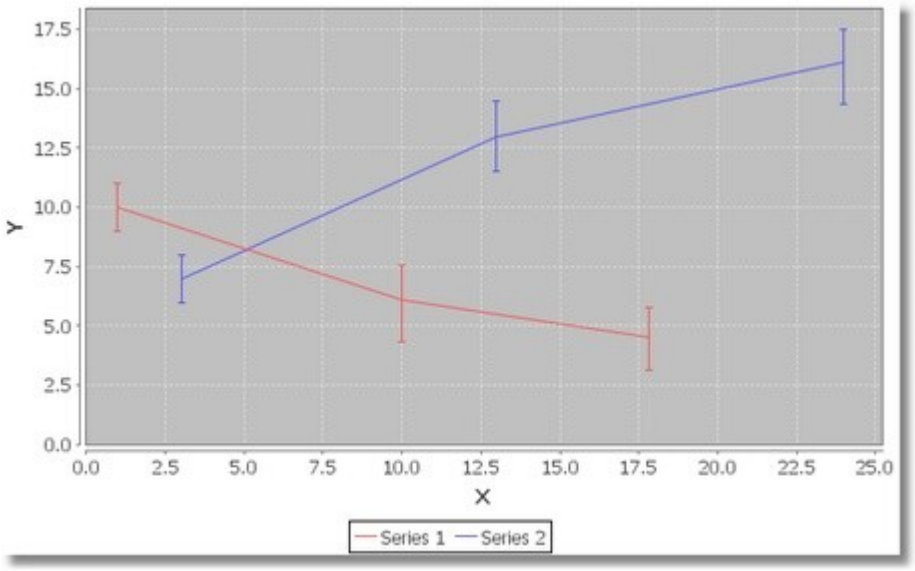
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1288, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

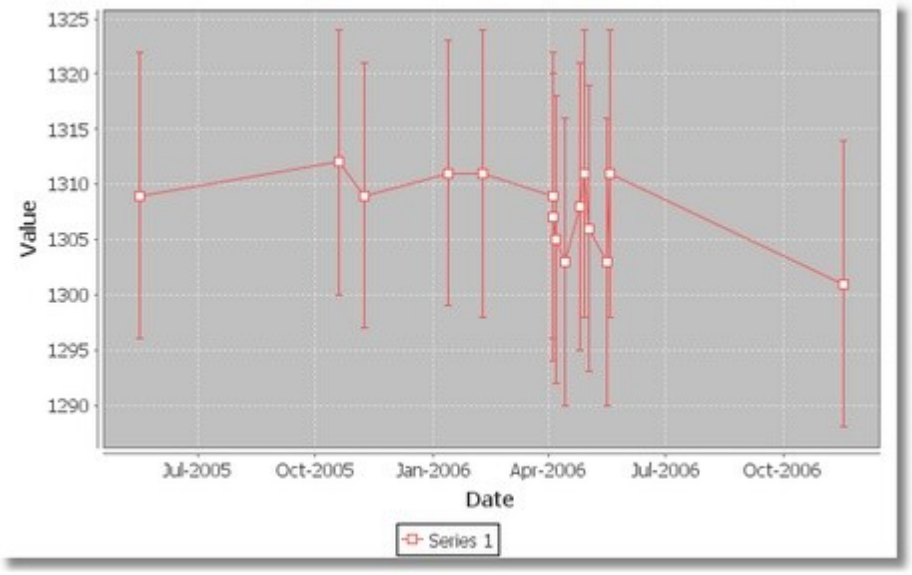
Все соответствующие события графика 1182.

Дополнительные примеры

Диаграмма погрешностей с отключенным свойством **Отображать X-ошибки** и включенным **Видимость линий по умолчанию**.



Приблизительно то же, что и выше, но с другими настройками:

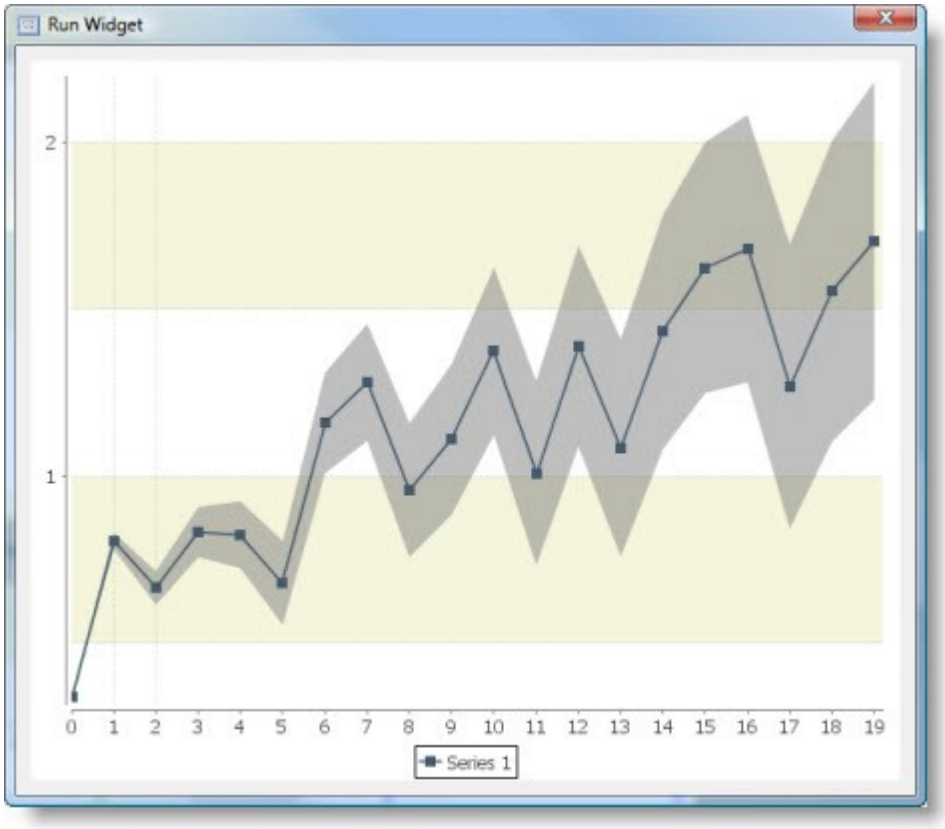


13.4.11.4.6 Диаграмма отклонений

Диаграмма отклонений выделяет диапазон y-значений на фоне серии данных, передаваемой в виде линии.

Диаграмма отклонений основана на координатной области построения 1197 и координатном отрисовщике 1234. Она наследует все их свойства.

Диаграмма отклонений выглядит следующим образом:



Массив данных

Диаграмма отклонений поддерживает только пользовательские данные 1057.

Она имеет следующие привязки исходных данных:

Привязка	Тип ожидаемого значения	Описание
Серия данных	строка	Текстовое имя серии данных.
X	число	Числовое значение, отображаемое вдоль оси области определений (X).
Y	число	Числовое значение, отображаемое вдоль оси измерений (Y).
Нижнее Y-значение	число	Нижняя граница Y-интервала.
Верхнее Y-значение	число	Верхняя граница Y-интервала.

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Непрозрачный 1275, Фон 1275, Рамка 1275

Все общие свойства графиков 1160.

Все свойства, относящиеся к данным 1053.

Все свойства координатной области построения 1197.

Все свойства координатного отрисовщика 1234.

Пользовательские свойства

АЛЬФА

Альфа-прозрачность для тени интервалов.

Имя свойства: **alpha**

Тип свойства: **Плавающее**

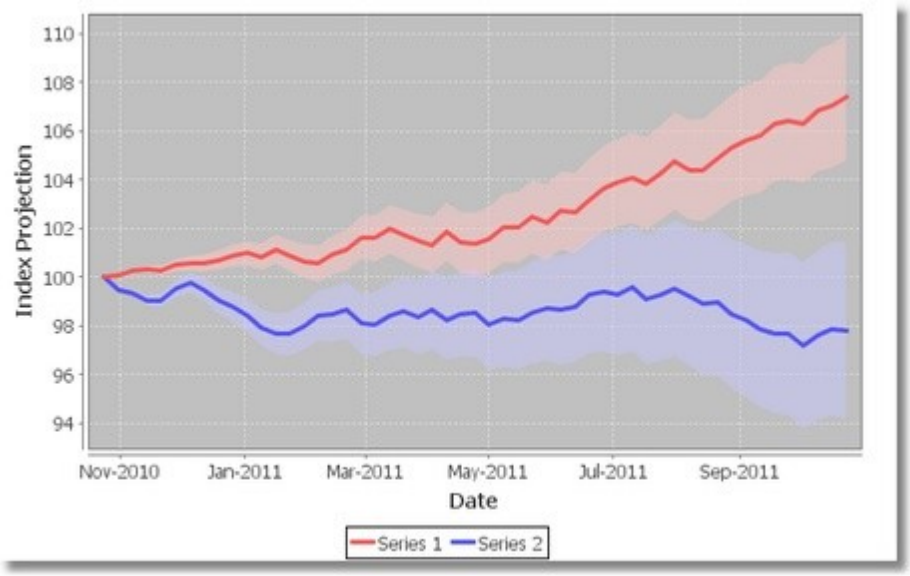
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

Все соответствующие события графика 1182.

Дополнительные примеры

Диаграмма отклонений с двумя сериями данных:

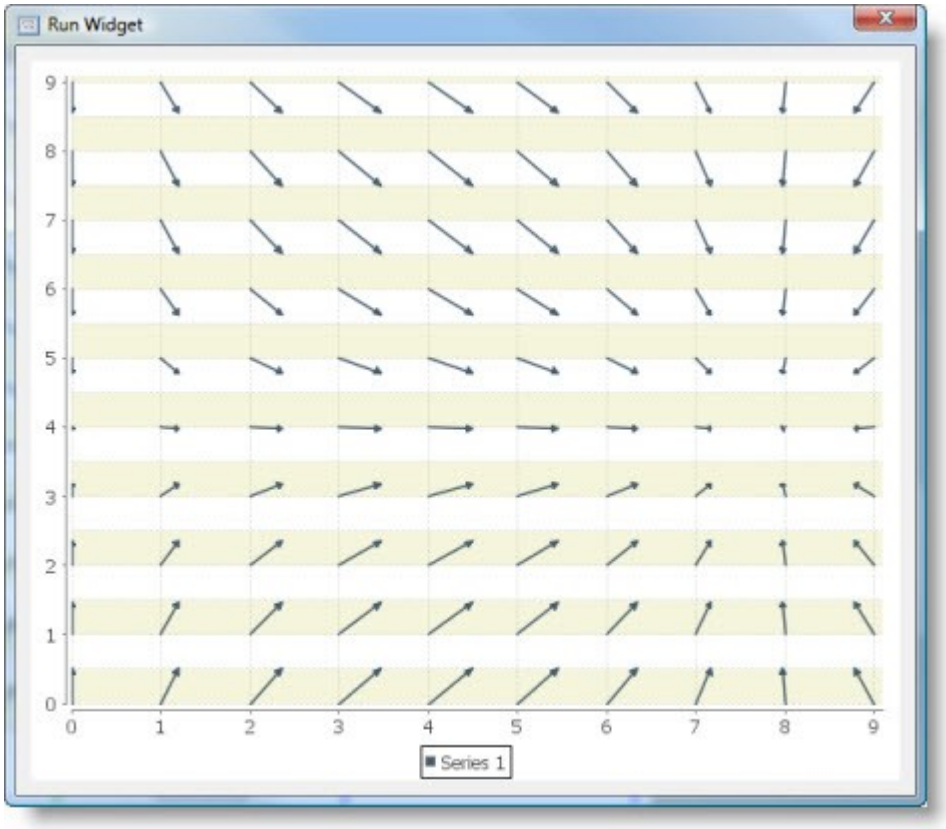


13.4.11.4.7 Векторная диаграмма

Векторная диаграмма отображает серии векторов в пространстве 2D. Каждый вектор описан значениями Дельты X и Дельты У.

Векторная диаграмма основана на координатной области построения[1197] и координатном отрисовщике[1234]. Она наследует все их свойства.

Векторная диаграмма выглядит следующим образом:



Массив данных

Диаграмма поддерживает только Пользовательские данные[1057].

Она имеет следующие привязки исходных данных:

Привязка	Тип ожидаемого значения	Описание
Серия данных	строка	Текстовое имя серии данных.
X	число	База вектора по оси определений (X).
Y	число	База вектора по оси измерения (Y).
Дельта X	число	Длина проекции вектора по оси определений X.
Дельта Y	число	Длина проекции вектора по оси измерений Y.

Дополнительные свойства массива данных:

АВТОСОРТИРОВКА

Флажок, контролирующий, будут ли элементы серии данных автоматически отсортированы в порядке возрастания по x-значению.

Имя свойства: **autoSort**

Тип свойства: **Логическое**

РАЗРЕШИТЬ ДУБЛИРОВАНИЕ X-ЗНАЧЕНИЙ

Флажок, контролирующий, могут ли два и более элементов в серии данных иметь одно и то же x-значение.

Имя свойства: **allowDuplicateXValues**

Тип свойства: **Логическое**

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Непрозрачный 1275, Фон 1275, Рамка 1275

Все Общие свойства графиков 1160.

Все свойства, относящиеся к данным 1053.

Все свойства координатной области построения 1197.

Все свойства координатного отрисовщика 1234.

Пользовательские свойства

Не определены.

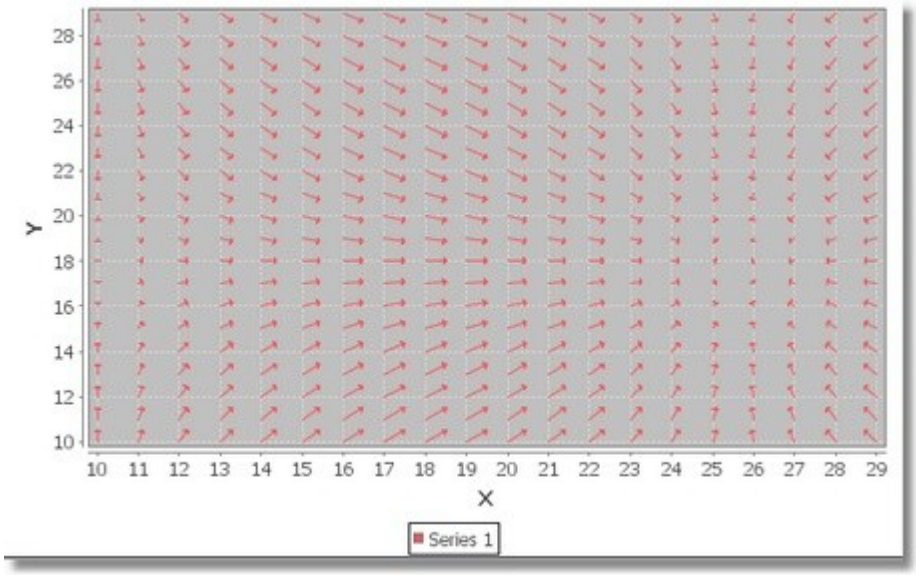
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

Все соответствующие события графика 1182.

Дополнительные примеры

Векторная диаграмма с большим массивом данных:

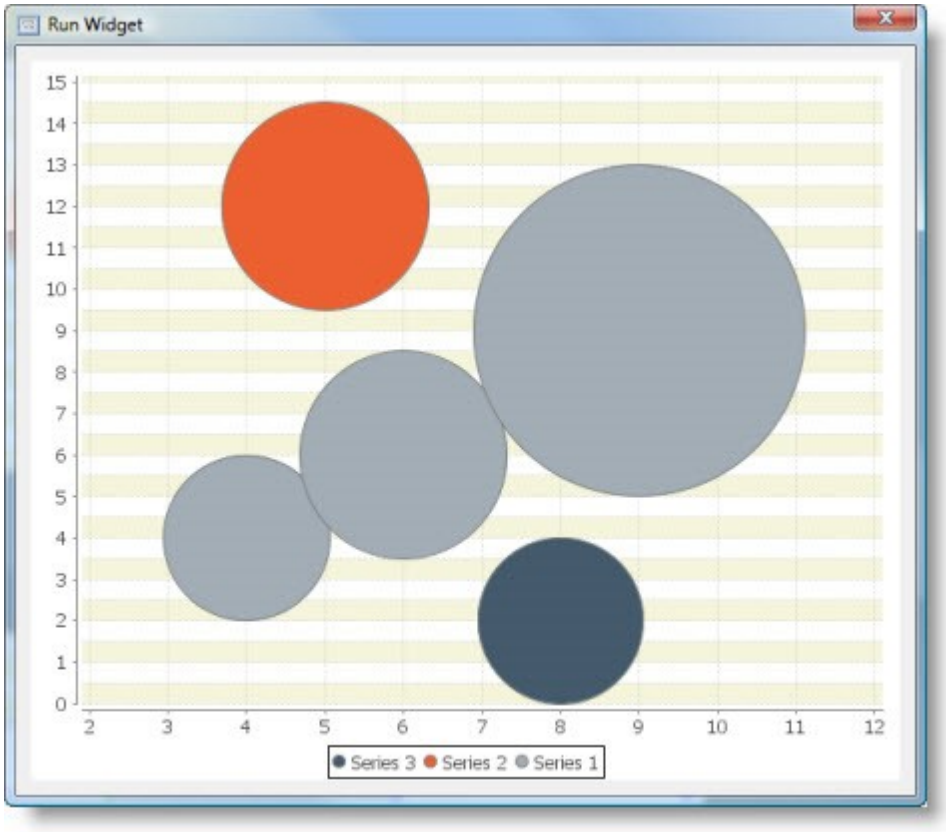


13.4.11.4.8 Пузырьковая диаграмма

Пузырьковая диаграмма представляет собой круги (пузырьки) различного размера (диаметра) в пространстве 2D.

Пузырьковая диаграмма основана на координатной области построения 1197 и координатном отрисовщике 1234. Она наследует все их свойства.

Пузырьковая диаграмма выглядит следующим образом:



Массив данных

Пузырьковая диаграмма поддерживает только пользовательские данные 1057.

Имеет следующие привязки исходных данных:

Привязка	Тип ожидаемого значения	Описание
Серия данных	строка	Текстовое имя серии данных.
X	число	Координата X (определений) центра пузырька.
Y	число	Координата Y (измерений) центра пузырька.
Z	число	Z-значение, которое может быть диаметром пузырька, его длиной вдоль оси X или длиной вдоль оси Y. Это зависит от свойства Тип шкалы.

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Непрозрачный 1275, Фон 1275, Рамка 1275

Все общие свойства графиков 1160.

Все свойства, относящиеся к данным 1053.

Все свойства координатной области построения 1197.

Все свойства координатного отрисовщика 1234.

Пользовательские свойства

ТИП ШКАЛЫ

Метод для определений размера пузырьков, отображаемых данным отрисовщиком:

- **Шкала на обеих осях**: пузырьки отображаются в виде эллипсисов, длина и ширина которых определяется z-значением, отмеченным по обеим осям;

- **Шкала на оси параметров**: пузырьки отображаются в виде кругов, диаметр которых определяется z-значением, отмеченным по оси параметров;
- **Шкала на оси значений**: пузырьки отображаются в виде кругов, диаметр которых определяется z-значением, отмеченным по оси значений.

Имя свойства: **scaleType**

Тип свойства: **Целое**

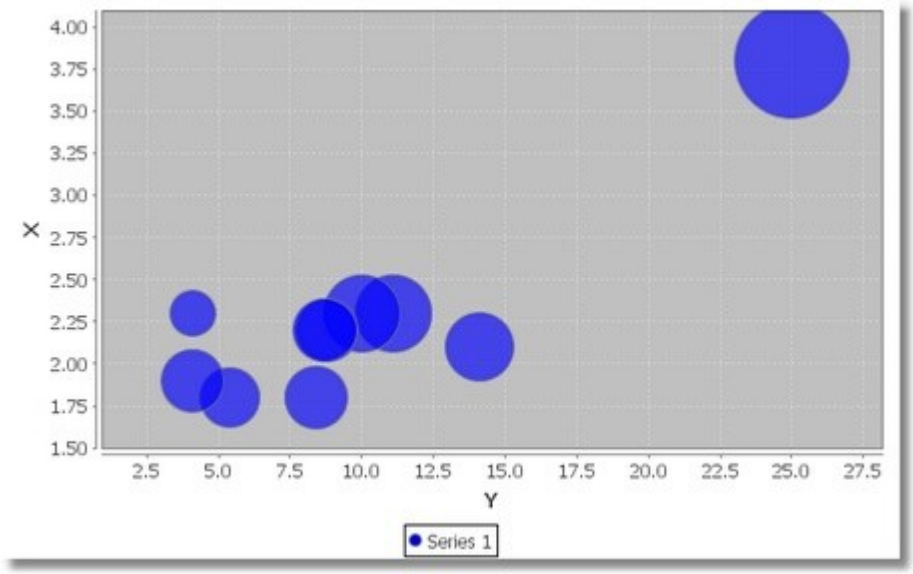
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1288, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

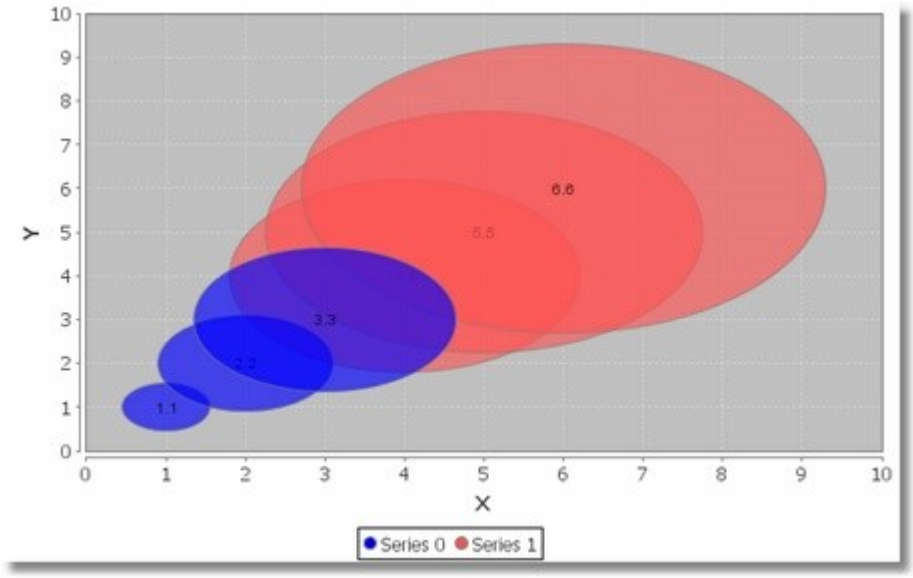
Все соответствующие события графика 1182.

Дополнительные примеры

Пример пузырькового графика с одной серией данных:



Пузырьковая диаграмма с метками элементов и **Типом шкалы**, установленным на "Шкала на обеих осях":



13.4.11.4.9 Финансовая диаграмма

Финансовая диаграмма график используется для построения графиков котировок Open-High-Low-Close (OHLC).

Финансовая диаграмма основана на координатной области построения 1197 и координатном отрисовщике 1234. Она наследует все их свойства.

Финансовая диаграмма выглядит следующим образом:

Финансовая диаграмма поддерживает два отрисовщика:

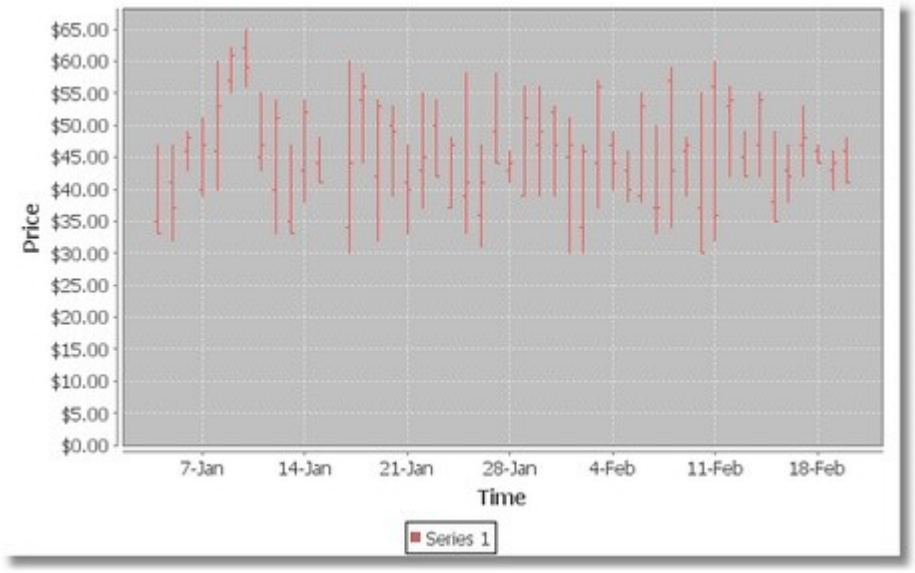
СВЕЧНОЙ ОТРИСОВЩИК

Отображает элемент из массива данных OHLC в виде столбца с исходящими из него сверху и снизу линиями. Японские свечи обычно применяются для отображения финансовых данных. Столбец представляет позиции открытия/закрытия цен, линии указывают самые высокие и низкие цены за период торговли (обычно за один день).

См. пример графика японских свечей на приведенном выше рисунке.

МАКС-МИН

Отображает элемент из массива данных OHLC при помощи линий, которые отмечают диапазон высоких и низких цен за период торговли, к ним добавляются еще метки для указания значений открытия/закрытия цен.



Массив данных

Финансовая диаграмма поддерживает только модель Пользовательские данные 1057.

Имеет следующие привязки исходных данных:

Привязка	Ожидаемый тип значения	Описание
Серия данных	строка	Текстовое имя серии данных
Дата	дата	Дата
Открытие	число	Значение открытия для даты
Верхнее	число	Верхнее значение для даты
Нижнее	число	Нижнее значение для даты
Закрытие	число	Значение закрытия для даты

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Непрозрачный 1275, Фон 1275, Рамка 1275

Все общие свойства графиков 1160.

Все свойства, относящиеся к данным 1053.

Все свойства координатной области построения 1197.

Все свойства координатного отрисовщика 1234.

Расчет ширины свечи

Если атрибут "Ширина свечи" является отрицательным, отрисовщик автоматически определяет ширину свечи во время построения графика.

После того, как ширина свечи рассчитана согласно методу автоматического определений ширины, она подвергается трем настройкам. Во-первых, вычитается интервал автоматической ширины (Auto Width Gap), затем ширина умножается на коэффициент автоматической ширины, и, наконец, ширина ограничивается согласно максимальной ширине свечи в миллисекундах.

МЕТОД АВТОВЫЧИСЛЕНИЯ ШИРИНЫ

Тип автоматического вычисления ширины свечи:

- **Средний**: основывает ширину на интервале по умолчанию между последовательными x-значениями в массиве данных

- **Наименьший**: основывает ширину на наименьшем интервале между последовательными x-значениями в массиве данных

Данное свойство пригодно для свечного отрисовщика.

Имя свойства: **autoWidthMethod**

Тип свойства: **Целое**

КОЭФФИЦИЕНТ АВТОШИРИНЫ

Фактор, на который умножается ширина свечи во время ее автоматического расчета.

Данное свойство пригодно для свечного отрисовщика.

Имя свойства: **autoWidthFactor**

Тип свойства: **Плавающее**

ПРОМЕЖУТОК АВТОШИРИНЫ

Количество оставляемого пространства с каждой стороны свечи.

Данное свойство пригодно для свечного отрисовщика.

Имя свойства: **autoWidthGap**

Тип свойства: **Плавающее**

ШИРИНА СВЕЧИ

Ширина каждой свечи. Если значение отрицательное, отрисовщик автоматически определяет ширину во время каждого перестроения графика.

Данное свойство пригодно для свечного отрисовщика.

Имя свойства: **candleWidth**

Тип свойства: **Плавающее**

МАКСИМАЛЬНАЯ ШИРИНА СВЕЧИ В МИЛЛИСЕКУНДАХ

Максимальная ширина свечи в миллисекундах. Значение по умолчанию равно 20 часам, что является разумным для отображения данных за день.

Данное свойство пригодно для свечного отрисовщика.

Имя свойства: **maxCandleWidthInMilliseconds**

Тип свойства: **Плавающее**

Другие свойства

ОКРАСКА ПОВЫШЕНИЯ

Цвет заливки 1279 свечей, у которых закрытие цены выше ее открытия (т.е. цена поднялась). Если данное свойство отключено (null), отрисовщик будет использовать обычный цвет заливки серии данных.

Данное свойство пригодно для свечного отрисовщика.

Имя свойства: **upPaint**

Тип свойства: **Таблица данных**

ОКРАСКА ПОНИЖЕНИЯ

Цвет заливки 1279 свечей, у которых закрытие цены ниже ее открытия (цена упала). Если данное свойство отключено (null), отрисовщик будет использовать обычный цвет заливки серии данных.

Данное свойство пригодно для свечного отрисовщика.

Имя свойства: **downPaint**

Тип свойства: **Таблица данных**

ИСПОЛЬЗОВАТЬ ОКАНТОВКУ

Данный отрисовщик может отобразить контур свечей, используя цвет заливки серии данных или цвет их контура.

Флажок Применить заливку контура контролирует, будет ли отрисовщик использовать цвет контура серии данных.

Данное свойство пригодно для свечного отрисовщика.

Имя свойства: **useOutlinePaint**

Тип свойства: **Логическое**

ОТОБРАЖАТЬ МЕТКИ ОТКРЫТИЯ

Флажок, контролирующий, будет ли отображена на графике метка открытия цены для каждого значения данных.

Данное свойство пригодно для отрисовщика гистограммы.

Имя свойства: **drawOpenTicks**

Тип свойства: **Логическое**

ОТОБРАЖАТЬ МЕТКИ ЗАКРЫТИЯ

Флажок, контролирующий, будет ли отображена на графике метка закрытия цены для каждого значения данных.

Данное свойство пригодно для отрисовщика гистограммы.

Имя свойства: **drawCloseTicks**

Тип свойства: **Логическое**

ОКРАСКА МЕТОК ОТКРЫТИЯ

Цвет заливки[1279], используемый для отображения метки открытия цены для каждого значения данных. Если данное свойство отключено (null), отрисовщик использует цвет серии данных.

Данное свойство пригодно для отрисовщика гистограммы.

Имя свойства: **openTickPaint**

Тип свойства: **Таблица данных**

ОКРАСКА МЕТОК ЗАКРЫТИЯ

Цвет заливки[1279], используемый для отображения метки закрытия цены для каждого значения данных. Если данное свойство отключено (null), отрисовщик использует цвет серии данных.

Данное свойство пригодно для отрисовщика гистограммы.

Имя свойства: **closeTickPaint**

Тип свойства: **Таблица данных**

ДЛИНА МЕТОК

Длина меток открытия и закрытия цены.

Данное свойство пригодно для отрисовщика гистограммы.

Имя свойства: **tickLength**

Тип свойства: **Плавающее**

Общие события

Скрытие[1285], Показ[1285], Перемещение[1285], Изменение размеров[1285], Клик мыши[1286], Нажатие кнопки мыши[1286], Отпускание кнопки мыши[1286], Вход мыши[1286], Выход мыши[1286], Перемещение мыши[1286], Вращение колесика мыши[1286], Печать клавиши[1287], Нажатие клавиши[1287], Отпускание клавиши[1287], Получение фокуса[1287], Потеря фокуса[1287]

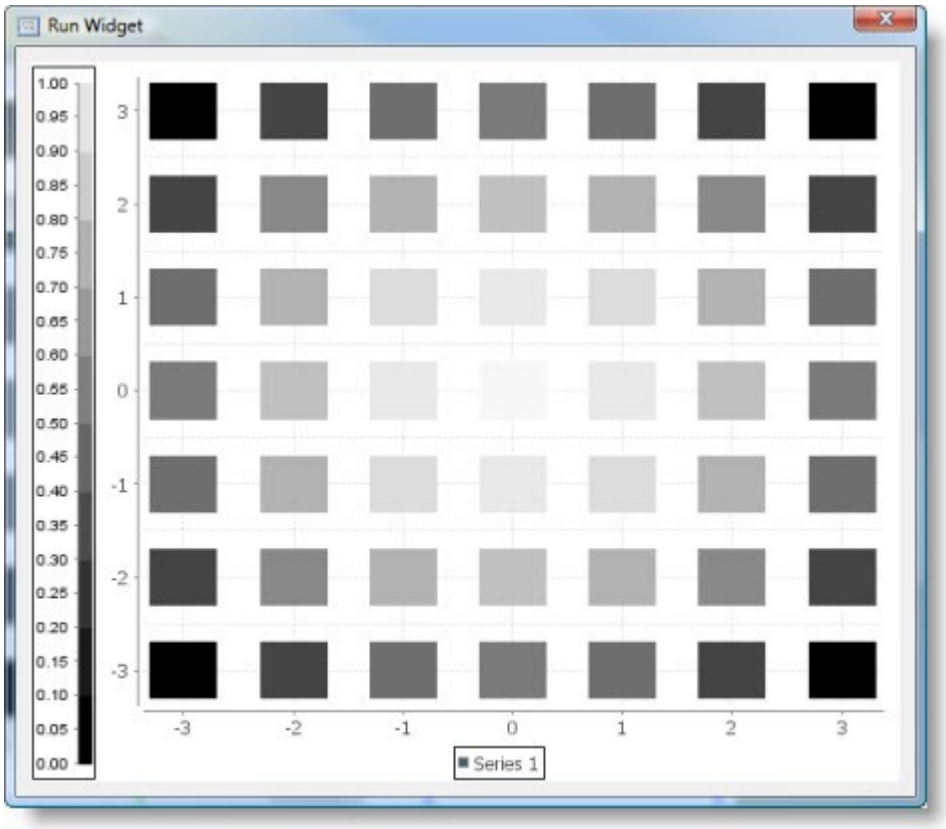
Все соответствующие события графика[1182].

13.4.11.4.10 Блочная диаграмма

Блочная диаграмма представляется собой цветные или различных оттенков серого цвета блоки в двухмерном пространстве, которые обозначают z-значения массива данных XYZ. Z-значения конвертируются в цвета.

Блочная диаграмма основана на координатной области построения 1197 и координатном отрисовщике 1234. Она наследует все их свойства.

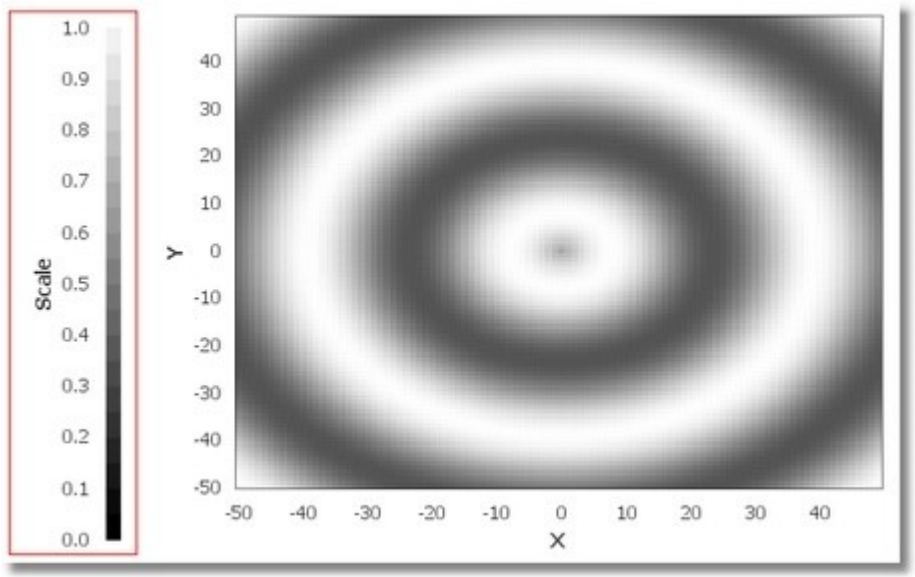
Блочная диаграмма выглядит следующим образом:



Блочная диаграмма поддерживает два отрисовщика:

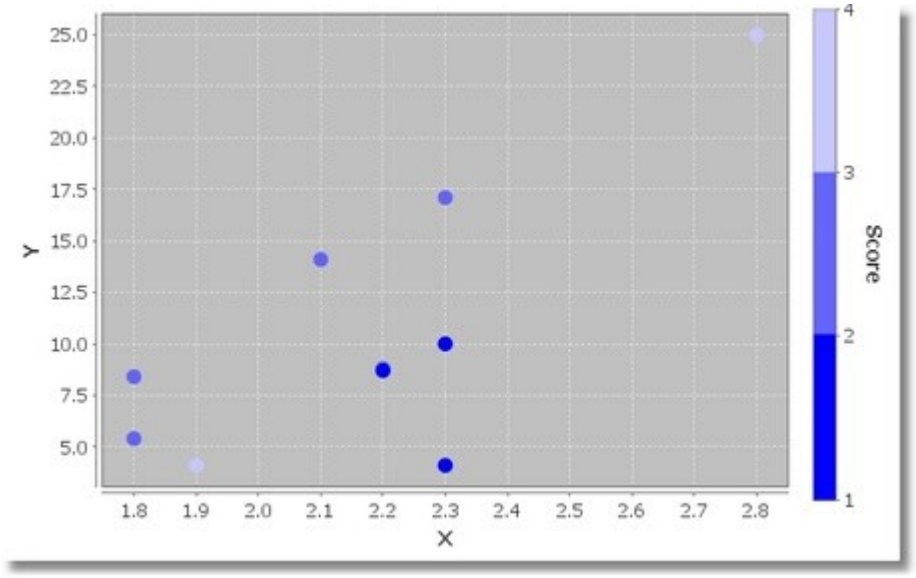
БЛОК

Данный отрисовщик использует цветные прямоугольные блоки для представления элементов данных.



ФОРМА

Данный отрисовщик использует цветные формы 1283 для представления элементов данных.



Массив данных

Блочная диаграмма поддерживает только пользовательские данные 1057.

Имеет следующие привязки исходных данных:

Привязка	Тип ожидаемого значения	Описание
Серия данных	строка	Текстовое имя серии данных. Серия данных представлена на графике в виде одной линии (или набора фигур).
X	число	Положение блока вдоль оси определений (X).
Y	число	Положение блока вдоль оси измерений (Y).
Z	число	Z-значение, используемое для расчета цвета блока/фигуры, отображаемого в точке (X, Y).

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Непрозрачный 1275, Фон 1275, Рамка 1275

Все общие свойства графиков 1160.

Все свойства, относящиеся к данным 1053.

Все свойства координатной области построения 1197.

Все свойства координатного отрисовщика 1234.

Пользовательские свойства

ЦВЕТОВАЯ ШКАЛА

Цветовая шкала применяется для конвертации значений определенного диапазона в цвета (Цвета заливки 1279).

Существуют два типа шкал:

Шкала **Уровни серого** определяет оттенки серого для представления значений в диапазоне. Для значений нижнего предела диапазона шкала определяет темно-серый или черный цвет, для значений верхнего предела - светло-серый или белый цвет. Здесь также возможно настроить прозрачность цветов.

Свойства цветовой шкалы:

Свойство	Имя	Тип	Описание
Тип	type	целое	Тип шкалы: **Полутоновая** или **Поисковая**.

Нижнее ограничение	lowerBound	плавающее	Нижняя граница важных значений для шкалы.
Верхнее ограничение	upperBound	плавающее	Верхняя граница важных значений для шкалы.
Прозрачность	alpha	плавающее	Значение прозрачности для цветов **Полутоновой** шкалы, от 0 (полностью прозрачный) до 255 (полностью непрозрачный).
Цвет по умолчанию	defaultPaint	таблица данных	Цвет заливки[1279] по умолчанию для **Полутоновой** шкалы.
Таблица значений	lookupTable	таблица данных	Таблица, сопоставляющая значения с цветами, используемыми **Полутоновой** шкалой. Состоит из двух полей: • **Значение**: z-значение; • Цвет: цвет заливки[1279] блоков.

Имя свойства: **paintScale**

Тип свойства: **Таблица данных**

ЛЕГЕНДЫ ШКАЛЫ ОКРАСКИ

Таблица, содержащая специальный легенды графика (обычно только одну), отображающие Цветовую шкалу.

Свойства легенды цветовой шкалы:

Свойство	Имя	Тип	Описание
Видимый	visible	логическое	Флажок, указывающий, будет ли отображаться легенда на графике.
Ширина	width	плавающее	Предпочитаемая ширина легенды.
Высота	height	плавающее	Предпочитаемая высота легенды.
Границы	margins	таблица данных	Граница вокруг фрейма легенды. См. Прямоугольные вставки[1180].
Отступы	padding	таблица данных	Пустое пространство внутри фрейма легенды. См. Прямоугольные вставки[1180].
Фрейм	frame	таблица данных	Граница, отображаемая вокруг легенды. См. Фрейм блока[1171].
Позиция	position	строка	Положение легенды на графике (Сверху, Снизу, Слева, Справа).
Горизонтальное выравнивание	horizontalAlignment	строка	Горизонтальное выравнивание легенды. Обычно используется, если Позиция установлена на Сверху или Снизу.
Вертикальное выравнивание	verticalAlignment	строка	Вертикальное выравнивание легенды. Обычно используется, если Позиция установлена на Слева или Справа.
Ось	axis	таблица данных	Ось значений[1165], отображающая числовой диапазон цветовой шкалы.
Положение оси	axisLocation	строка	Положение оси относительно легенды.

Смещение оси	axisOffset	плавающее	Смещение между цветовой полоской и осью. См. Прямоугольные вставки[1180].
Ширина полоски	stripWidth	плавающее	Ширина цветовой полоски.
Видимость окантовки полоски	stripOutlineVisible	логическое	Флажок, контролирующий, будет ли отображаться контур цветовой полоски.
Окраска окантовки полоски	stripOutlinePaint	таблица данных	Цвет[1279] контура цветовой полоски (если контур видимый).
Штрих контура полоски	stripOutlineStroke	таблица данных	Штрих[1282] контура цветовой полоски (если контур видимый).
Окраска фона	backgroundPaint	таблица данных	Цвет заливки[1279] фона легенды.
Количество градаций	subdivisionCount	целое	Количество делений, отображаемых на цветовой шкале.

Имя свойства: **paintScaleLegends**

Тип свойства: **Таблица данных**

ШИРИНА БЛОКА

Ширина блока в единицах данных (измеряемая против оси определений). Значение по умолчанию - 1.0.

Данное свойство пригодно для отрисовщика блоков.

Имя свойства: **blockWidth**

Тип свойства: **Плавающее**

ВЫСОТА БЛОКА

Высота блоко в единицах данных (измеряемая против оси определений). Значение по умолчанию - 1.0.

Данное свойство пригодно для отрисовщика блоков.

Имя свойства: **blockHeight**

Тип свойства: **Плавающее**

ТОЧКА ПРИВЯЗКИ БЛОКА

Точка привязки на блоке, которая будет выровнена относительно положения (x, y) на области построения графика.

Доступные значения - По центру, Сверху, Снизу, Слева, Справа, Сверху слева, Сверху справа, Снизу слева, Снизу справа.

Данное свойство пригодно для отрисовщика блоков.

Имя свойства: **blockAnchor**

Тип свойства: **Строка**

ОТРИСОВКА ОКАНТОВОК

Флажок, контролирующий, будет ли отрисовщик отображать контур форм.

Данное свойство пригодно для отрисовщика форм.

Имя свойства: **drawOutlines**

Тип свойства: **Логическое**

ИСПОЛЬЗОВАТЬ ОКАНТОВКУ

Флажок, контролирующий, будет ли использоваться атрибут Цвет заливки контура для отображения контура фигур. Если флажок установлен на false, для контура применяется обычный цвет серии данных.

Данное свойство пригодно для отрисовщика форм.

Имя свойства: **useOutlinePaint**

Тип свойства: **Логическое**

ИСПОЛЬЗОВАТЬ ЗАЛИВКУ

Флажок, контролирующий, будет ли отрисовщик использовать обычный цвет или цвет заливки для окрашивания форм.

Данное свойство пригодно для отрисовщика форм.

Имя свойства: **useFillPaint**

Тип свойства: **Логическое**

ВИДИМОСТЬ НАПРАВЛЯЮЩИХ

Флажок, контролирующий, будут ли отображаться направляющие линии.

Данное свойство пригодно для отрисовщика форм.

Имя свойства: **guideLinesVisible**

Тип свойства: **Логическое**

ОКРАСКА НАПРАВЛЯЮЩИХ

Цвет[1279] направляющих линий.

Данное свойство пригодно для отрисовщика форм.

Имя свойства: **guideLinePaint**

Тип свойства: **Таблица данных**

ШТРИХ НАПРАВЛЯЮЩИХ

Штрих[1282] направляющих линий.

Данное свойство пригодно для отрисовщика форм.

Имя свойства: **guideLineStroke**

Тип свойства: **Таблица данных**

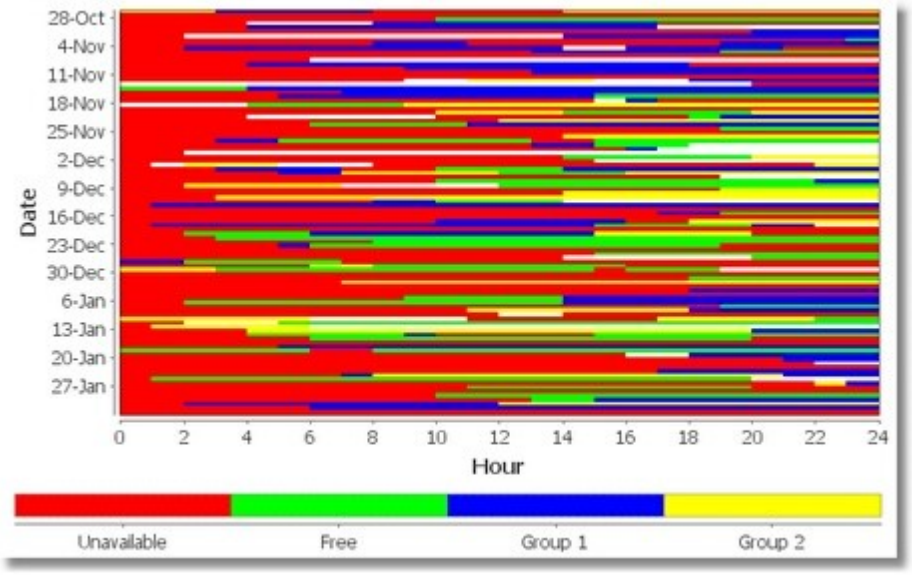
Общие события

Скрытие[1285], Показ[1285], Перемещение[1285], Изменение размеров[1285], Клик мыши[1286], Нажатие кнопки мыши[1286], Отпускание кнопки мыши[1286], Вход мыши[1286], Выход мыши[1286], Перемещение мыши[1286], Вращение колесика мыши[1286], Печать клавиши[1287], Нажатие клавиши[1287], Отпускание клавиши[1287], Получение фокуса[1287], Потеря фокуса[1287]

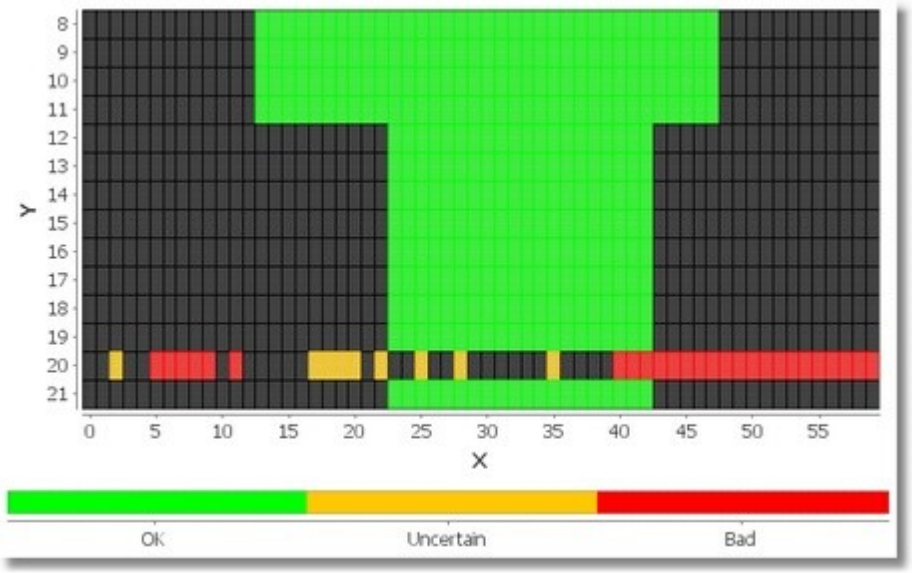
Все соответствующие события графика[1182].

Дополнительный примеры

Блочная диаграмма, использующая ось определений дат:



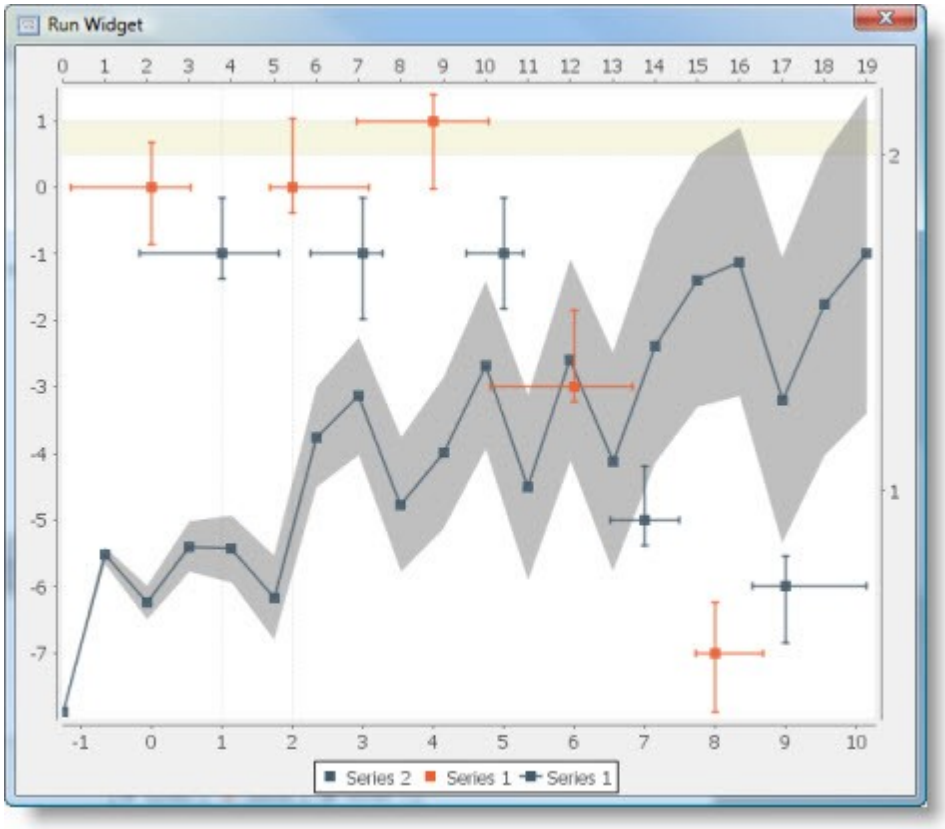
Другой пример блочной диаграммы:



13.4.11.4.11 Смешанная XY-диаграмма

Смешанная XY-диаграмма представляет собой составную диаграмму 1160, предназначенную для совмещения двух и более "простых" координатных XY-графиков 1099 в одной области построения. Эти подграфики имеют общую область построения и оси, но каждый из них имеет собственный массив данных и отрисовщик.

Смешанная XY-диаграмма выглядит следующим образом:



Когда "простой" график добавляется в качестве подграфика к смешанной диаграмме, он имеет некоторые отличия от своей обычной версии:

- Оси подграфика добавляются в список осей графика-контейнера. Подграфик не имеет своих осей. Вместо этого он имеет ссылки на оси графика-родителя, которые он использует для отображения данных. По умолчанию, ссылка осуществляется на переданные оси подграфика в график-родитель.
- Подграфик не имеет свойств, относящихся к области построения 1185, т.к. данные будут отображены в области построения графика-родителя.
- Подграфик не имеет общих свойств графика 1160, смешанный график-родитель обладает своим набором общих свойств.
- Подграфик не имеет свойств компонента виджета по умолчанию 1274, кроме свойства **Привязки**, т.к. он не отображается отдельно.

Пользовательские свойства подграфиков

Каждый "простой" график, добавляемый как подграфик к смешанному координатному графику, поддерживат все свойства трендов 1068 и имеет несколько дополнительных свойств:

ИНДЕКС ОСИ ПАРАМЕТРОВ

Используемый индекс оси параметров смешанной диаграммы. Первая ось в таблице осей параметров графика-родителя имеет индекс **1**.

Имя свойства: **domainAxisIndex**

Тип свойства: **Целое**

ИНДЕКС ОСИ ЗНАЧЕНИЙ

Используемый индекс оси значений смешанной диаграммы. Первая ось в таблице осей значений графика-родителя имеет индекс **1**.

Имя свойства: **rangeAxisIndex**

Тип свойства: **Целое**

ВЫСОТА (Z-ORDER)

Z-индекс подграфика в области построения графика-родителя. Графики с более низкими порядковыми номерами перекрывают графики с большими порядковыми номерами.

Имя свойства: **index**

Тип свойства: **Целое**

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Непрозрачный 1275, Фон 1275, Рамка 1275

Все общие свойства графиков 1160.

Все свойства координатной области построения 1197.

Пользовательские свойства

Не определены.

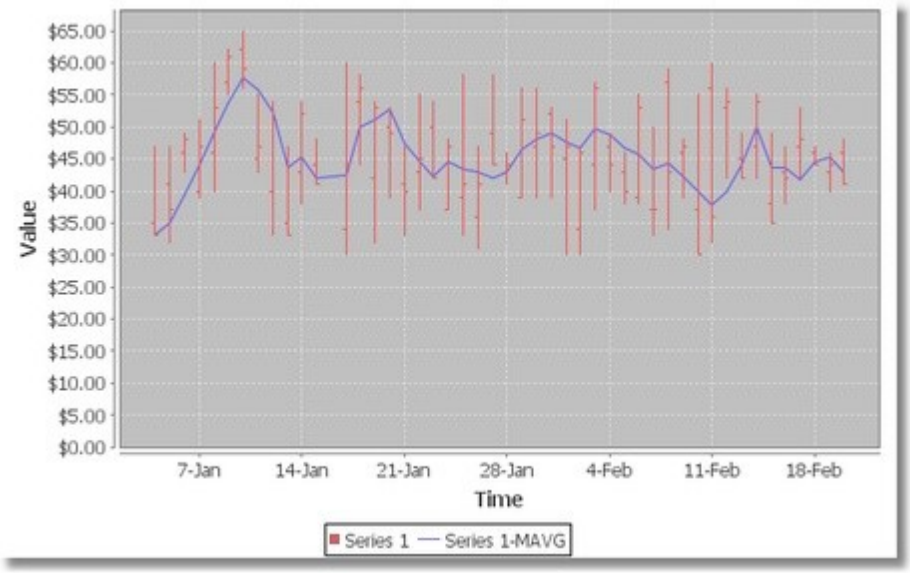
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

Все соответствующие события графика 1182.

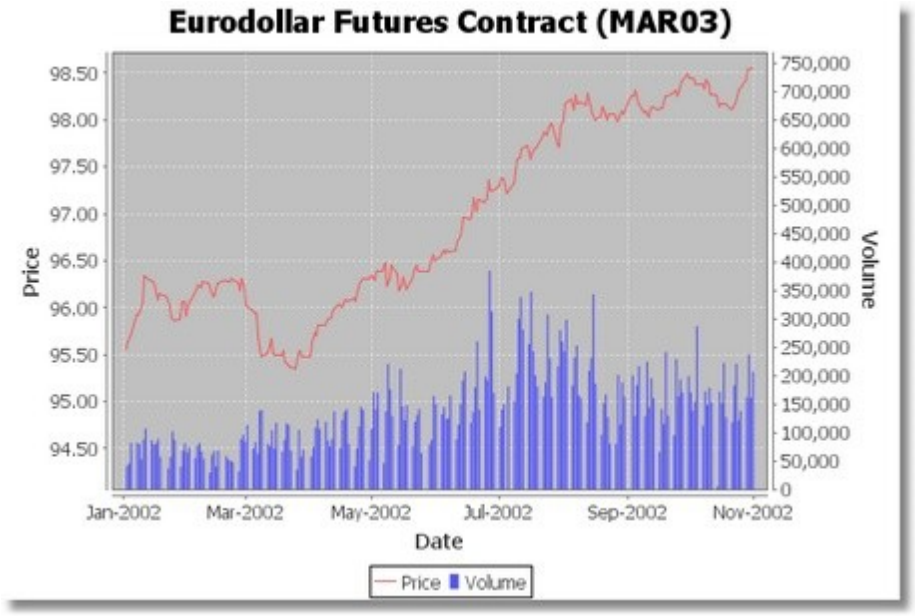
Дополнительные примеры

Смешанная XY-диаграмма с финансовой диаграммой 1130 и трендом 1065 Смещенная средняя:

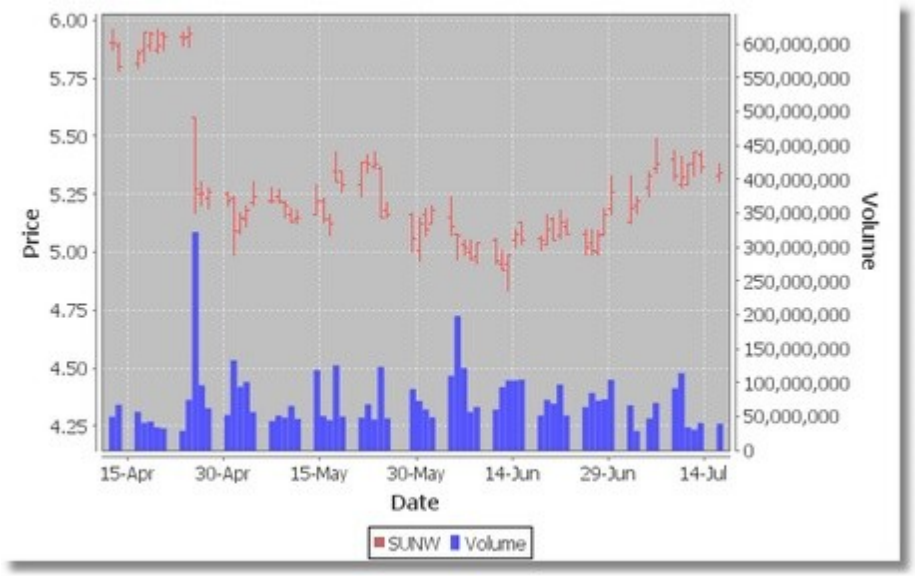


Смешанная XY-диаграмма с XY-диаграммой 1100 и трендом 1065 Смещенная средняя::

Смешанная XY-диаграмма с XY-диаграммой[1100] и столбчатой XY-диаграммой[1101]:

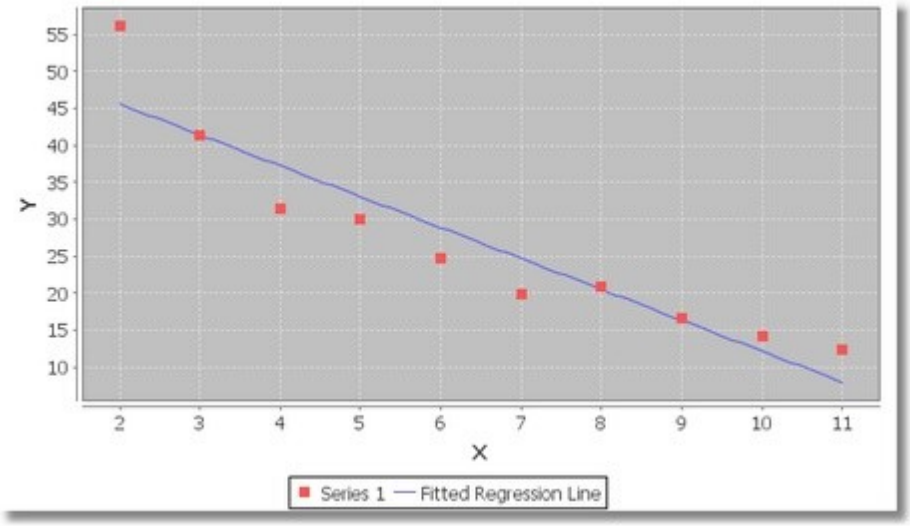


Смешанная XY-диаграмма с финансовой[1130] и столбчатой[1110] диаграммами:



Другая Смешанная XY-диаграмма с XY-диаграммой 1067 и столбчатой XY-диаграммой 1074 с различными осями параметров:

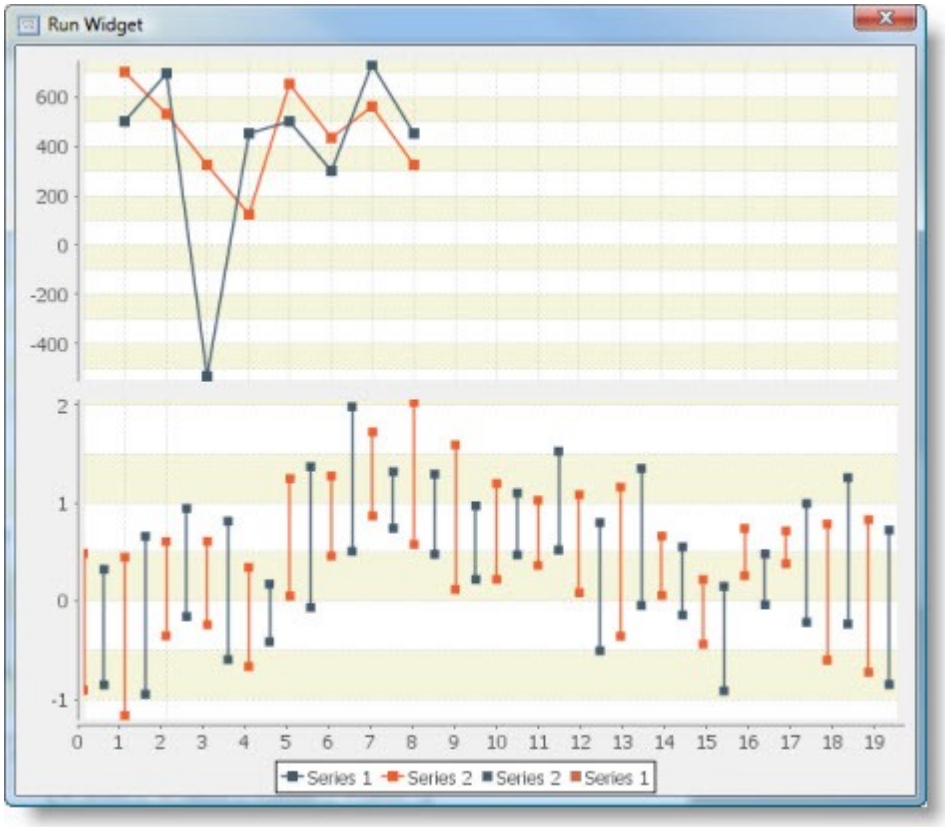
Смешанный график с XY-диаграммой 1100, использующий только отрисовщик фигур (т.е. свойство **Видимые линии по умолчанию** отключено), и линейный тренд 1065 подграфик:



13.4.11.4.12 XY-Диаграмма с общей осью параметров

XY-Диаграмма с общей осью параметров является составной диаграммой 1160, в которой две и более XY-диаграммы 1099 используют одну ось параметров.

XY-Диаграмма с общей осью параметров выглядит следующим образом:



Когда "простая" диаграмма добавляется в качестве подграфика к диаграмме с общей областью параметров, она имеет некоторые отличия от своей обычной версии:

- Нет свойств Оси параметров 1200 и Фиксированное пространство оси параметров 1201, т.к. ось параметров является частью диаграммы-родителя.
- Нет свойства Ориентация 1204, т.к. он унаследует ориентацию смешанного графика-родителя.
- Нет общих свойств диаграммы 1160, т.к. диаграмма-родитель имеет свои общие свойства.
- Нет общих свойств компонента виджета по умолчанию 1274, кроме свойства **Привязки**, т.к. он не отображается отдельно.

Пользовательские свойства подграфиков

Каждая диаграмма, добавляемая как подграфик к XY-Диаграмме с общей осью параметров, имеет несколько дополнительный свойств:

ВЕС

Вес определяет, какое пространство области построения графика-родителя предназначено для данного подграфика. Например, если родитель имеет три подграфика, вес которых равен 1, 2 и 4, пространство, предназначенное для каждого подграфика, будет составлять 1/7, 2/7 и 4/7 соответственно (где 7 - сумма индивидуального веса).

Имя свойства: **weight**

Тип свойства: **Integer**

ВЫСОТА (Z-ORDER)

Подграфики с большим порядковым номером размещаются ближе к общей оси.

Имя свойства: **index**

Тип свойства: **Integer**

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Непрозрачный 1275, Фон 1275, Рамка 1275

Все общие свойства графиков 1160.

Свойство Оси определений 1200 .

Свойство Фиксированное пространство оси параметров 1201.

Пользовательские свойства

ОРИЕНТАЦИЯ

Ориентация диаграммы (Вертикальная или Горизонтальная). Данная ориентация применяется ко всем подграфикам.

Имя свойства: **orientation**

Тип свойства: **Строка**

ПРОМЕЖУТОК

Интервал между подграфиками.

Имя свойства: **gap**

Тип свойства: **Плавающее**

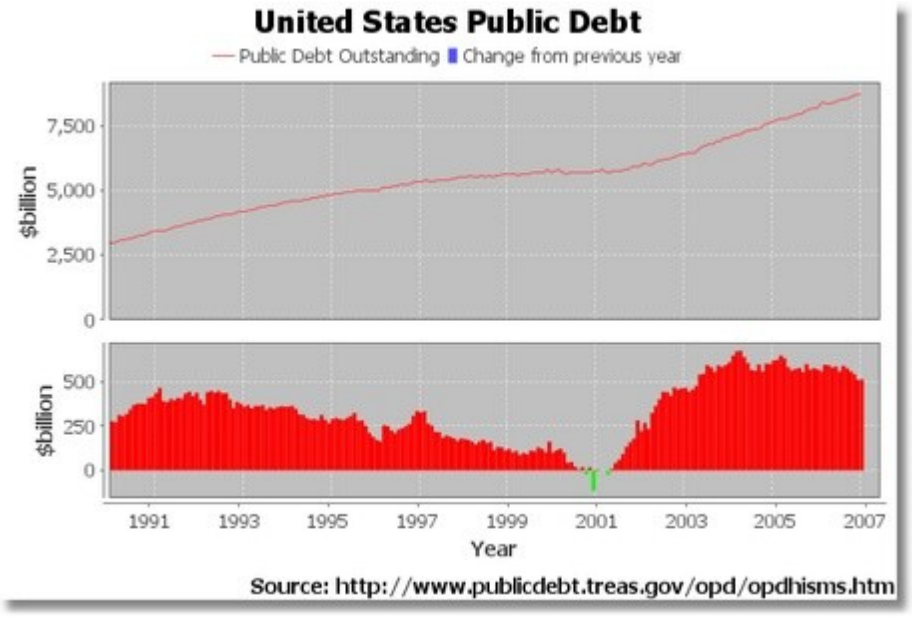
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

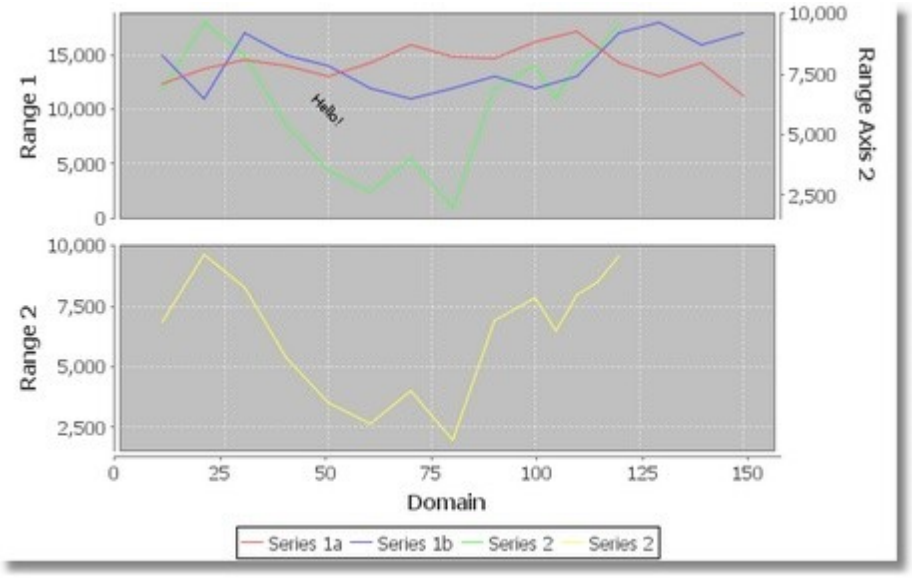
Все соответствующие события графика 1182.

Дополнительные примеры

Два подграфика с общей осью параметров:



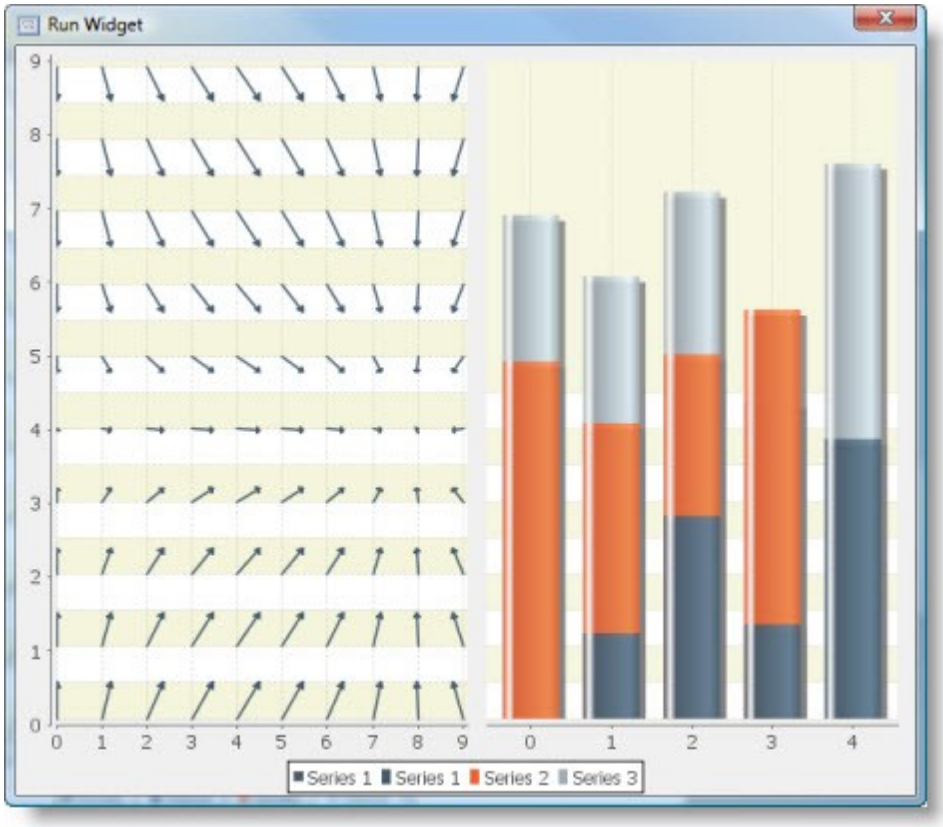
Два подграфика с общей осью параметров. Обратите внимание, что верхняя область построения имеет две оси параметров и Текстовое примечание 1213:



13.4.11.4.13 XY-Диаграмма с общей осью значений

XY-Диаграмма с общей осью значений является составной диаграммой[1160], в которой две и более "простых" диаграммы[1099] используют одну ось определений.

XY-Диаграмма с общей осью значений выглядит следующим образом:



Когда "простая" диаграмма добавляется в качестве подграфика к диаграмме с общей областью определений, она имеет некоторые отличия от своей обычной версии:

- Нет свойств Оси измерений[1200] и Фиксированное пространство оси значений[1201], т.к. Ось значений является частью диаграммы-родителя.
- Нет свойства Ориентация[1204], т.к. она наследует ориентацию диаграммы-родителя.
- Нет общих свойств диаграммы[1160], т.к. диаграмма-родитель имеет свои общие свойства.

- Нет общих свойств компонента виджета по умолчанию 1274, кроме свойства **Привязки**, т.к. он не отображается отдельно.

Пользовательские свойства подграфиков

Каждый "простой" координатный график, добавляемый как подграфик к координатному графику с совмещенной осью измерений, имеет несколько дополнительный свойств:

ВЕС

Вес определяет, какое пространство области построения графика-родителя предназначено для данного подграфика. Например, если родитель имеет три подграфика, вес которых равен 1, 2 и 4, пространство, предназначенное для каждого подграфика, будет составлять 1/7, 2/7 и 4/7 соответственно (где 7 - сумма индивидуального веса).

Имя свойства: **weight**

Тип свойства: **Целое**

ВЫСОТА (Z-ORDER)

Подграфики с большим порядковым номером размещаются ближе к общей оси.

Имя свойства: **index**

Тип свойства: **Целое**

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Непрозрачный 1275, Фон 1275, Рамка 1275

Все общие свойства графиков 1160.

Свойство Оси измерений 1200.

Свойство Фиксированное пространство оси значений 1201.

Пользовательские свойства

ОРИЕНТАЦИЯ

Ориентация диаграммы (Вертикальная или Горизонтальная). Данная ориентация применяется ко всем подграфикам.

Имя свойства: **orientation**

Тип свойства: **Строка**

ИНТЕРВАЛ

Интервал между диаграммами.

Имя свойства: **gap**

Тип свойства: **Плавающее**

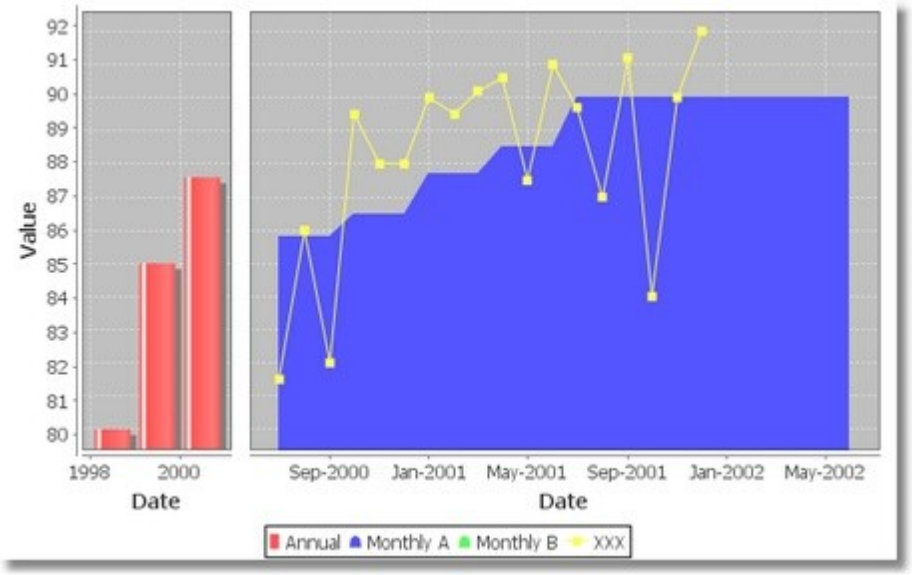
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

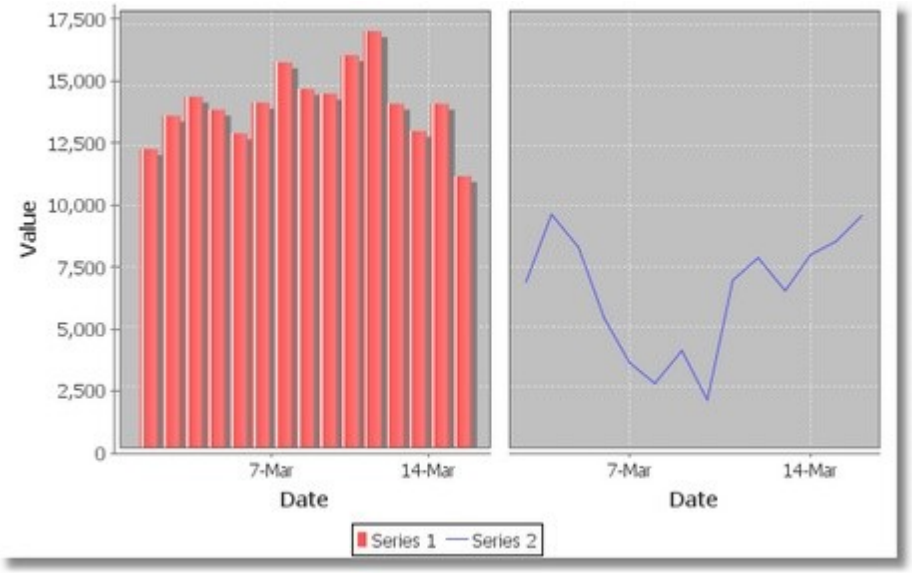
Все соответствующие события графика 1182.

Дополнительные примеры

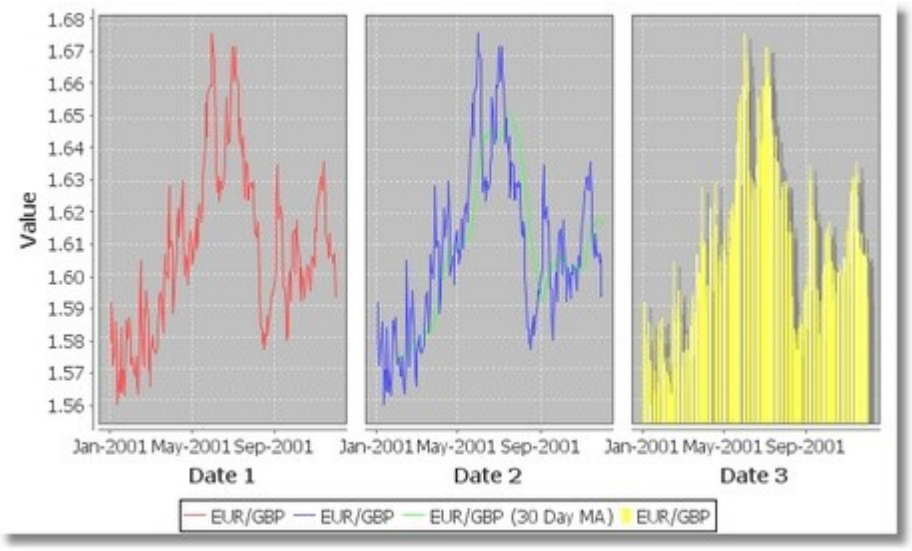
Две диаграммы с общей осью значений:



Две диаграммы с общей осью значений:



Три диаграммы с общей осью дат:



13.4.11.5 Прочие графики

Данный раздел посвящен графикам, использующим нестандартные области построения 1185.

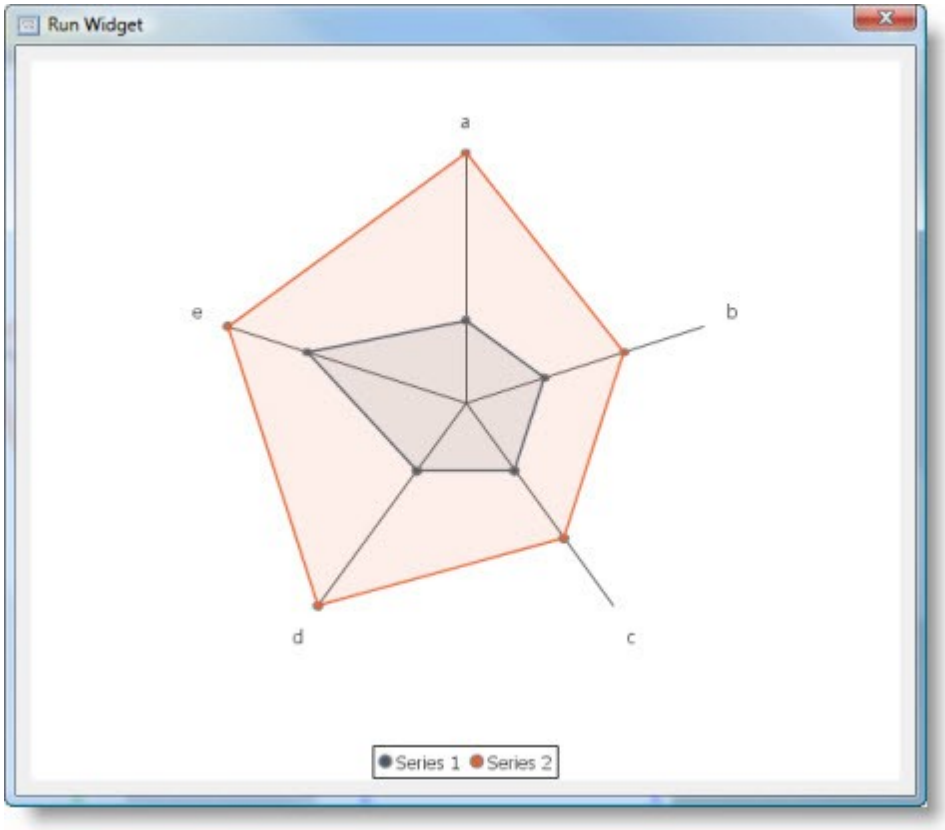
13.4.11.5.1 Лепестковая диаграмма

Лепестковая диаграмма отображает элементы данных (Категорию, Значение) в формате, напоминающей лепестки.

Лепестковая диаграмма основана на стандартной области построения 1185. Он унаследует все ее свойства.

Лепестковая диаграмма не имеет отрисовщика.

Лепестковая диаграмма выглядит так:



Данные

Поддерживает только модель Пользовательские данные 1057.

Она имеет следующие привязки исходных данных:

Привязка	Тип ожидаемого значения	Описание
Серия	строка	Текстовое имя серии данных, отображаемой на графике одноцветной областью (или линиями).
Категория	строка	Имя категории. Каждая категория представлена в виде лучевой линии.
Значение	число	Числовое значение для представления вышеуказанных серий данных/категории. Значения отображаются вдоль лучевых линий.

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Непрозрачный 1275, Фон 1275, Рамка 1275

Все общие свойства графиков 1160.

Свойства Исходные данные 1057 и Привязки исходных данных 1057.

Все свойства области построения 1185.

Пользовательские свойства

ЗАЛИВКА

Флажок, контролирующий, будет ли внутренняя часть многоугольника, образованного точками данных для каждой серии данных, окрашена в цвет.

Имя свойства: **webFilled**

Тип свойства: **Логическое**

ОТНОСИТЕЛЬНЫЙ РАЗМЕР ТОЧЕК ДАННЫХ

Размер фигур, отображаемых в каждой точке данных. Является процентным отношением от ширины и высоты области построения. Значение по умолчанию - 0.01 (один процент)

Имя свойства: **headPercent**

Тип свойства: **Плавающее**

НАЧАЛЬНЫЙ УГОЛ

Угол первой категории, в градусах, относительно лучевой линии, исходящей из центра области построения горизонтально направо (в направлении 3 часов на циферблате) против часовой стрелки. Значение по умолчанию - 90.0, которое отображает первую категорию в верхней части области построения (т.е. 12 часов).

Имя свойства: **startAngle**

Тип свойства: **Плавающее**

МАСИМАЛЬНОЕ ЗНАЧЕНИЕ

Максимальное значение, отображаемое на осях.

Имя свойства: **maxValue**

Тип свойства: **Плавающее**

НАПРАВЛЕНИЕ

Направление, в котором добавляются категории к области построения: **По часовой стрелке** и **Против часовой стрелки**. Направлением по умолчанию является По часовой стрелке.

Имя свойства: **direction**

Тип свойства: **Строка**

ВНУТРЕННИЙ ПРОМЕЖУТОК

Величина в процентах между 0.0 и 0.40 (сорок процентов), указывающая количество пустого пространства вокруг области построения графика (часть которого используется для меток). Значением по умолчанию является 0.25.

Имя свойства: **interiorGap**

Тип свойства: **Плавающее**

ПРОМЕЖУТОК МЕТКИ И ОСИ

Интервал между окончанием каждой "лучевой" оси и соответствующей меткой, выраженное в процентах от длины оси. Значение по умолчанию - 0.10 (десять процентов).

Имя свойства: **axisLabelGap**

Тип свойства: **Плавающее**

ОКРАСКА ЛИНИИ ОСИ

Цвет 1279 лучевых линий осей.

Имя свойства: **axisLinePaint**

Тип свойства: **Таблица данных**

ШТРИХ ЛИНИИ ОСИ

Штрих[1282] лучевых линий осей.

Имя свойства: **axisLineStroke**

Тип свойства: **Таблица данных**

ОКРАСКА СЕРИЙ ПО УМОЛЧАНИЮ

Цвет[1279] серий данных по умолчанию, используемый, если нет пользовательской настройки для серии данных.

Имя свойства: **baseSeriesPaint**

Тип свойства: **Таблица данных**

ОКРАСКА ОКАНТОВКИ СЕРИЙ ПО УМОЛЧАНИЮ

Цвет[1279] контура серий данных по умолчанию, используемый для отображения контура формы в каждой точке данных. Пригодно, если нет пользовательской настройки для серии данных.

Имя свойства: **baseSeriesOutlinePaint**

Тип свойства: **Таблица данных**

ШТРИХ ОКАНТОВКИ СЕРИЙ ПО УМОЛЧАНИЮ

Штрих[1282] контура серий данных по умолчанию, используемый для отображения контура формы в каждой точке данных. Пригодно, если нет пользовательской настройки для серии данных.

Имя свойства: **baseSeriesOutlineStroke**

Тип свойства: **Таблица данных**

ФОРМА ЭЛЕМЕНТОВ ЛЕГЕНДЫ

Форма[1283] элемента легенды.

Имя свойства: **legendItemShape**

Тип свойства: **Таблица данных**

ШРИФТ МЕТОК

Шрифт[1278], используемый для отображения меток категорий.

Имя свойства: **labelFont**

Тип свойства: **Таблица данных**

ОКРАСКА МЕТОК

Цвет[1279], используемый для отображения меток категорий.

Имя свойства: **labelPaint**

Тип свойства: **Таблица данных**

ОКРАСКА СЕРИЙ

Таблица, определяющая цвета для серий данных в отдельности:

Свойство	Имя	Тип	Описание
Индекс	index	целое	Индекс серии данных.
Цвет серии данных	seriesPaint	таблица данных	Цвет[1279] серии данных.

Имя свойства: **seriesPaint**

Тип свойства: **Таблица данных**

ОКРАСКА ОКАНТОВКИ СЕРИЙ

Таблица, определяющая цвета контура для серий данных в отдельности.

Свойство	Имя	Тип	Описание
Индекс	index	целое	Индекс серии данных.
Цвет контура серии данных	seriesOutlinePaint	таблица данных	Цвет 1279 контура фигур серии данных.

Имя свойства: **seriesOutlinePaint**

Тип свойства: **Таблица данных**

ШТРИХ ОКАНТОВКИ СЕРИЙ

Таблица, определяющая штрих контура для серий данных в отдельности.

Свойство	Имя	Тип	Описание
Индекс	index	целое	Индекс серии данных.
Штрих контура серии данных	seriesOutlineStroke	таблица данных	Штрих 1282 контура форм серии данных.

Имя свойства: **seriesOutlineStroke**

Тип свойства: **Таблица данных**

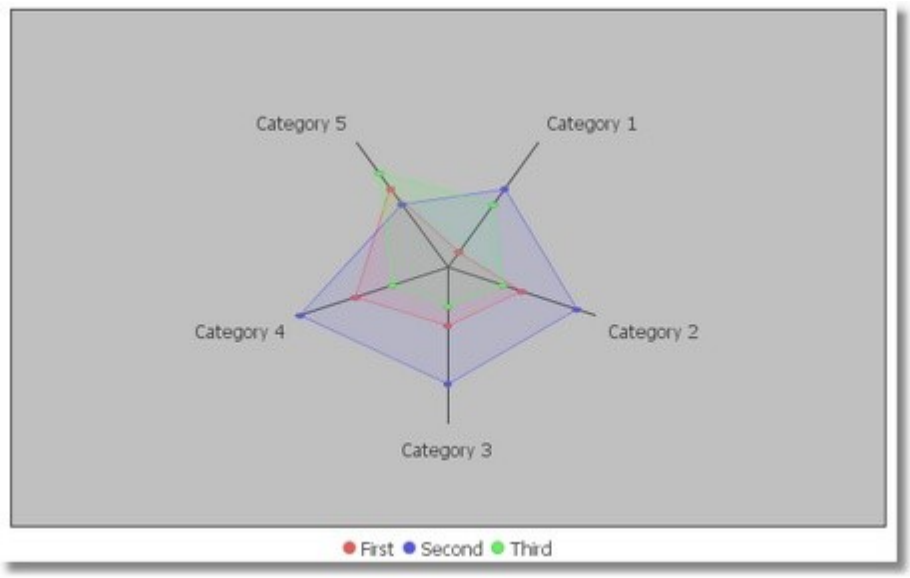
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

Все соответствующие события графика 1182.

Дополнительные примеры

Лепестковая диаграмма с тремя сериями данных:

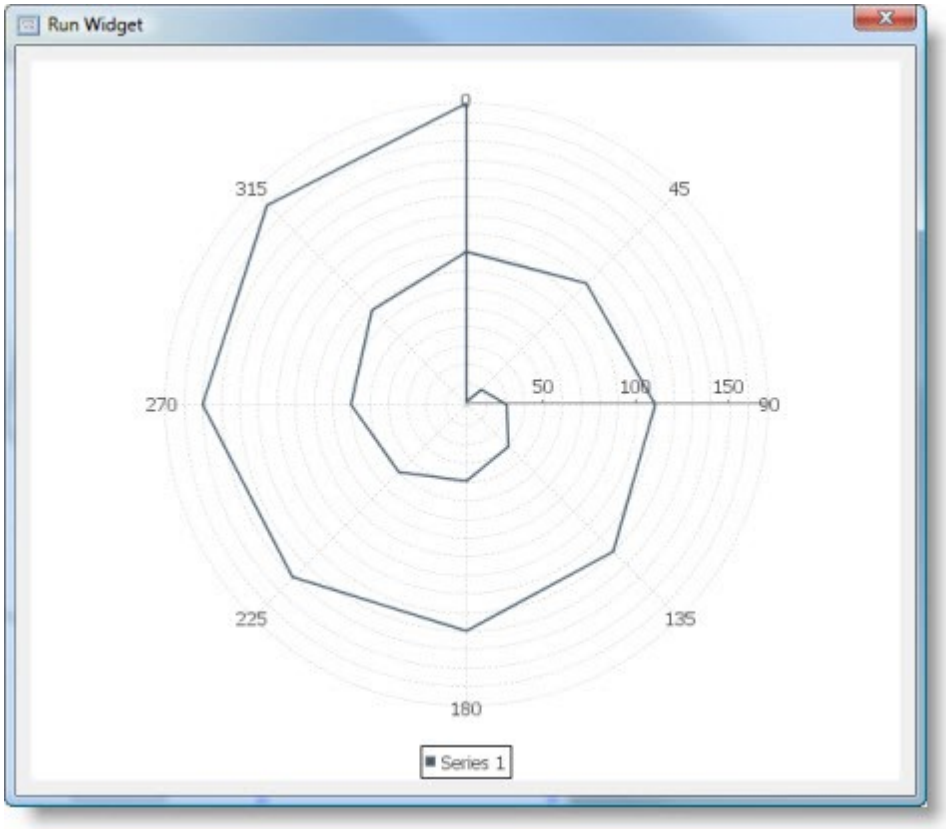


13.4.11.5.2 Полярная диаграмма

Полярная диаграмма отображает элементы данных (X, Y) при помощи полярных координат.

Полярная диаграмма основана на стандартных области построения[1185] и отрисовщике[1221]. Она наследует все их свойства.

Полярная диаграмма выглядит следующим образом:



Массив данных

Полярная диаграмма поддерживает только модель Пользовательские данные[1057].

Она имеет следующие Привязки исходных данных:

Привязка	Тип ожидаемого значения	Описание
Серия данных	строка	Текстовое имя серии данных, представленной на графике одноцветной линией.
Тета	число	Угол элемента данных в градусах.
Радиус	число	Радиус элемента данных (расстояние от центра области построения) для вышеуказанной серии данных/угла.

Общие свойства

Ширина[1274], Высота[1274], Привязки[1275], Видимый[1275], Непрозрачный[1275], Фон[1275], Рамка[1275]

Все общие свойства графиков[1160].

Свойства Исходные данные[1057] и Привязки исходных данных[1057].

Все свойства области построения[1185].

Все свойства отрисовщика[1221].

Пользовательские свойства

ОСЬ

Ось значений1165, представляющая собой шкалу значений для области построения. Ось берет начало в центре и идет в правую сторону графика.

Имя свойства: **axis**

Тип свойства: **Таблица данных**

Общие события

Скрытие1285, Показ1285, Перемещение1285, Изменение размеров1285, Клик мыши1286, Нажатие кнопки мыши1286, Отпускание кнопки мыши1286, Вход мыши1286, Выход мыши1286, Перемещение мыши1286, Вращение колесика мыши1286, Печать клавиши1287, Нажатие клавиши1287, Отпускание клавиши1287, Получение фокуса1287, Потеря фокуса1287

Все соответствующие события графика1182.

Угловые линии сетки

"Угловые линии сетки" являются дополнительными лучевыми линиями, исходящими из центра графика.

ВИДИМОСТЬ УГЛОВЫХ ЛИНИЙ СЕТКИ

Флажок, контролирующий, будут ли отображаться линии сетки углов.

Имя свойства: **angleGridlinesVisible**

Тип свойства: **Логическое**

ЕДИНИЦА ИЗМЕРЕНИЯ УГЛОВЫХ МЕТОК

Угол между двумя лучевыми осями в градусах, контролирующий пространство между угловыми линиями сетки.

Имя свойства: **angleTickUnit**

Тип свойства: **Плавающее**

ШТРИХ УГЛОВЫХ ЛИНИЙ СЕТКИ

Штрих1282, используемый для отображения угловых линий сетки.

Имя свойства: **angleGridlineStroke**

Тип свойства: **Таблица данных**

ОКРАСКА УГЛОВЫХ ЛИНИЙ СЕТКИ

Цвет1279, используемый для отображения угловых линий сетки.

Имя свойства: **angleGridlinePaint**

Тип свойства: **Таблица данных**

ВИДИМОСТЬ УГЛОВЫХ МЕТОК

Флажок, контролирующий, будут ли отображаться метки углов.

Имя свойства: **angleLabelsVisible**

Тип свойства: **Логическое**

ШРИФТ УГЛОВЫХ МЕТОК

Шрифт1278 меток угла.

Имя свойства: **angleLabelFont**

Тип свойства: **Таблица данных**

ОКРАСКА УГЛОВЫХ МЕТОК

Цвет1279 меток угла.

Имя свойства: **angleLabelPaint**

Тип свойства: **Таблица данных**

Радиальные линии сетки

Радиальные линии отображаются в виде кругов на одинаковом расстоянии в области построения.

ВИДИМОСТЬ РАДИАЛЬНЫХ ЛИНИЙ СЕТКИ

Флажок, контролирующий, будут ли отображаться радиальные линии.

Имя свойства: **radiusGridlinesVisible**

Тип свойства: **Таблица данных**

ШТРИХ РАДИАЛЬНЫХ ЛИНИЙ СЕТКИ

Штрих 1282 радиусных линий сетки.

Имя свойства: **radiusGridlineStroke**

Тип свойства: **Таблица данных**

ОКРАСКА РАДИАЛЬНЫХ ЛИНИЙ СЕТКИ

Цвет 1279 радиусных линий сетки.

Имя свойства: **radiusGridlinePaint**

Тип свойства: **Таблица данных**

Угловые текстовые элементы

Область построения предоставляет возможность добавить один и более элементов с кратким текстом (так называемые "угловые текстовые элементы") в правый нижний угол области построения.

УГЛОВЫЕ ТЕКСТОВЫЕ ЭЛЕМЕНТЫ

Список текстовых строк для отображения в нижнем правом углу.

Имя свойства: **cornerTextItems**

Тип свойства: **Таблица данных**

ЗАПОЛНЕНИЕ СЕРИИ ДАННЫХ

Таблица, определяющая, будет ли отображаться каждая серия данных в виде "заполненного" многоугольника:

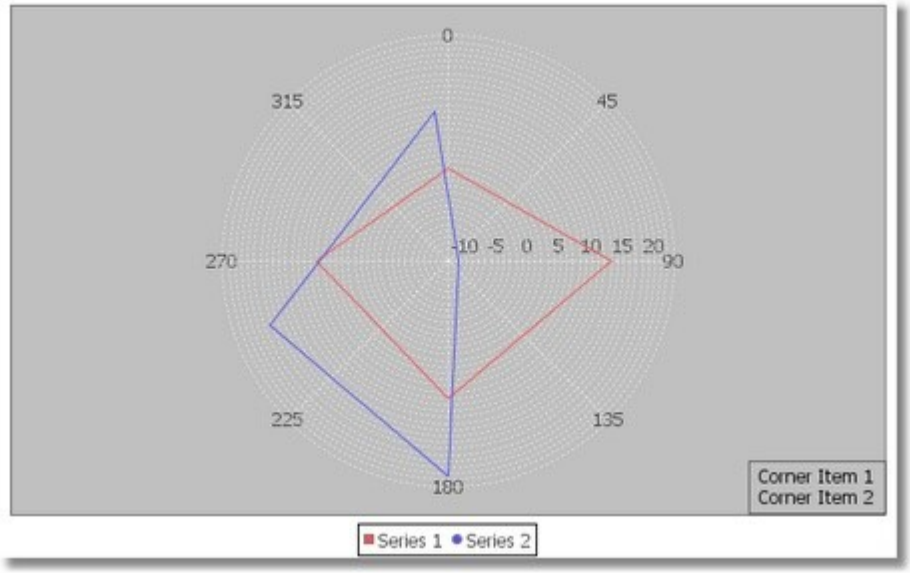
Свойство	Имя	Тип	Описание
Индекс	index	целое	Индекс серии данных.
Серия данных заполнена	seriesFilled	логическое	Флажок, указывающий, должна ли быть заполнена область "внутри" серии данных.

Имя свойства: **seriesFilled**

Тип свойства: **Таблица данных**

Дополнительные примеры

Полярная диаграмма с двумя сериями данных:



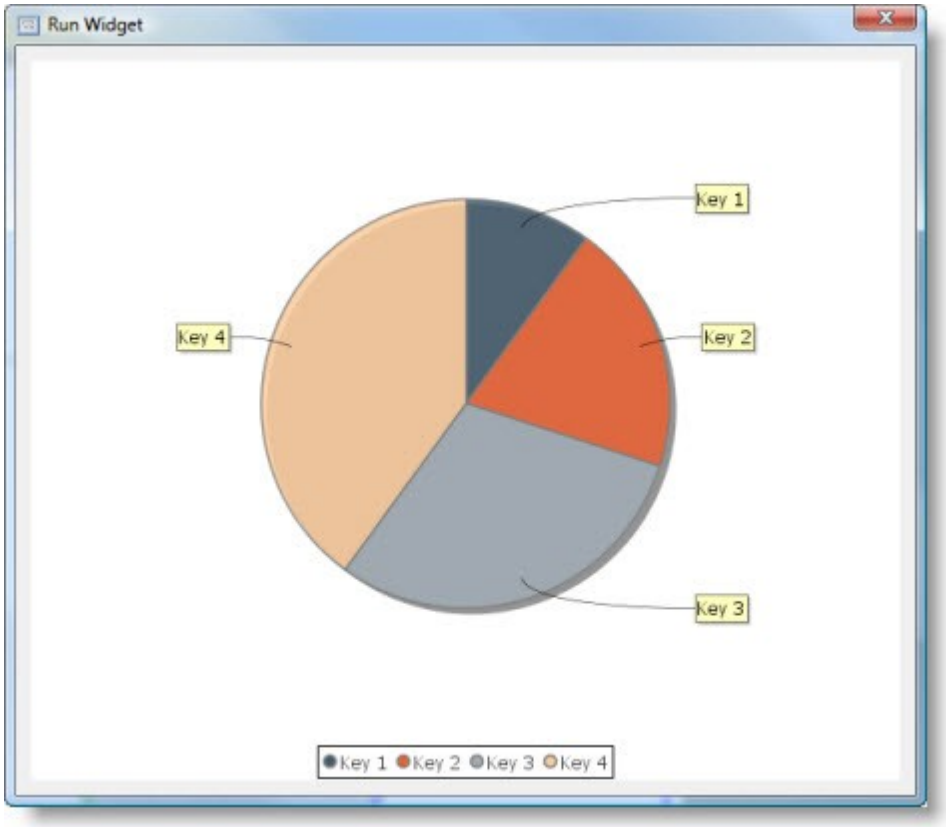
13.4.11.5.3 Круговая диаграмма

Круговая диаграмма является круговым графиком, разбитым на секторы, отображающие пропорции. Длина дуги каждого сектора (и следовательно его центральный угол и область) пропорциональна значению, которое она представляет.

Круговая диаграмма основана на секторной области построения. Она наследует ее свойства.

Круговая диаграмма не имеет отрисовщика.

Круговая диаграмма выглядит следующим образом:



Массив данных

Круговая диаграмма поддерживает только модель Пользовательские данные 1057.

Она имеет следующие привязки исходных данных:

Привязка	Тип ожидаемого значения	Описание
Серия данных	строка	Текстовое имя серии данных. Серия данных представлена отдельным сектором.
Значение	число	Значение для серии данных, т.е. размер сектора графика. Нет необходимости нормализации значения и приравнивать их сумму к 1 или 100%, это делается графиком автоматически.

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Непрозрачный 1275, Фон 1275, Рамка 1275

Все Общие свойства графиков 1160.

Свойства Исходные данные 1057 и Привязки исходных данных 1057.

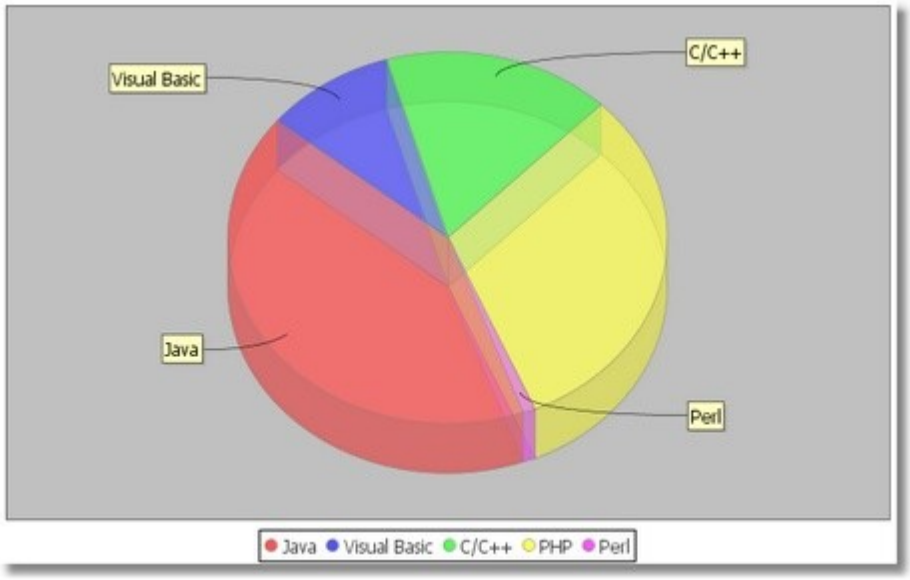
Все свойства секторной области построения 1215.

Пользовательские свойства

3D

Флажок, контролирующий отображение графика в режиме 3D.

Пример 3D секторного графика:



3D Круговая диаграмма не поддерживает видимое отделение секторов.

Имя свойства: **chart3D**

Тип свойства: **Логическое**

КОЭФФИЦИЕНТ ГЛУБИНЫ

Коэффициент глубины определяет размер 3D эффекта в виде процентного отношения к высоте области построения. Значение по умолчанию - 0.12 (двенадцать процентов).

Данное свойство только для режима 3D.

Имя свойства: **depthFactor**

Тип свойства: **Плавающее**

ЗАТЕМНИТЬ БОКОВИНЫ

Флажок, контролирующий, будут ли боковые стороны секторной области построения окрашены в более темный цвет.

Данное свойство пригодно только для режима 3D.

Имя свойства: **darkerSides**

Тип свойства: **Логическое**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

Все соответствующие события графика 1182.

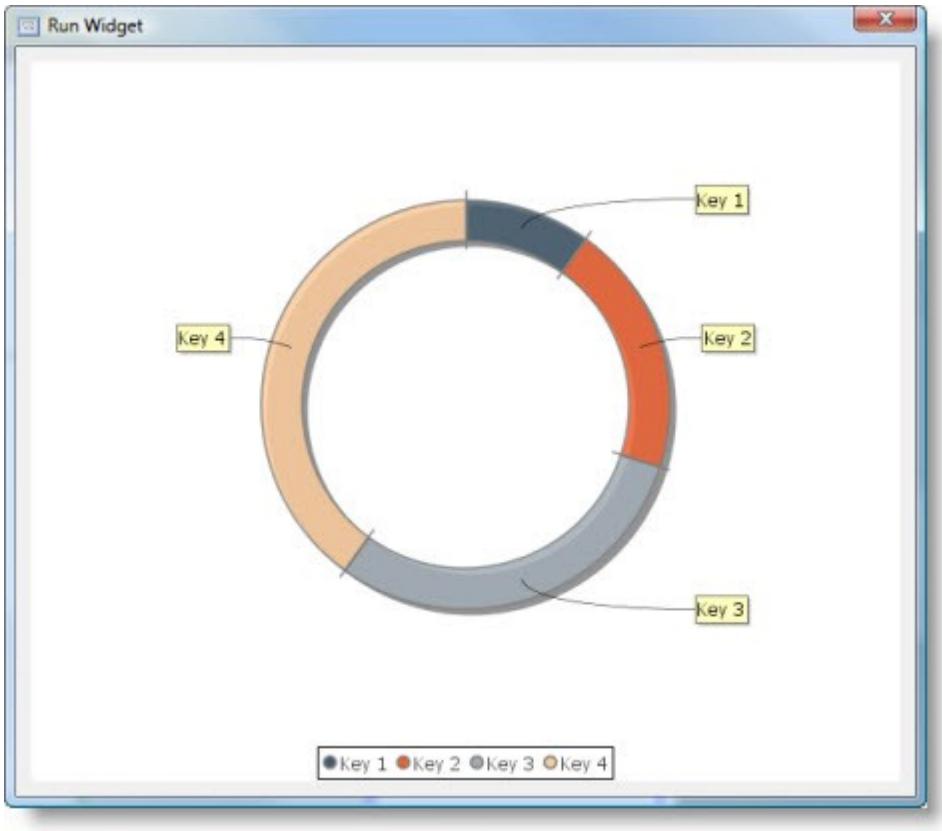
13.4.11.5.4 Кольцевая диаграмма

Кольцевая диаграмма является неким подобием круговой диаграммы 1156 с пустым пространством в середине.

Кольцевая диаграмма основана на секторной области построения 1215. Она наследует ее свойства.

Кольцевая диаграмма не имеет отрисовщика.

Кольцевая диаграмма выглядит следующим образом:



Массив данных

Кольцевая диаграмма поддерживает только модель Пользовательские данные 1057.

Она имеет следующие привязки исходных данных:

Привязка	Тип ожидаемого значения	Описание
Серия данных	строка	Текстовое имя серии данных, отображаемое в виде кольцевого сектора.
Значение	число	Значение серии данных, т.е. размер кольцевого сектора. Нет необходимости нормализовывать значения и приравнивать их сумму к 1 или 100%, это делается графиком автоматически.

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Видимый 1275, Непрозрачный 1275, Фон 1275, Рамка 1275

Все общие свойства графиков 1160.

Свойства Исходные данные 1057 и Привязки исходных данных 1057.

Все свойства секторной области построения 1215.

Пользовательские свойства

ВИДИМОСТЬ РАЗДЕЛИТЕЛЕЙ

Флажок, контролирующий отображение разделителей между секторами.

Имя свойства: **separatorsVisible**

Тип свойства: **Логическое**

ШТРИХ РАЗДЕЛИТЕЛЕЙ

Штрих 1282, используемый для отображения линий разделителей.

Имя свойства: **separatorStroke**

Тип свойства: **Таблица данных**

ЦВЕТ РАЗДЕЛИТЕЛЕЙ

Цвет 1279, используемый для отображения линий разделителей.

Имя свойства: **separatorPaint**

Тип свойства: **Таблица данных**

ВНУТРЕННИЙ ВЫСТУП РАЗДЕЛИТЕЛЕЙ

Длина линии разделителя внутри круга для каждого кольцевого сектора. Значение в процентах от длины кольца (по умолчанию -- 0.20 или 20%).

Имя свойства: **innerSeparatorExtension**

Тип свойства: **Плавающее**

ВНЕШНИЙ ВЫСТУП РАЗДЕЛИТЕЛЕЙ

Длина линии разделителя с внешней стороны круга для каждого кольцевого сектора. Значение в процентах от длины кольца (по умолчанию - 0.20 или 20%).

Имя свойства: **outerSeparatorExtension**

Тип свойства: **Плавающее**

ГЛУБИНА СЕКЦИИ

Глубина секции в процентах от радиуса области построения. Значение по умолчанию -- 0.20 (двадцать процентов).

Имя свойства: **sectionDepth**

Тип свойства: **Плавающее**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

Все соответствующие события графика 1182.

13.4.11.6 Смешанные диаграммы

Смешанные диаграммы предназначены для объединения простых диаграмм. Существует два типа смешанных диаграмм:

- **Смешанные** диаграммы, позволяющие нескольким простым диаграммам использовать одну *область построения* и *оси*, в то время как каждая из них имеет свой *массив данных* и отрисовщик.
- **Совмещенные** диаграммы, которые используют одну общую *ось* между несколькими простыми диаграммами, имеющими отдельные *области построения, массивы данных* и *прорисовщики*.

Смешанные диаграммы

Два типа смешанных диаграмм:

- Смешанный график категорий 1092, объединяющий несколько Графиков категорий 1067
- Смешанный XY график 1139, объединяющий несколько XY графиков 1099

Совмещенные графики

Четыре типа совмещенных графиков:

- График категорий с общей осью параметров 1096, позволяющий нескольким Графикам категорий 1067 использовать одну ось параметров.
- График категорий с общей осью значений 1097, позволяющий нескольким Графикам категорий 1067 использовать одну ось значений.
- XY график с общей осью параметров 1143, позволяющий нескольким XY графикам 1099 использовать одну ось параметров.
- XY график с общей осью значений 1146, позволяющий нескольким XY графикам 1099 использовать одну ось значений.

Добавление подграфиков к составным графикам

Технически составной график действует, как контейнер 1041 для подграфиков.

Чтобы добавить новый подграфик к составному в Редакторе виджетов 423, перетащите график подходящего типа из Панели компонентов 430 в составной график. Таким образом будет добавлен новый график.

Редактирование и удаление подграфиков

Доступ к свойствам подграфиков может быть получен в любое время путем выбора данного графика в Окне источников 428.

Чтобы удалить подграфик в Редакторе виджетов, выберите его в Окне источников 428 и нажмите `Delete`, или выберите **Удалить** в контекстном меню.

13.4.11.7 Общие свойства диаграмм

Данный раздел описывает свойства, поддерживаемые всеми диаграммами.

Рамка

Рамка может быть очерчена вокруг внешних сторон графика, если это необходимо. По умолчанию рамка не отображается, т.к. во многих случаях вместо нее может быть использована рамка по умолчанию 1275 компонента.

ВИДИМОСТЬ ОКАНТОВКИ ГРАФИКА

Флажок, контролирующий отображение "внутренней" рамки, состоящей из одной линии, вокруг графика.

Имя свойства: **borderVisible**

Тип свойства: **Integer**

ОКРАСКА ОКАНТОВКИ ГРАФИКА

Цвет 1279, используемый для отображения "внутренней" рамки.

Имя свойства: **borderPaint**

Тип свойства: **Data Table**

ШТРИХ ОКАНТОВКИ ГРАФИКА

Штрих 1282, используемый для отображения "внутренней" границы.

Имя свойства: **borderStroke**

Тип свойства: **Data Table**

Другие свойства

ОТСТУПЫ

Отступы между границей графика и его областью отображения данных. См. прямоугольные вставки 1180.

Имя свойства: **padding**

Тип свойства: **Data Table**

ЗАГОЛОВОК

Заголовок 1181 графика.

Имя свойства: **title**

Тип свойства: **Data Table**

ТЕКСТОВЫЕ ПОДЗАГОЛОВКИ

Список подзаголовков 1181 графика.

Имя свойства: **textSubtitles**

Тип свойства: **Data Table**

ПОДЗАГОЛОВКИ - ЛЕГЕНДЫ

Список легенд 1173 графика.

Имя свойства: **legendSubtitles**

Тип свойства: **Data Table**

ПОДЗАГОЛОВКИ - ИЗОБРАЖЕНИЯ

Список изображений субтитров 1172 графика.

Имя свойства: **imageSubtitles**

Тип свойства: **Data Table**

СГЛАЖИВАНИЕ КРАЕВ

Применение специальной техники под названием anti-aliasing, которая улучшает внешний вид графика, "сглаживая" края линий и фигур. Использование сглаживания для операций отображения обычно замедляет работу, однако, полученный результат будет лучше.

Имя свойства: **antiAlias**

Тип свойства: **Boolean**

13.4.11.7.1 Оси

Данный раздел посвящен описанию основных свойств оси графика.

Основные особенности оси:

- **Метка оси**: метка оси обычно описывает, что измеряет ось (например, "Продажи в US$").
- **Линия оси**: ось прочерчивает линию вдоль края области построения графика.
- **Метки делений**: маленькие деления на одинаковом расстоянии друг от друга, представляющие шкалу значений, отображаемых осью.
- **Подписи делений**: текстовые описания делений.

Имейте в виду, что классы категорийный график 1187 и координатный график 1197 также отображают контур вокруг области данных. Контур очерчивается перед (под) линией(ями) оси. Штрих контура области построения и его цвет описаны в разделе Граница области построения 1185.

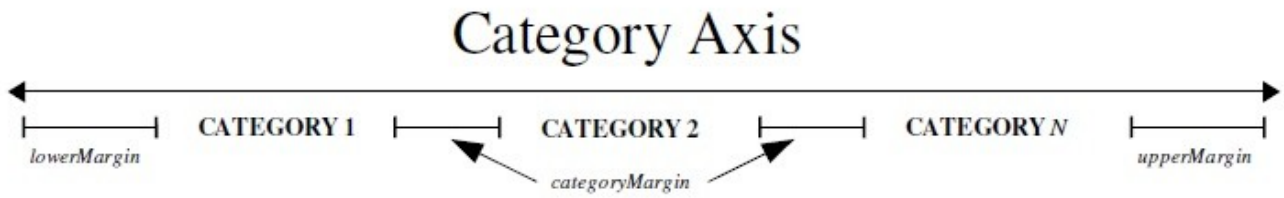
Свойства оси

Свойство	Имя	Тип	Описание
Видимый	visible	логическое	Флажок, контролирующий, будет ли отображаться ось.
Метка	label	строка	Метка оси обычно описывает, что измеряет ось (например, "Продажи в US$"). Если значение является NULL, метка не отображается.
Шрифт меток	labelFont	таблица данных	Шрифт 1278, применяемый для отображения метки оси.
Окраска меток	labelPaint	таблица данных	Цвет 1279, применяемый для отображения метки оси.
Отступы метки	labelInsets	таблица данных	Вставки 1180 метки оси.
Угол метки	labelAngle	плавающее	Угол вращения метки оси (в градусах).
Видимость линии оси	axisLineVisible	логическое	Флажок, контролирующий, будет ли ось отображать линию вдоль края области данных.
Окраска линии оси	axisLinePaint	таблица данных	Цвет 1279, применяемый для отображения линии оси.
Штрих линии оси	axisLineStroke	таблица данных	Штрих 1282, применяемый для отображения линии оси.
Видимость подписей деления	tickLabelsVisible	логическое	Флажок, контролирующий видимость подписей делений.
Шрифт подписей делений	tickLabelFont	таблица данных	Шрифт 1278 подписей делений.
Окраска подписей делений	tickLabelPaint	таблица данных	Цвет 1279 подписей делений.
Отступы подписей	tickLabelInsets	таблица	Пустое пространство между каждой меткой делений. См. Прямоугольные вставки 1180.

делений		данных	
Видимость делений	tickMarksVisible	логическое	Флажок, контролирующий видимость делений.
Окраска делений	tickMarkPaint	таблица данных	Цвет[1279] делений.
Штрих делений	tickMarkStroke	таблица данных	Штрих[1282] делений.
Внутренняя длина делений	tickMarkInsideLength	плавающее	Длина части делений, уходящая внутрь области данных.
Вешняя длина делений	tickMarkOutsideLength	плавающее	Длина внешней части делений, исходящих "наружу" из области данных.
Видимость дополнительных делений	minorTickMarksVisible	логическое	Флажок, указывающий, будут ли отображаться малые деления на оси.
Количество дополнительных делений	minorTickCount	целое	Количество дополнительных делений для каждого шага больших делений. Это переопределяемое поле, если значение > 0, то оно используется, в противном случае ось ссылается на количество дополнительных делений в данном **Сегменте Делений**.
Внутренняя длина дополнительных делений	minorTickMarkInsideLength	плавающее	Длина отметок дополнительных делений внутри области данных (ноль разрешен).
Внешняя длина дополнительных делений	minorTickMarkOutsideLength	плавающее	Длина отметок дополнительных делений, исходящих из области данных (ноль разрешен).

13.4.11.7.1.1 Ось категорий

Ось категорий используется в качестве оси параметров в области построения категорий[1187]. Категории отображаются на одинаковом расстоянии друг от друга, с интервалом перед первой категорией (нижняя граница), интервалом после последней категорией (верхняя граница) и интервалом между каждой категорией (граница категорий).



Свойства

См. Оси[1162], где представлен список типичных свойств осей.

Ось категорий имеет следующие дополнительные свойства:

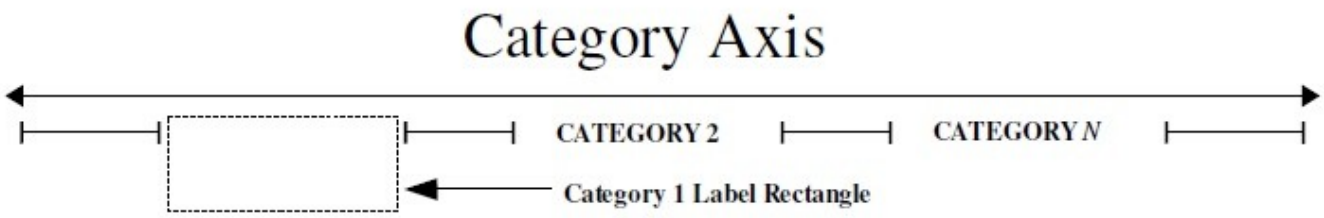
Свойство	Имя	Тип	Описание
Тип	type	целое	Тип оси: **Ось категорий** или **Расширенная ось категорий**.
Нижняя граница	lowerMargin	плавающее	Нижняя граница оси в процентах от общей длины оси. Значение по умолчанию -- 0.05 (пять процентов).

Верхняя граница	upperMargin	плавающее	Верхняя граница оси в процентах от общей длины оси. Значение по умолчанию -- 0.05 (пять процентов).
Граница категорий	categoryMargin	плавающее	Граница категорий оси. Является процентным отношением от длины оси (например, 0.20 для границы в двадцать процентов). Общая граница распределена через интервалы N-1, где N -- количество категорий, отображаемых на оси.
Максимальное количество строк меток категорий	maximumCategoryLabelLines	целое	По умолчанию метки категорий отображаются на одной строке и при необходимости урезаются. Тем не менее, поддерживаются многострочные метки. Данная опция определяет максимальное количество строк для отображения меток категорий.
Максимальное значение коэффициента ширины меток категорий	maximumCategoryLabelWidthRation	плавающее	Отношение, умножаемое на ширину одной категории для определений максимальной ширины метки. Нулевое (0.0) или отрицательное значения деактивируют эту настройку.
Смещение положения подписей категорий	categoryLabelPositionOffset	целое	Смещение между осью и меткой категорий.
Положение подписей категорий	categoryLabelPositions	таблица данных	Табличное свойство, контролирующее позицию, выравнивание и вращение меток категорий вдоль оси. См. Позиции меток категорий 1164.
Свойства подписей категорий	categoryLabelProperties	таблица данных	Табличное свойство, контролирующее шрифт, цвет и всплывающие подсказки отдельных меток категорий. См. Свойства меток категорий 1165.

Положение подписей категорий

Когда необходимо определить, где должны быть расположены подписи категорий, ось выберет один из экземпляров, в зависимости от текущего положения оси (сверху, снизу, слева или справа в области построения). Являются атрибутами Позиции подписей категорий, которые полностью определяют, где должны быть отображены подписи.

- Первый атрибут - точка привязки относительно прямоугольника, рассчитываемая осью. Внутри данного прямоугольника задается *точка привязки.*



- Вторым атрибутом является точка привязки текста, который определяет точку в подписи категорий, которая выравнивается с точкой привязки в прямоугольнике категории, упомянутом выше.
- Два дополнительных атрибута определяют точку вращения и угол вращения, которые применяются после того, как текст подписи был расположен при помощи двух предыдущих атрибутов.
- Относительная ширина и тип относительной ширины контролируют максимальную ширину подписи категорий.

СВОЙСТВА ПОЗИЦИЙ ПОДПИСЕЙ КАТЕГОРИЙ

Свойство	Имя	Тип	Описание
Точка привязки	anchor	строка	Используется для определений точки на оси, по которой выравнивается метка категории. Он задается относительно прямоугольной области, которую Ось категорий предоставляет для категории.

Точка привязки метки	labelAnchor	строка	Определяет точку на метке категории, которая центрируется с точкой привязки категории.
Точка привязки вращения	rotationAnchor	строка	Точка на метке категории, вокруг которой вращается метка (имейте в виду, что вращения может и не быть).
Угол	angle	плавающее	Угол вращения в градусах.
Тип ширины	widthType	строка	Контролирует отношение максимальной ширины меток к ширине прямоугольника метки категории (**категория**) или же к длине оси измерений (**измерений**). Последняя опция удобна для применения, когда метки вращаются так, что они становятся перпендикулярно по отношению к оси категорий.
Коэффициент ширины	widthRatio	плавающее	Максимальная относительная ширина метки категории, измеряемая в процентах от ширины прямоугольника метки категории или же от длины оси категорий (см. предыдущую настройку).

Свойства подписей категорий

Данная настройка определяет, как отображаются метки категорий по оси.

Свойство	Имя	Тип	Описание
Категория	category	строка	Категория, для которой применяются настройки метки.
Шрифт	font	таблица данных	Шрифт[1278] определенной метки категории.
Окраска	paint	таблица данных	Цвет[1279] определенной метки категории.
Всплывающая подсказка	tooltip	строка	Всплывающая подсказка метки категории.

Расширенная ось категорий является подтипом оси категорий[1165], позволяющая отображать Дополнительные метки вместе с категориями.

Дополнительные свойства расширенной оси категорий:

Свойство	Имя	Тип	Описание
Дополнительные метки	subLabels	таблица данных	Таблица субметок, каждая из которых включает в себя **Ключ категории** и Текст **метки**.
Шрифт дополнительных меток	subLabelFont	таблица данных	Шрифт[1278], используемый для отображения дополнительных меток.
Окраска дополнительных меток	subLabelPaint	таблица данных	Цвет[1279], используемый для отображения дополнительных меток.

13.4.11.7.1.2 Ось значений

Ось, отображающая цифры или даты (внутренне представляемые как числовые значения).

Применение:

- В Области построения категорий[1187] осью значений по умолчанию является Ось значений.

- В Координатной области построения[1197] оси параметров и значений являются Осями значений по умолчанию.

Диапазон оси

Диапазон оси определяет самое высокое и самое низкое значения, отображаемые на оси. На графике значения данных вне диапазона оси обычно урезаются и, следовательно, невидимы.

АВТОМАТИЧЕСКИЙ РАСЧЕТ ГРАНИЦ

По умолчанию оси конфигурируются для автоматического вычисления диапазонов так, чтобы все данные в вашем массиве отображались на графике. Это происходит в результате определений самого высокого и самого низкого значений в массиве данных, добавляя небольшую границу (во избежание отображения данных у самого края графика) и устанавливая диапазон оси. Чтобы проверить, настраивается ли диапазон оси автоматически для вмещения доступных данных, используйте параметр Автодиапазон.

Свойства

См. Оси[1162], где указан список типичных свойств осей.

Ось значений имеет следующие дополнительный свойства:

Свойство	Имя	Тип	Описание
Тип	type	целое	Тип оси: **Ось чисел, Ось дат, Ось периодов, Логарифмическая ось** или **Ось символов**.
Инвертировать	inverted	логическое	Флажок, контролирующий, инвертирована ли шкала оси.
Вертикальные метки делений	verticalTickLabels	логическое	Флажок, контролирующий вращение меток делений на 90 градусов. Обычно он настроен так, чтобы вмещать большее количество меток.
Нижняя граница	lowerMargin	плавающее	Нижняя граница оси в процентах от общей длины оси. Значением по умолчанию является 0.05 (пять процентов). Имейте в виду, что граница применяется только тогда, когда место, занимаемое осью, вычисляется автоматически. Если вы вручную определяете место расположения оси, границы игнорируются.
Верхняя граница	upperMargin	плавающее	Верхняя граница оси в виде процентного отношения от общей длины оси. Значением по умолчанию является 0.05 (пять процентов). Имейте в виду, что граница применяется только тогда, когда место, занимаемое осью, вычисляется автоматически. Если вы вручную определяете место расположения оси, границы игнорируются.
Видимость стрелки с положительной стороны	positiveArrowVisible	логическое	Флажок, контролирующий, будет ли отображаться стрелка на конце оси положительных значений.
Видимость стрелки с отрицательной стороны	negativeArrowVisible	логическое	Флажок, контролирующий, будет ли отображаться стрелка на конце оси отрицательных значений.
Стрелка вверх	upArrow	таблица данных	Фигура[1283], используемая для отображения стрелки на конце оси, идущей вверх.

Стрелка вниз	downArrow	таблица данных	Фигура 1283, используемая для отображения стрелки на конце оси, идущей вниз.
Стрелка влево	leftArrow	таблица данных	Фигура 1283, используемая для отображения стрелки на конце оси, идущей влево.
Стрелка вправо	rightArrow	таблица данных	Фигура 1283, используемая для отображения стрелки на конце оси, идущей вправо.
Авто выбор единицы измерения делений	autoTickUnitSelection	логическое	Флажок, контролирующий, будет ли шаг делений выбираться автоматически.
Авто диапазон	autoRange	логическое	Флажок, контролирующий, будет ли диапазон оси настраиваться автоматически для вмещения всех доступных значений.
Фиксированный авто диапазон	fixedAutoRange	плавающее	Если задано (ненулевое), автодиапазон рассчитывается путем вычета данного значения из максимального значения домена в массиве данных.
Минимальный размер автодиапазона	autoRangeMinimumSize	плавающее	Наименьшее значение, допустимое при автоматическом расчете.

МЕТКИ ДЕЛЕНИЙ

Ось поддерживает еще несколько "групп" меток, где каждая метка определяется следующими свойствами:

Свойство	Имя	Тип	Описание
Период	periodClass	строка	Тип Временного периода: Год, Месяц, День, Час, Минута, Секунда, Миллисекунда.
Отступ	padding	таблица данных	Отступ контролирует минимальное расстояние между метками. См. Прямоугольные вставки 1180.
Формат даты	dateFormat	строка	Шаблон форматирования даты/времени 2146 для меток.
Шрифт меток	labelFont	таблица данных	Шрифт 1278, используемый для отображения меток для каждого временного периода.
Цвет меток	labelPaint	таблица данных	Цвет 1279, используемый в качестве цвета заливки фона при отображении меток для каждого временного периода.
Отображать разделители	drawDividers	логическое	Флажок, определяющий, будут ли отображаться разделители между временными периодами.
Штрих разделителей	dividerStroke	таблица данных	Штрих 1282, используемый для отображения разделителей между временными периодами.

Цвет разделителей	dividerPaint	таблица данных	Цвет[1279], используемый для отображения разделителей между временными периодами.

Числовая ось является подтипом оси значений[1165], отображающая числовые данные вдоль линейной шкалы.

Дополнительные свойства числовой оси:

Свойство	Имя	Тип	Описание
Автовыбор единицы измерения делений	numberTickUnits	целое	Тип автоматически выбираемых шагов делений: **Integer** или **Float**.
Единица измерения делений	numberTickUnit	таблица данных	Данная опция позволяет конфигурировать шаги делений, если Автоматический выбор шагов отключен. Включает в себя два поля: • **Размер.** Число с плавающей запятой, определяющее расстояние между двумя делениями. • **Количество малых делений**. Целочисленное количество малых делений между двумя большими.
Формат единиц измерения делений	numberFormatOverride	строка	Шаблон форматирования чисел[2147] для меток делений на оси.
Тип диапазона	rangeType	строка	Данная настройка используется для отображения только положительных или только отрицательных значений на оси. Возможными значениями являются **Все, Положительные** и **Отрицательные.**
Диапазон	numberRange	таблица данных	Заданный вручную диапазон шкалы. Определяется в виде значений типа плавающее **Нижней** и **Верхней границ**. Данная настройка доступна, если Автодиапазон отключен.
Автодиапазон по умолчанию	defaultNumberAutoRange	таблица данных	Диапазон оси, используемый в случае, когда массив данных не имеет данных. Определяется в виде значений типа плавающее **Нижней** и **Верхней границ**.
Включать нули в автодиапазон	autoRangeIncludesZero	логическое	Если Автодиапазон установлен на true (так, чтобы диапазон оси настраивался автоматически для вмещения текущих данных), вы также можете установить данный флажок на включение нуля в диапазон оси. Имейте в виду, что некоторые отрисовщики (например, отрисовщик столбцов) имеют флажок, контролирующий включение некоторого "базового" значения в диапазоне оси -- т.к. базовое значение часто является нулем по умолчанию, вам необходимо установить также флажок данного отрисовщика, чтобы получить требуемый диапазон для оси.
Залипающий ноль в автодиапазоне	autoRangeStickyZero	логическое	Флажок, контролируйщий, будет ли усекаться граница оси, если нулевое значение попадает в ее пределы.

Временная ось **является** подтипом оси значений[1165], которая отображает значения даты/времени. По сути является универсальной для миллисекунд отображения различных диапазонов даты/времени - от миллисекунд до столетий.

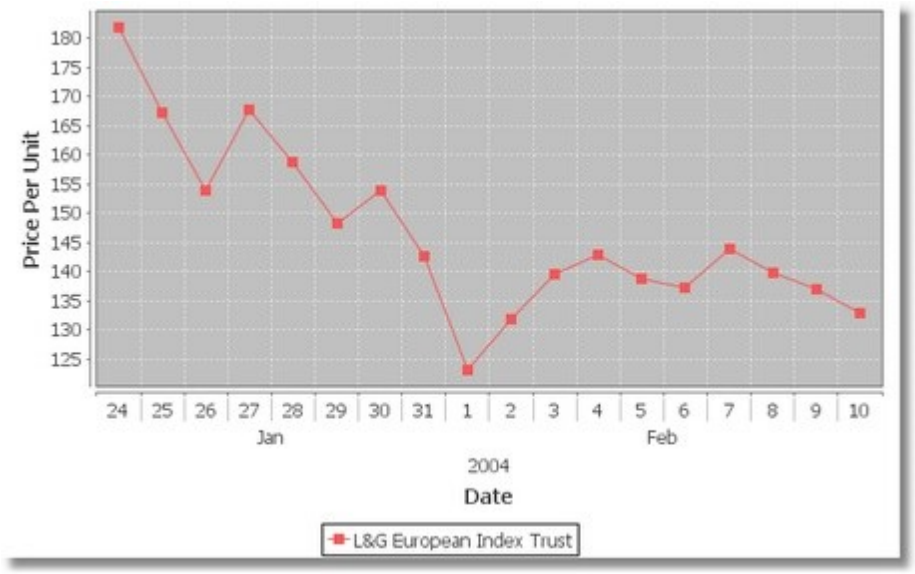
Несмотря на то, что ось отображает даты в качестве меток делений, на нижнем уровне она все равно оперирует с числами с плавающей запятой, полученными из массива данных области построения. Значения интерпретируются в качестве количества миллисекунд, прошедших с 1 января 1970.

Дополнительные свойства оси дат:

Свойство	Имя	Тип	Описание
Единица измерения делений	dateTickUnit	таблица данных	Данная опция позволяет конфигурировать шаги делений, если отключен Автоматический выбор шага делений. Включает в себя два поля: • **Тип шага делений.** Год, Месяц, День, Час, Минута, Секунда или Миллисекунда. • **Множество**. Количество шагов между двумя смежными метками делений.
Формат единиц измерения делений	dateFormatOverride	строка	Шаблон форматирования даты/времени 2146 для меток делений на оси.
Автодиапазон	dateRange	таблица данных	Заданный вручную диапазон шкалы. Определяется в виде значений типа плавающее **Нижней** и **Верхней границ**. Данная настройка доступна, если Автодиапазон отключен.
Автодиапазон по умолчанию	defaultDateAutoRange	таблица данных	Диапазон оси, используемый в случае, когда массив данных не имеет данных. Определяется в виде временных меток **Нижней** и **Верхней границ**.
Положение отметок	tickMarkPosition	строка	Позиция делений: **В начале**, **В середине**, **В конце** временного периода.
Временная зона	timeZone	строка	Временная зона для оси, которая влияет на конвертацию значений дат в строковые метки.

Ось периодов является подтипом оси значений 1165, которая отображает значения даты/времени и имеет следующие характеристики:

- Множество групп меток, каждая из которых делится на временные периоды;
- Автоматический расчет диапазона, основанный на (целых) временных периодах
- Устанавливаемая пользователем временная зона.



В вышеуказанном примере можно увидеть три группы. Первая группа разбита на временные периоды, равные одному дню, вторая - периоды, равные 1 месяцу, третья - периоды, равные 1 году.

Дополнительные свойства оси периодов:

Свойство	Имя	Тип	Описание
Временная зона	timeZone	строка	Временная зона, которая "прикрепляет" периоды времени к линии абсолютного времени.
Локаль	locale	строка	Локаль используется для форматирования временных меток в строки меток делений.
Период автодиапазона	autoRangeTimePeriodClass	строка	Временной период, используемый, когда диапазон оси рассчитывается автоматически: Год, Месяц, День, Час, Минута, Секунда или Миллисекунда. Диапазон оси всегда будет являться целым числом периодов.
Период основных делений	majorTickTimePeriodClass	строка	Временной период, используемый для определения пространства между большими делениями: Год, Месяц, День, Час, Минута, Секунда или Миллисекунда.
Период дополнительных делений	minorTickTimePeriodClass	строка	Временной период, используемый для определения пространства между малыми делениями: Год, Месяц, День, Час, Минута, Секунда или Миллисекунда.
Штрих дополнительных делений	minorTickMarkStroke	таблица данных	Штрих 1282, используемый для отображения малых делений.
Окраска дополнительных делений	minorTickMarkPaint	таблица данных	Цвет 1279, используемый для отображения малых делений.
Метки периодов	labelInfo	таблица данных	Описания группы меток, см. далее.

Логарифмическая ось является подтипом оси значений 1165, которая отображает значения при помощи логарифмической оси. Данная ось может представлять только положительные значения.

Дополнительные свойства логарифмической оси:

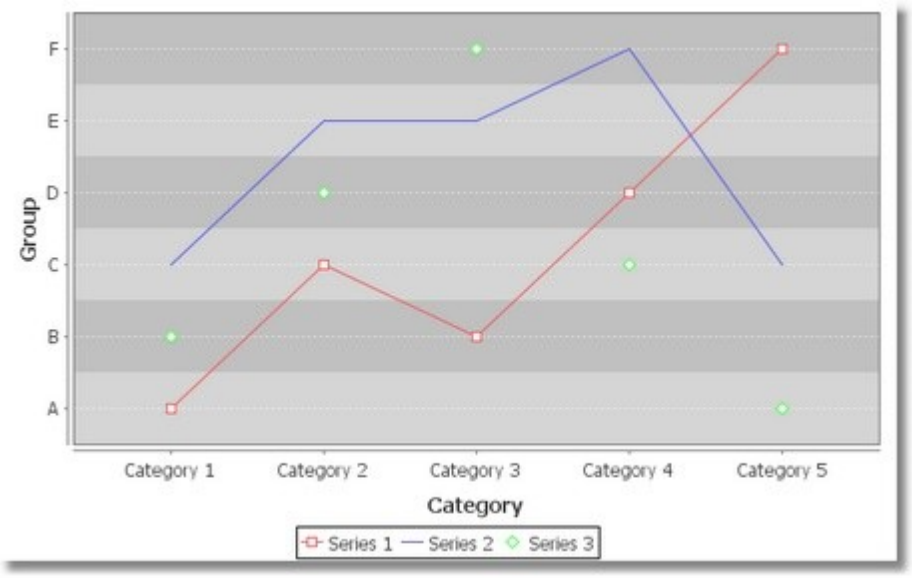
Свойство	Имя	Тип	Описание
Основание	base	плавающее	Основание для вычисления логарифма.
Наименьшее значение	smallestValue	плавающее	Возвращает наименьшее (положительное) значение, которое будет отображено на оси. Значение по умолчанию - 1E-100.
Шаг делений	tickUnit	таблица данных	Данная опция позволяет конфигурировать шаги делений, если отключен Автоматический выбор шага делений. Включает два поля: • **Размер.** Число с плавающей точкой, определяющее расстояние между двумя делениями. • **Количество малых делений**. Целое число, представляющее количество малых делений между двумя большими делениями.
Переопределение формата	overrideFormat	логическое	Включает пользовательское форматирование меток делений.
Метка основания	baseLabel	строка	Метка основания логарифма.
Метка аргумента	powerLabel	строка	Метка аргумента логарифма.
Показать основание	showBase	логическое	Флажок, указывающий, необходимо ли отображать основание логарифма в метках.
Формат показателя степени	exponentFormatOverride	строка	Формат 2147 показателя степени логарифма.

Ось символов является подтипом оси значений[1165], которая преобразовывает целочисленные значения (начиная с нуля) в символы (строки).

Дополнительные свойства оси символов:

Свойство	Имя	Тип	Описание
Символы	symbols	таблица данных	Таблица с метками значений включает в себя два поля: • **Значение**, обозначенное меткой (целое) • **Символ**, т.е. текст метки (строка)
Видимость полос сетки	gridBandsVisible	логическое	Флажок, контролирующий видимость связок линий сетки для различных значений делений.
Окраска полос сетки	gridBandPaint	таблица данных	Цвет[1279] линий сетки в области построения.
Окраска промежуточных полос сетки	gridBandAlternatePaint	таблица даннных	Альтернативный цвет[1279] линий сетки в области построения.

Линейный[1067] график, в котором ось измерений является Осью символов:



13.4.11.7.2 Фрейм

Рамка для различных блоков графика, таких как заголовок.

Свойства:

Свойство	Имя	Тип	Описание
Тип	type	целое	Тип фрейма: Линейная граница или Граница блоков.
Отступы	insets	таблица данных	Доступное пространство для построения границы. См. Отступы[1180].
Окраска	paint	таблица	Цвет[1279], используемый для отображения границы.

		данных	
Штрих	stroke	таблица данных	Штрих 1282, используемый для отображения границы.

13.4.11.7.3 Подзаголовок изображения

Подзаголовок изображения графика отображает рисунок. В основном используется для отображения логотипа или другого рисунка на графике.

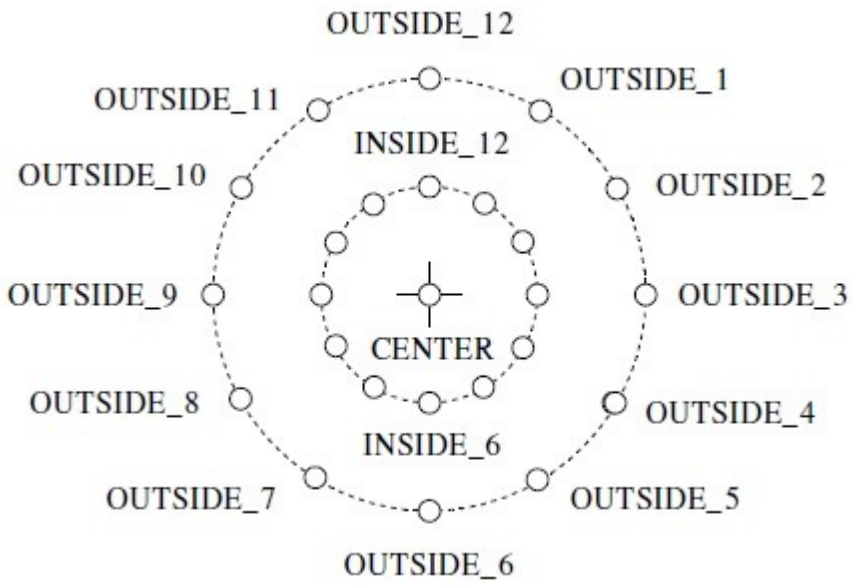
Свойства подзаголовка изображения

Свойство	Имя	Тип	Описание
Видимый	visible	логическое	Флажок, указывающий, должен ли отображаться данный заголовок.
Изображение	image	блок данных	Изображение для подзаголовка.
Авторазмер	autoSize	логическое	Использует размер самого изображения вместо параметров Ширина/Высота.
Ширина	width	плавающее	Требуемая ширина заголовка.
Высота	height	плавающее	Требуемая высота заголовка.
Границы	margins	таблица данных	Внешняя граница фрейма заголовка. См. Прямоугольные вставки 1180.
Отступы	padding	таблица данных	Область пустого пространства внутри фрейма заголовка. См. Прямоугольные вставки 1180.
Фрейм	frame	таблица данных	Граница вокрук заголовка. См. Фрейм блоков 1171.
Позиция	position	строка	Позиция заголовка на графике (Сверху, Слева, Снизу или Справа).
Горизонтальное выравнивание	horizontalAlignment	строка	Выравнивание заголовка по горизонтали. Обычно используется, если Позиция установлена на Сверху или Снизу.
Вертикальное выравнивание	verticalAlignment	строка	Выравнивание заголовка по вертикали. Обычно используется, если Позиция установлена на Слева или Справа.

13.4.11.7.4 Положение меток элементов серий

Свойства меток элементов серий используется для определения местоположения меток элементов на графике. Для этого применяются четыре свойства.

Свойства

Свойство	Имя	Тип	Описание
Точка привязки меток элементов	itemLabelAnchor	строка	Точка привязки меток элементов, используемая отрисовщиком для расчета фиксированной точки (точки привязки метки элемента) относительно элемента данных на графике. Данная точка становится ориентиром, по которому выравнивается метка элемента. Доступно 25 точек привязки. Числа от 1 до 12 соответствуют позиции часов на циферблате. Помимо этого, позиции определяются относительно "внутреннего" и "внешнего" кругов - см. приведенный далее рисунок. 12 точек на внутреннем круге, 12 точек на внешнем круге, к ним добавим "центральную" точку привязки, в итоге получим 25 возможных точек привязки. Для некоторых отрисовщиков (например, отрисовщика столбцов) круговое размещение точек привязки не подходит, поэтому отрисовщик может сам изменять позиции точек. 
Точка привязки текста	textAnchor	строка	Точка относительно текста метки элемента, которая выравнивается по точке привязки метки элемента, указанной выше. Доступными значениями являются: По центру, По центру слева, По центру справа, По центру сверху, Сверху слева, Сверху справа, По центру снизу, Снизу слева, Снизу справа, По центру базовой линии, По левому краю базовой линии, По правому краю базовой линии, По центру середины подъема, Слева по середине подъема, Справа посередине подъема.
Точка привязки вращения	rotationAnchor	строка	Еще одна точка где-либо на метке элемента, вокруг которой будет вращаться текст (при наличии вращения). Доступными значениями являются: По центру, По центру слева, По центру справа, По центру сверху, Сверху слева, Сверху справа, По центру снизу, Снизу слева, Снизу справа, По центру базовой линии, По левому краю базовой линии, По правому краю базовой линии, По центру середины подъема, Слева по середине подъема, Справа посередине подъема.
Угол	angle	плавающее	Определяет угол вращения вокруг точки вращения.

13.4.11.7.5 Легенда

Легенда отображает метки для серий данных на графике, обычно вместе с небольшим графическим элементом, определяющим серию (по цвету и/или стилю). Возможно также добавление нескольких легенд к графику.

Легенда в основном содержит один элемент легенды для каждой серии данных, отображаемой на графике. Данный элемент состоит из маленького изображения, которое определяет серию на графике, и метки, соответствующей имени серии данных.

Свойства легенды

Свойство	Имя	Тип	Описание
Видимость серий	visible	логическое	Флажок, указывающий, будет ли отображаться данная легенда.
Ширина	width	плавающее	Необходимая ширина легенды.
Высота	height	плавающее	Необходимая высота легенды.
Границы	margins	таблица данных	Внешняя граница вокруг фрейма легенды. См. Прямоугольные вставки 1180.
Отступы	padding	таблица данных	Область пустого пространства внутри фрейма легенды. См. Прямоугольные вставки 1180.
Фрейм	frame	таблица данных	Граница вокруг легенды. См. Фрейм блоков 1171.
Позиция	position	строка	Позиция легенды на графике (Сверху, Снизу, Слева или Справа).
Горизонтальное выравнивание	horizontalAlignment	строка	Выравнивание легенды по горизонтали. Обычно используется, если Позиция установлена на Сверху или Снизу.
Вертикальное выравнивание	verticalAlignment	строка	Выравнивание легенды по вертикали. Обычно используется, если Позиция установлена на Слева или Справа.
Окраска фона	backgroundPaint	таблица данных	Цвет 1279 фона легенды.
Шрифт элемента	itemFont	таблица данных	Шрифт 1278 меток элементов легенды.
Отступы для меток элементов	itemLabelPadding	таблица данных	Отступы для меток эелемнтов. См. Прямоугольные вставки 1180.
Окраска элемента	itemPaint	таблица данных	Цвет 1279 элемента легенды.
Точка привязки элемента легенды	legendItemGraphicAnchor	строка	Положение изображения элемента в его прямоугольнике определяется двумя атрибутами, точкой привязки и местоположением. Точка привязки - точка на изображении элемента, которая выравнивается с точкой местоположения. Доступными значениями точки привязки являются: По центру, Сверху, Снизу, Слева, Справа, Сверху слева, Сверху справа, Снизу слева, Снизу справа.
Положение изображения элемента легенды в тексте	legendItemGraphicEdge	строка	Положение изображения элемента относительно его текста (Сверху, Снизу, По левому краю, По правому краю).
Позиция изображения	legendItemGraphicLocation	строка	Положение изображения элемента относительно ограничивающего блока элемента легенды.

элемента легенды			Доступными значениями точки привязки являются: По центру, Сверху, Снизу, Слева, Справа, Сверху слева, Сверху справа, Снизу слева, Снизу справа.
Отступы вокруг изображения элемента легенды	legendItemGraphicPadding	таблица данных	Отступы вокруг изображения элемента легенды. См. Прямоугольные вставки 1180.

13.4.11.7.5.1 Элемент легенды

Элемент легенды записывает атрибуты элемента, которые должны появиться в легенде 1173 графика.

Элемент легенды состоит из:

- **Фигуры элемента** (например, прямоугольник или звезда)
- **Линии элемента**, внешний вид которой обычно совпадает с линией, отображающей серию данных
- **Метки элемента**, представляющей имя серии данных.

Свойства

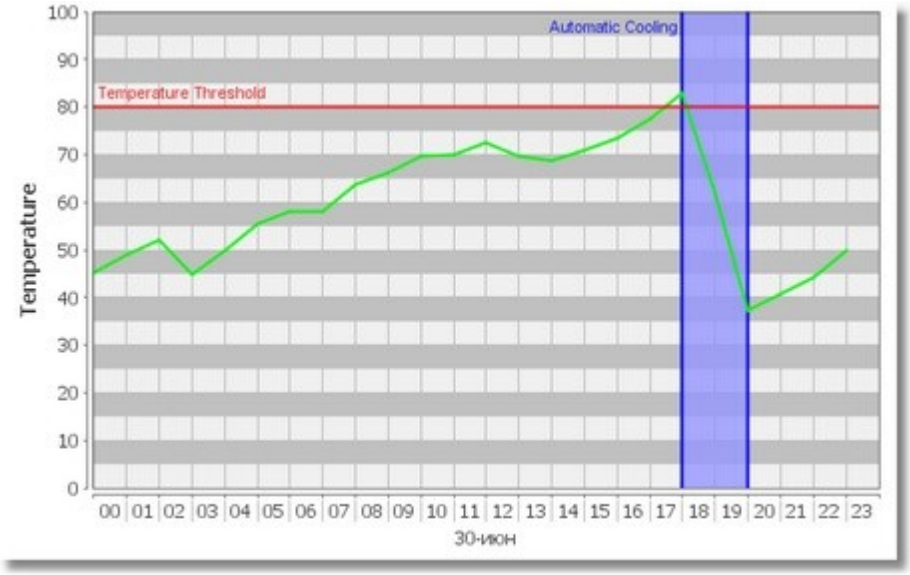
Свойство	Имя	Тип	Описание
Ключ сериии данных	seriesKey	строка	Ключ серии данных, представляемой легендой.
Метка	label	строка	Метка элемента легенды.
Шрифт меток	labelFont	таблица данных	Шрифт 1278, используемый для отображения меток элементов.
Цвет меток	labelPaint	таблица данных	Цвет 1279, используемый для отображения меток элементов.
Текст всплывающих подсказок	toolTipText	строка	Текст всплывающих подсказок для меток элементов.
Видимость	shapeVisible	логическое	Флажок, контролирующий, будет ли отображаться фигура элемента легенды.
Форма	shape	таблица данных	Фигура 1283, используемая в качестве изображения для элемента легенды.
Заливка формы	shapeFilled	логическое	Флажок, контролирующий, будет ли применяться заливка цветом фигуры элемента легенды.
Окраска заливки	fillPaint	таблица данных	Цвет 1279, используемый для отображения маркера.
Видимость окантовки формы	shapeOutlineVisible	логическое	Флажок, контролирующий, будет ли отображаться контур фигуры элемента легенды.
Окраска окантовки	outlinePaint	таблица данных	Цвет 1279, используемый для отображения маркера.
Штрих окантовки	outlineStroke	таблица данных	Штрих 1282, используемый для отображения контура маркера.
Видимость линии	lineVisible	логическое	Флажок, контролирующий, будет ли отображаться линия как часть изображения элемента легенды.
Линия	line	таблица данных	Линия, если таковая имеется, отображаемая для рисунка элемента легенды. См. Фигуры 1283.
Линия штриха	lineStroke	таблица данных	Штрих 1282, используемый для отображения линии рисунка элемента легенды.

Окраска линии	linePaint	таблица данных	Цвет1279, используемый для отображения линии для рисунка элемента легенды.

13.4.11.7.6 Маркер

Маркеры используются для выделения определенных значений или диапазонов значений напротив оси определений или оси измерений. Маркеры могут отображаться с или без меток.

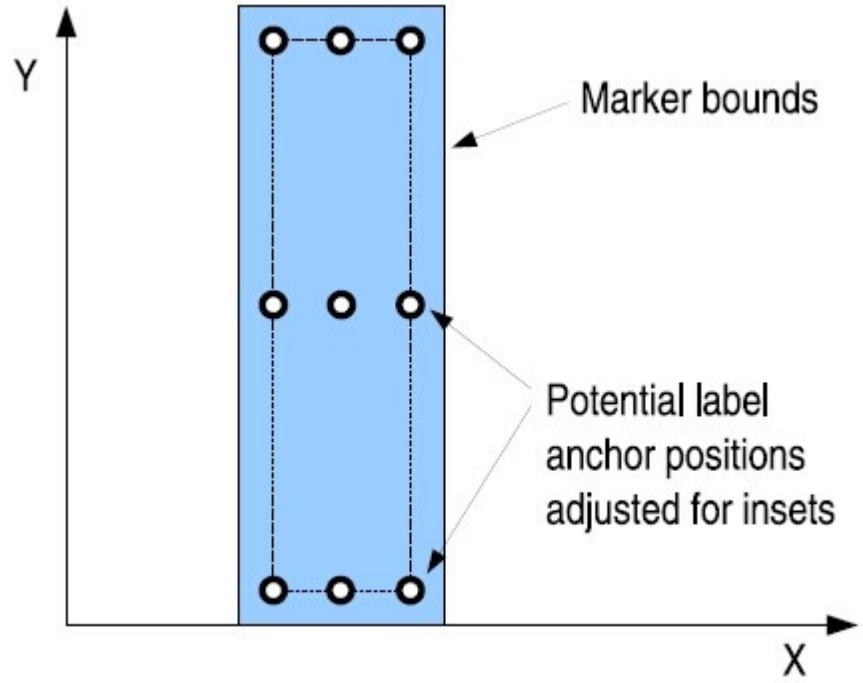
Приведенный далее рисунок представляет применение маркеров на графике:



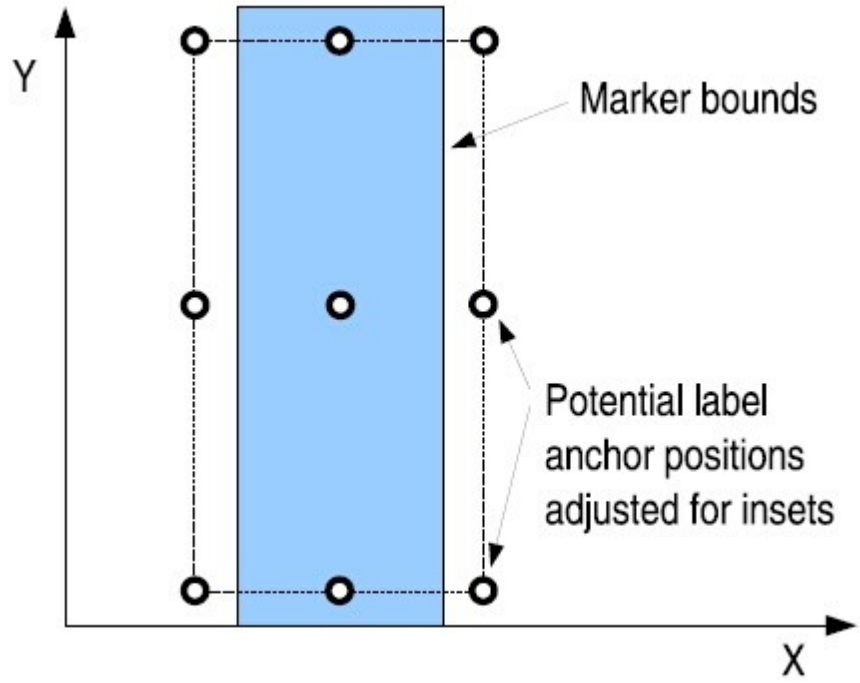
ПОЗИЦИЯ МЕТКИ

Рисунок ниже иллюстрирует, как рассчитывается позиция точки привязки метки маркера относительно его границ. Одна из девяти потенциальных точек привязки выбирается через свойство Точка привязки метки, а расстояние между границами маркера и потеницальными точками привязки определяется методами Смещение метки и Тип смещения метки.

Тип смещения метки Сжатие:



Тип смещения метки Расширение:



Свойства

Данные свойства применимы ко всем типам маркеров. Каждый тип маркеров имеет дополнительные свойства, по крайней мере, определяющие их положение (например, маркеры категорий, значений и т.д.)

Свойство	Имя	Тип	Описание
Цвет	paint	таблица данных	Цвет1279, используемый для отображения маркера.
Штрих	stroke	таблица данных	Штрих1282, используемый для маркера, если он отображается в виде линии. Если маркер представлен в виде прямоугольника, контур прорисовывается при помощи атрибута Штрих контура, поэтому в данном случае этот атрибут не применяется.
Окраска контура	outlinePaint	таблица данных	Цвет1279, используемый для отображения контура маркера. Данное поле не применяется, если маркер является линейным.
Штрих контура	outlineStroke	таблица данных	Штрих1282, используемый для отображения контура маркера. Данное поле не применяется, если маркер является линейным.
Прозрачность	alpha	плавающее	Альфа прозрачность, используемая при отображении маркера. Является значением в пределах от 0.0 (полностью прозрачный) до 1.0f (полностью непрозрачный).
Метка	label	строка	Текст метки (может быть пустым). Если строка метки пустая (по умолчанию), маркер отображается без метки.
Шрифт меток	labelFont	таблица данных	Штрих1282, используемый для отображения меток.
Окраска меток	labelPaint	таблица данных	Цвет1279, используемый для отображения меток.
Точка привязки меток	labelAnchor	строка	Точка на границах маркера, необходимая для выравнивания метки. Данная точка привязки (после ее настройки смещениями метки) определяет фиксированную точку на графике, по которой выравнивается метка маркера. Доступными значениями точки привязки являются: По центру, Сверху, Снизу, Слева, Справа, Сверху слева, Сверху справа, Снизу слева и Снизу справа.

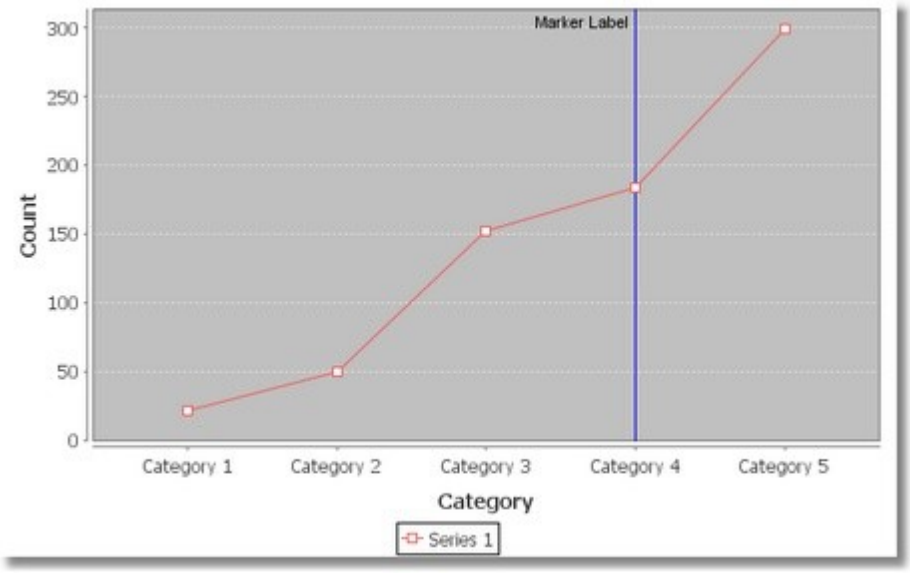
Точка привязки текста	labelTextAnchor	строка	Точка на метке, которая выравнивается относительно фиксированной точки на графике, определяемой свойством Точка привязки меток.
Смещение меток	labelOffset	таблица данных	Смещение между границами маркера и точками привязки метки. Значение по умолчанию - (3.0, 3.0, 3.0, 3.0), т.е. точки привязки лежат в трех единицах внутри (снаружи) ограничительного прямоугольника маркера. См. Прямоугольные вставки[1180].
Тип смещения меток	labelOffsetType	строка	Тип смещения меток: **Без изменений**, **Расширение**, **Сжатие**. Контролирует, как применяются вставки, полученные в результате Смещения меток, для расчета позиции точки привязки метки.
Тип	type	целое	Тип маркера: Маркер категорий[1178], Маркер значений[1179], или Маркер интервалов[1180]. Поле может быть недоступным, если применяется только один тип маркера.

13.4.11.7.6.1 Маркер категорий

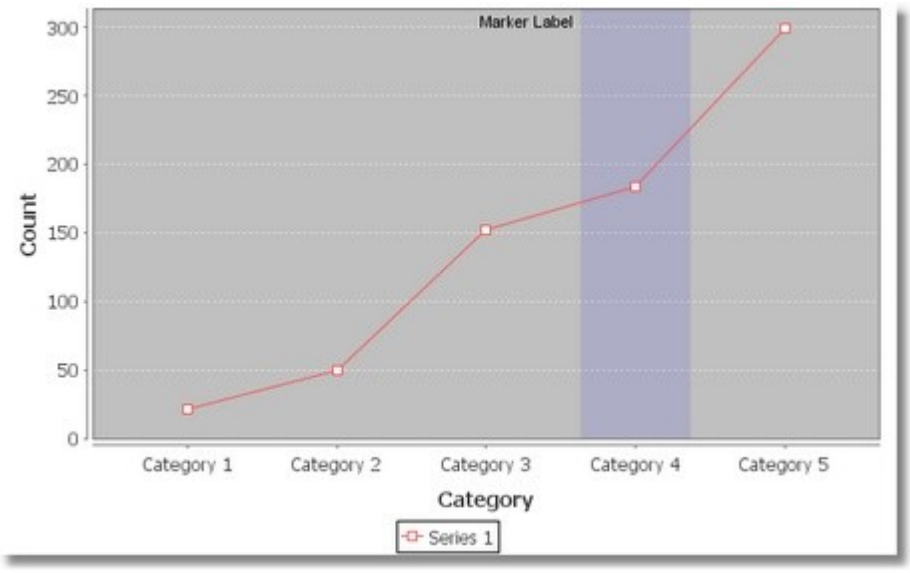
Маркер категорий используется для выделения категории в области построения категорий.

Маркер категорий отображается в виде **линии** или **прямоугольника**.

Режим линии:



Режим прямоугольника:



Свойства

Список типичных свойств маркера доступен в разделе Маркер 1176.

Маркеры категорий имеют следующие дополнительные свойства:

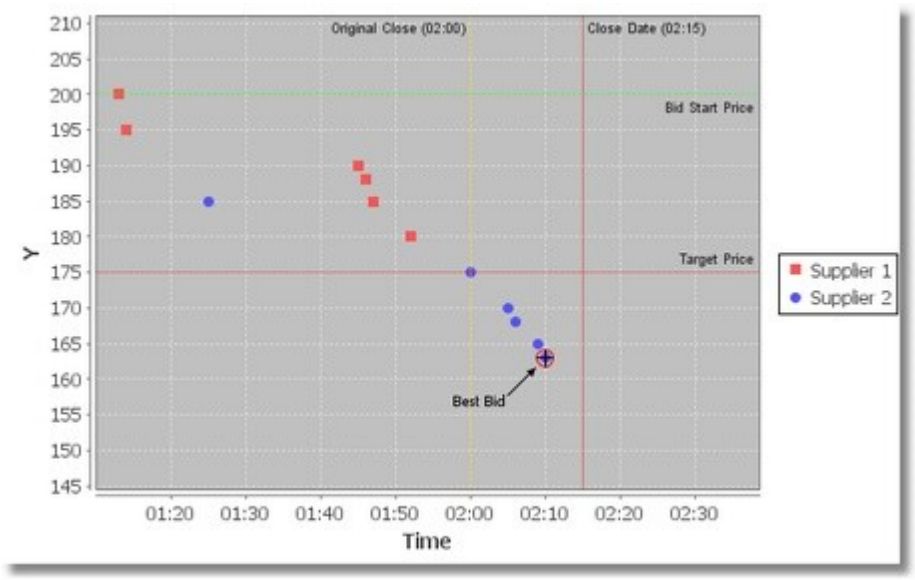
Свойство	Имя	Тип	Описание
Ключ категории	key	строка	Категория, которую необходимо выделить.
Отображать в виде линии	drawAsLine	логическое	Флажок, контролирующий, будет ли маркер отображаться в виде линии или прямоугольника.

13.4.11.7.6.2 Маркер значений

Маркер значений используется для указания постоянного значения напротив оси определений или оси измерений для категорийной 1187 или координатной 1197 областей построения.

Маркер чаще всего отображается в виде линии, однако, на графике с эффектом 3D маркер представлен в виде многоугольника. По этой причине он имеет атрибуты Цвет и Контур, Штрих и Контур.

Рисунок далее представляет график с несколькими маркерами значений и указателем с аннотацией.



Свойства

Список типичных свойств маркера см. в разделе Маркер 1176.

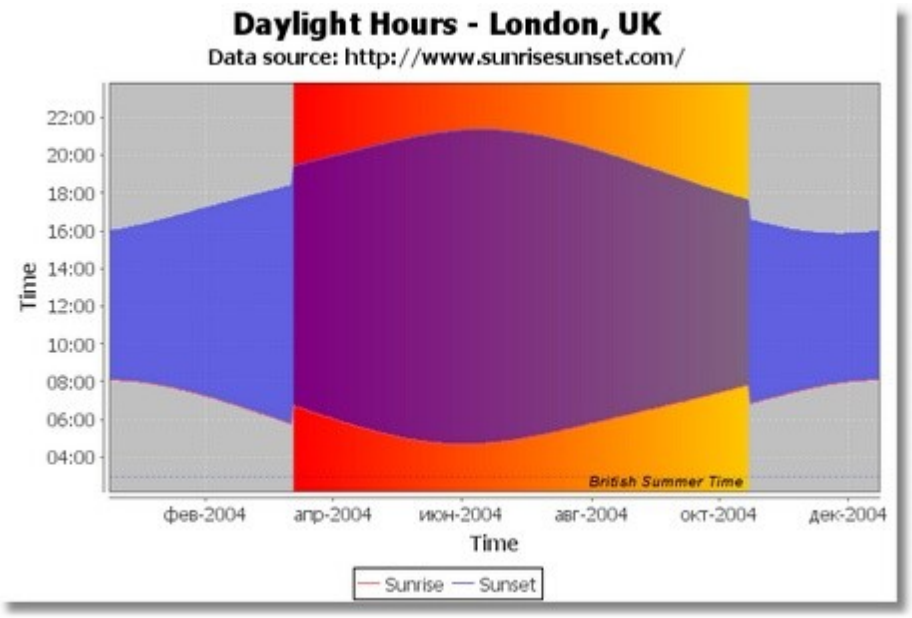
Маркеры значений имеют следующие дополнительные свойства:

Свойство	Имя	Тип	Описание
Значение	value	плавающее	Значение, которое необходимо выделить.

13.4.11.7.6.3 Маркер интервалов

Маркер интервалов используется для выделения определенного диапазона значений напротив оси определений или оси измерений для категорийной[1187] или координатной[1197] областей построения.

Приведенный ниже рисунок отображает маркер интервалов, который выделяет диапазон значений по оси определений:



Свойства

Список типичных свойств маркера см. в разделе Маркер[1176].

Маркеры интервалов значений имеют следующие дополнительные свойства:

Свойство	Имя	Тип	Описание
Начальное значение	startValue	плавающее	Нижняя граница выделяемого интервала.
Конечное значение	endValue	плавающее	Верхняя граница выделяемого интервала.

13.4.11.7.7 Отступы

Данный класс используется для определения отступов слева, справа, сверху и снизу относительно произовольного прямоугольника. Пространство может определяться в абсолютных величинах (пикселях) или относительным выражением (в процентах от высоты или ширины прямоугольника).

Свойства

Свойство	Имя	Тип	Описание
Сверху	top	плавающее	Верхнее значение вставок.
Слева	left	плавающее	Левое значение вставок.
Снизу	bottom	плавающее	Нижнее значение вставок.
Справа	right	плавающее	Правое значение вставок.

Тип единиц	unitType	целое	Абсолютные или Относительные.

13.4.11.7.8 Заголовок

Текстовый заголовок или подзаголовок графика представляет собой текстовую строку. Длинные заголовки автоматически переносятся на следующую строку, вы также можете задать перенос строки, вставив символ новой строки в текст заголовка.

Свойства заголовка

Свойство	Имя	Тип	Описание
Видимый	visible	логическое	Флажок, указывающий, будет ли отображаться заголовок.
Текст	text	строка	Текст заголовка.
Текст подсказки	toolTipText	строка	Текст, отображаемый в качестве всплывающей подсказки для заголовка. Значение может быть пустым, вследствие этого всплывающая подсказка не будет появляться.
Шрифт	font	таблица данных	Шрифт 1278 заголовка.
Ширина	width	плавающее	Предпочитаемая ширина заголовка.
Высота	height	плавающее	Предпочитаемая высота заголовка.
Границы	margins	таблица данных	Внешние границы вокруг фрейма заголовка. См. Прямоугольные вставки 1180.
Отступы	padding	таблица данных	Область пустого пространства внутри фрейма заголовка. См. Прямоугольные вставки 1180.
Фрейм	frame	таблица данных	Граница, очерчиваемая вокруг заголовка. См. Фрейм блоков 1171.
Позиция	position	строка	Позиция заголовка на графике (Сверху, Слева, Снизу или Справа).
Выравнивание по горизонтали	horizontalAlignment	строка	Выравнивание заголовка по горизонтали. Обычно применяется, когда Позиция установлена на Сверху или Снизу.
Выравнивание по вертикали	verticalAlignment	строка	Выравнивание заголовка по вертикали. Обычно применяется, когда Позиция установлена на Слева или Справа.
Окраска фона	backgroundPaint	таблица данных	Цвет 1279 заливки фона в пределах границ заголовка.
Окраска	paint	таблица данных	Цвет 1279 отображения текста заголовка.
Выравнивание текста	textAlignment	строка	Выравнивание текста в пределах границ заголовка (По левому краю, По центру, По правому краю). Контролирует выравнивание текста в пределах

			границ заголовка, тогда как выравнивание заголовка по горизонтали контролирует то, как ограничительный прямоугольник заголовка выравнивается в пределах пространства построения.
Расширение границ заголовка	expandToFitSpace	логическое	Флажок, контролирующий, будет ли выполняться раширение границ заголовка для его заполнения доступного пространства.
Максимальное количество отображаемых строк	maximumLinesToDisplay	целое	Максимальное количество отображаемых строк.

13.4.11.8 Общие события графиков

Этот раздел описывает события, поддерживаемые всеми графиками.

КЛИК НА ГРАФИКЕ

Событие формируется, когда кликают на определеннаой секции графика (вне ее области построения, осей, легенды и заголовков, у которых есть свои события клика).

Имя события: **chartClick**

Поля события: это событие имеет все поля обычного события мыши 1288. Также оно определяет следующие поля:

Поле	Имя	Тип	Описание
Текст всплывающей подсказки	toolTipText	строка	Содержит всплывающую подсказку секции графика, на которой кликнули.
Область	area	таблица данных	Форма области графика, на которой кликнули. См. Формы 1283 для получения более подробной информации.

КЛИК НА ОБЛАСТИ ПОСТРОЕНИЯ

Событие формируется, когда кликают на области построения графика.

Имя события: **plotClick**

Поля события: это событие имеет все поля обычного события мыши 1288. Также оно определяет следующие поля:

Поле	Имя	Тип	Описание
Текст всплывающей подсказки	toolTipText	строка	Содержит всплывающую подсказку секции графика, на которой кликнули.
Область	area	таблица данных	Форма области графика, на которой кликнули. См. Формы 1283 для получения более подробной информации.

КЛИК НА ЭЛЕМЕНТЕ ЛЕГЕНДЫ

Это событие формируется, когда кликают на элементе легенды графика.

Имя события: **legendItemClick**

Поля события: это событие имеет все поля обычного события мыши 1288. Также оно определяет следующие поля:

Поле	Имя	Тип	Описание
Текст всплывающей подсказки	toolTipText	строка	Содержит всплывающую подсказку секции графика, на которой кликнули.
Область	area	таблица данных	Форма области графика, на которой кликнули. См. Формы 1283 для получения более подробной информации.

Индекс массива данных	datasetIndex	целое	Индекс массива данных, к которому относится легенда.
Ключ серии	seriesKey	строка	Ключ серии, к которой относится легенда.

КЛИК НА ЗАГОЛОВКЕ

Это событие формируется, когда кликают на заголовке графика.

Имя события: **titleClick**

Поля события: это событие имеет все поля обычного события мыши 1288. Также оно определяет следующие поля:

Поле	Имя	Тип	Описание
Текст всплывающей подсказки	toolTipText	строка	Содержит всплывающую подсказку секции графика, на которой кликнули.
Область	area	таблица данных	Форма области графика, на которой кликнули. См. Формы 1283 для получения более подробной информации.

КЛИК НА ОСИ

Это событие формируется, когда кликают на оси графика.

Имя события: **axisClick**

Поля события: это событие имеет все поля обычного события мыши 1288. Также оно определяет следующие поля:

Поле	Имя	Тип	Описание
Текст всплывающей подсказки	toolTipText	строка	Содержит всплывающую подсказку секции графика, на которой кликнули.
Область	area	таблица данных	Форма области графика, на которой кликнули. См. Формы 1283 для получения более подробной информации.
Индекс оси параметров	domainAxisIndex	целое	Индекс оси параметров, на которой кликнули, или null, если кликнули на оси значений.
Индекс оси значений	rangeAxisIndex	целое	Индекс оси значений, на которой кликнули, или null, если кликнули на оси параметров.

КЛИК НА МЕТКЕ КАТЕГОРИИ

Это событие формируется, когда кликают на метке графика категорий. Доступно только для графиков категорий 1067.

Имя события: **categoryLabelClick**

Поля события: это событие имеет все поля обычного события мыши 1288. Также оно определяет следующие поля:

Поле	Имя	Тип	Описание
Текст всплывающей подсказки	toolTipText	строка	Содержит всплывающую подсказку секции графика, на которой кликнули.
Область	area	таблица данных	Форма области графика, на которой кликнули. См. Формы 1283 для получения более подробной информации.
Ключ	key	строка	Ключ массива данных категории, на котором кликнули.

КЛИК НА ЭЛЕМЕНТЕ КАТЕГОРИИ

Это событие формируется, когда кликают на элементе графика категорий. Доступно только для графиков категорий 1067.

Имя события: **categoryItemClick**

Поля события: это событие имеет все поля обычного события мыши 1288. Также оно определяет следующие поля:

Поле	Имя	Тип	Описание
Текст всплывающей подсказки	toolTipText	строка	Содержит всплывающую подсказку секции графика, на которой кликнули.
Область	area	таблица данных	Форма области графика, на которой кликнули. См. Формы 1283 для получения более подробной информации.
Индекс массива данных	datasetIndex	целое	Индекс массива данных элемента, на котором кликнули.
Ключ строки	rowKey	строка	Ключ массива данных строки, на котором кликнули.
Ключ столбца	columnKey	строка	Ключ массива данных столбца, на котором кликнули.

КЛИК НА ЭЛЕМЕНТЕ XY

Это событие возникает, когда кликают на элементе координатного графика XY. Доступно только для графика XY 1099.

Имя события: **xyItemClick**

Поля события: это событие имеет все поля обычного события мыши 1288. Также оно определяет следующие поля:

Поле	Имя	Тип	Описание
Текст всплывающей подсказки	toolTipText	строка	Содержит всплывающую подсказку секции графика, на которой кликнули.
Область	area	таблица данных	Форма области графика, на которой кликнули. См. Формы 1283 для получения более подробной информации.
Индекс массива данных	datasetIndex	целое	Индекс массива данных элемента, на котором кликнули.
Серия	series	целое	Индекс серии элемента, на которой кликнули.
Элемент	item	целое	Индека элемента, на котором кликнули.

КЛИК НА КРУГОВОМ СЕКТОРЕ

Это событие формируется, когда кликают на секторе круговой диаграммы. Доступно только для круговой диаграммы 1156.

Имя события: **pieSectionClick**

Поля события: это событие имеет все поля обычного события мыши 1288. Также оно определяет следующие поля:

Поле	Имя	Тип	Описание
Текст всплывающей подсказки	toolTipText	строка	Содержит всплывающую подсказку секции графика, на которой кликнули.
Область	area	таблица данных	Форма области графика, на которой кликнули. См. Формы 1283 для получения более подробной информации.
Индекс кругового сектора	pieIndex	целое	Индекс сектора круговой диаграммы с тем элементом, на котором кликнули.
Индекс секции	sectionIndex	целое	Индекс секции, на котором кликнули.
Ключ секции	sectionKey	строка	Ключ секции, на котором кликнули.

МАССИВ ДАННЫХ ОБНОВЛЕН

Это событие формируется каждый раз, когда график получает новый массив данных.

Имя события: **datasetUpdated**

Поля события: нет полей.

13.4.11.9 Области построения графиков

Область построения контролирует визуальное представление данных на графике.

Данный раздел посвящен описанию свойств, поддерживаемых всеми областями построения.

Граница области построения

Внешняя граница очерчивается вокруг большинства типов областей построения. Ее вид можно менять, редактируя цвет и штрих, используемые для ее отображения на графике.

Контур отображается вокруг всех четырех сторон области построения. Имейте в виду, что каждая из осей области построения может также отображать линию по краю области данных. См. свойство оси **Видимая линия оси**.

ВИДИМОСТЬ ОКАНТОВКИ

Флажок, контролирующий, будет ли отображаться контур области построения.

Имя свойства: **outlineVisible**

Тип свойства: **Логическое**

ОКРАСКА ОКАНТОВКИ

Цвет 1279, используемый для отображения контура вокруг области построения.

Имя свойства: **outlinePaint**

Тип свойства: **Таблица данных**

ШТРИХ ОКАНТОВКИ

Штрих 1282, используемый для отображения контура вокруг области построения.

Имя свойства: **outlineStroke**

Тип свойства: **Таблица данных**

Фон области построения

Фон области построения представляет собой область между осями (если область построения имеет оси). Она не включает заголовки графика, легенду или метки оси.

ОКРАСКА ФОНА

Цвет 1279 заливки фона области построения.

Имя свойства: **backgroundPaint**

Тип свойства: **Таблица данных**

ПРОЗРАЧНОСТЬ ФОНА

Альфа-прозрачность фона области построения. Допустимый диапазон значений -- от 0.0f (полностью прозраный) до 1.0f (полностью непрозрачный).

Имя свойства: **backgroundAlpha**

Тип свойства: **Плавающее**

ФОНОВОЕ ИЗОБРАЖЕНИЕ

Фоновое изображение области построения.

Имя свойства: **backgroundImage**

Тип свойства: **Бинарный блок**

ВЫРАВНИВАНИЕ ФОНОВОГО ИЗОБРАЖЕНИЯ

Тип выравнивания фонового изображения. Допустимые значения:

- Немасштабный: По центру, Снизу, Слева, Справа, Сверху слева, Сверху справа, Снизу слева, Снизу справа.
- Масштабный: По вертикали, По горизонтали и Растянуть (до полного заполнения области построения).

Имя свойства: **backgroundImageAlignment**

Тип свойства: **Целое**

ПРОЗРАЧНОСТЬ ФОНОВОГО ИЗОБРАЖЕНИЯ

Альфа-прозрачность фонового изображения области построения.

Имя свойства: **backgroundImageAlpha**

Тип свойства: **Плавающее**

Сообщение "Нет данных"

Некоторые области построения выдают сообщение при отсутствии данных, доступных для отображения. Сообщение является простой строкой, контролируемой следующими свойствами:

СООБЩЕНИЕ ОТСУТСТВИЯ ДАННЫХ

Строка, появляющаяся при отсутствии данных для отображения в области построения.

Имя свойства: **noDataMessage**

Тип свойства: **Строка**

ШРИФТ СООБЩЕНИЯ ОТСУТСТВИЯ ДАННЫХ

Шрифт[1278], используемый для отображения сообщения "нет данных".

Имя свойства: **noDataMessageFont**

Тип свойства: **Таблица данных**

ОКРАСКА СООБЩЕНИЯ ОТСУТСТВИЯ ДАННЫХ

Цвет[1279], используемый для отображения сообщения "нет данных".

Имя свойства: **noDataMessagePaint**

Тип свойства: **Таблица данных**

Другие свойства

ОТСТУПЫ

Пустое пространство вокруг области построения. См. Прямоугольные вставки[1180].

Имя свойства: **insets**

Тип свойства: **Таблица данных**

ПРОЗРАЧНОСТЬ

Альфа-прозрачность переднего плана, используемая при отображении элементов на переднем плане области построения.

Имя свойства: **foregroundAlpha**

Тип свойства: **Плавающее**

ИНДИКАТОР ЗАГРУЗКИ ДАННЫХ

Флажок, контролирующий, будет ли виден индиактор загрузки данных.

Property name: **dataLoadingIndicator**

Property type: **Логическое**

13.4.11.9.1 Область построения категорий

Категорийная область построения обычно используется для отображения столбчатых графиков, тем не менее, поддерживает также и линейные графики, графики областей и стеков и т.д.

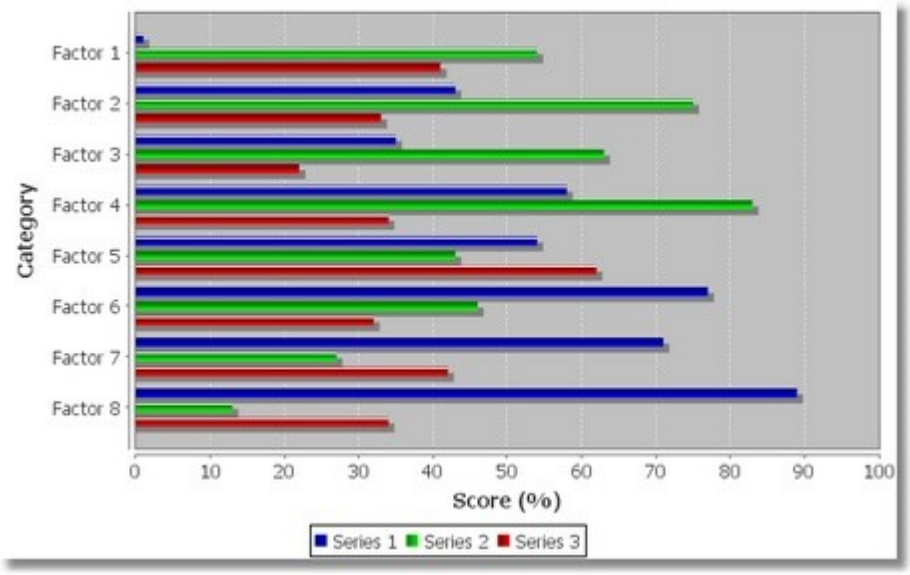
Категорийная область построения обладает:

- Одной и более осями определений, отображающих строковые имена категорий. Данные оси являются осями Категорий 1163.
- Одной и более осями измерений, отображающих числовые значения. Данные оси являются осями Значений 1165.

Категорийные области построения могут быть отображены при помощи одной из двух типов ориентации:

- **Горизонтальная ориентация**: Ось параметров (категорий) появится слева или справа на графике, а ось значений появится сверху или снизу;
- **Вертикальная ориентация**: Ось параметров (категорий) появится сверху или снизу на графике, а ось значений появится слева или справа.

Категорийная область построения с горизонтальной ориентацией:



Цвет серий данных контролируется отрисовщиком области построения.

Область построения поддерживает горизонтальную прокрутку и масштабирование вдоль оси измерений.

Данный раздел описывает свойства категорийной области построения. Для описания типичных свойств области построения обратитесь к разделу Области построения графиков 1185.

ПРИБЛИЖЕНИЕ ЗНАЧЕНИЯ

Контролирует, включено ли приближение мыши вдоль оси значений.

Имя свойства: **rangeZoomable**

Тип свойства: **Boolean**

Порядок отображения элементов

Внутри каждого массива элементы данных отображаются в виде столбцов или линий, в восходящем порядке (по умолчанию). В некоторых случаях вам может понадобиться изменить порядок отображения элементов (например, если отрисовщик отображает элементы таким образом, что происходит их наложение, как в случае с 3D отрисовщиком столбцов):

ПОРЯДОК ОТРИСОВКИ СТОЛБЦОВ

Порядок отображения элементов данных в виде столбцов.

Имя свойства: **columnRenderingOrder**

Тип свойства: **String**

ПОРЯДОК ОТРИСОВКИ СТРОК

Порядок отображения элементов данных в виде строк.

Имя свойства: **rowRenderingOrder**

Тип свойства: **String**

Аннотации области построения

Для выделения некоторых элементов данных в области построения могут быть добавлены аннотации. Доступны следующие стандартные типы аннотаций:

- Линия 1193
- Текст 1193
- Указатель 1195

АННОТАЦИИ

Список аннотаций.

Имя свойства: **annotations**

Тип свойства: **Data Table**

Маркеры

Маркеры используются для выделения различных категорий (маркеры параметров) или значений (маркеры значений).

Каждый маркер может быть помещен в один из двух *слоев*:

- **Слой переднего плана**: маркер отображаеся поверх серий данных.
- **Фоновый слой**: маркер отображается под сериями данных.

МАРКЕРЫ ПАРАМЕТРОВ

Таблица, содержащая категорийные маркеры 1178 и их слои.

Имя свойства: **domainMarkers**

Тип свойства: **Data Table**

МАРКЕРЫ ЗНАЧЕНИЙ

Таблица, содержащая маркеры 1178 значений и их слои.

Имя свойства: **rangeMarkers**

Тип свойства: **Data Table**

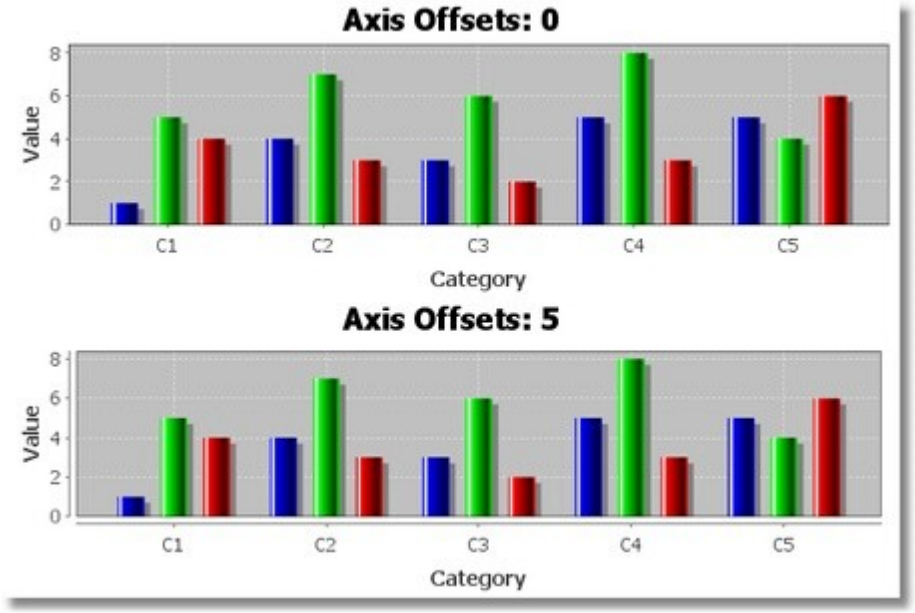
Свойства осей

Категорийная область построения обычно имеет одну ось определений и одну ось измерений. Однако могут быть добавлены дополнительные оси.

СМЕЩЕНИЕ ОСИ

Оси при необходимости могут быть немного смещены от краев области построения. Данное свойство контролирует смещение между областью построения графика и осями. См. Прямоугольные вставки 1180.

Пример смещения оси:



Имя свойства: **axisOffset**

Тип свойства: **Data Table**

ОСИ ПАРАМЕТРОВ

Таблица, содержащая оси параметров (категорий) графика и их позиции.

Существуют четыре варианта положения оси параметров:

- **Сверху или Слева**: Сверху, если ориентация области построения является вертикальной, и слева, если ориентация области построения является горизонтальной
- **Сверху или Справа**: Сверху, если ориентация области построения является вертикальной, и справа, если ориентация области построения является горизонтальной
- **Снизу или Слева**: Снизу, если ориентация области построения является вертикальной, и слева, если ориентация области построения является горизонтальной
- **Снизу или Справа**: Снизу, если ориентация области построения является вертикальной, и справа, если ориентация области построения является горизонтальной

Имя свойства: **domainAxes**

Тип свойства: **Data Table**

ОСИ ЗНАЧЕНИЙ

Таблица, содержащая оси значений графика и их позиции.

Существует четыре варианта положения оси значений:

- **Сверху или Слева**: Сверху, если ориентация области построения является горизонтальной, и слева, если ориентация области построения является вертикальной
- **Сверху или Справа**: Сверху, если ориентация области построения является горизонтальной, и справа, если ориентация области построения является вертикальной
- **Снизу или Слева**: Снизу, если ориентация области построения является горизонтальной, и слева, если ориентация области построения является вертикальной
- **Снизу или Справа**: Снизу, если ориентация области построения является горизонтальной, и справа, если ориентация области построения является вертикальной

Имя свойства: **rangeAxes**

Тип свойства: **Data Table**

ВИДИМОСТЬ БАЗОВОЙ ЛИНИИ ЗНАЧЕНИЙ

Флажок, контролирующий видимость нулевой линии напротив оси измерений.

Нулевая линия является базовой линией напротив оси измерений около нулевого значения.

Имя свойства: **rangeZeroBaselineVisible**

Тип свойства: **Data Table**

ШТРИХ БАЗОВОЙ ЛИНИИ ЗНАЧЕНИЙ

Штрих 1282 нулевой линии напротив оси измерений.

Имя свойства: **rangeZeroBaselineStroke**

Тип свойства: **Data Table**

ОКРАСКА НУЛЕВОЙ ЛИНИИ

Цвет 1279 нулевой линии напротив оси измерений.

Имя свойства: **rangeZeroBaselinePaint**

Тип свойства: **Data Table**

Фиксированные размеры оси

Ширина и высота осей обычно определяются автоматически для использования необходимого пространства, т.е. ни больше, ни меньше. В некоторых случаях вам может понадобиться изменить данное поведение и задать определенное пространство для расположения осей. Это послужит упрощению выравнивания содержимого множества графиков.

Таблица фиксированного пространства осей содержит четыре значения (Сверху, Снизу, Слева, Справа), которые определяют пространство для расположения осей в верхней, нижней, левой и правой частях области построения соответственно. Так как область построения может включать в себя несколько осей, данные значения сопоставляют требования к пространству всех осей.

ФИКСИРОВАННОЕ ПРОСТРАНСТВО ОСИ ПАРАМЕТРОВ

Задает фиксированное пространство для отображения оси параметров.

Имя свойства: **fixedDomainAxisSpace**

Тип свойства: **Data Table**

ФИКСИРОВАННОЕ ПРОСТРАНСТВО ОСИ ЗНАЧЕНИЙ

Задает фиксированное пространство для отображения оси значений.

Имя свойства: **fixedRangeAxisSpace**

Тип свойства: **Data Table**

Перекрестия

Категорийная область построения поддерживает перекрестия для основных осей параметров и значений.

Перекрестия устанавливаются при нажатии мыши на графике в работающем виджете.

ВИДИМОЕ ПЕРЕКРЕСТИЕ НА ОСИ ПАРАМЕТРОВ

Флажок, контролирующий, будет ли отображаться перекрестие на оси параметров.

Имя свойства: **domainCrosshairVisible**

Тип свойства: **Boolean**

КАТЕГОРИЯ ПЕРЕКРЕСТИЯ НА ОСИ ПАРАМЕТРОВ

Ключ категории для точки перекрестия на оси параметров.

Имя свойства: **domainCrosshairColumnKey**

Тип свойства: **String**

СЕРИЯ ПЕРЕКРЕСТИЯ НА ОСИ ПАРАМЕТРОВ

Ключ серии для точки перекрестия на оси параметров.

Данное свойство поддерживается только Линейными 1067 графиками и графиками Ганта 1090.

Имя свойства: **domainCrosshairRowKey**

Тип свойства: **String**

ШТРИХ ПЕРЕКРЕСТИЯ ПАРАМЕТРОВ

Штрих 1282 перекрестия на оси параметров, если оно видимое.

Имя свойства: **domainCrosshairStroke**

Тип свойства: **Data Table**

ОКРАСКА ПЕРЕКРЕСТИЯ НА ОСИ ПАРАМЕТРОВ

Цвет 1279 перекрестия на оси параметров, если оно видимое.

Имя свойства: **domainCrosshairPaint**

Тип свойства: **Data Table**

ВИДИМОЕ ПЕРЕКРЕСТИЕ НА ОСИ ЗНАЧЕНИЙ

Флажок, контролирующий видимость перекрестия на оси значений.

Имя свойства: **rangeCrosshairVisible**

Тип свойства: **Boolean**

ЗНАЧЕНИЕ ПЕРЕКРЕСТИЯ НА ОСИ ЗНАЧЕНИЙ

Значение точки перекрестия на оси значений.

Имя свойства: **rangeCrosshairValue**

Тип свойства: **Float**

ШТРИХ ПЕРЕКРЕСТИЯ НА ОСИ ЗНАЧЕНИЙ

Штрих 1282, используемый для отображения перекрестия на оси значений, если оно видимое.

Имя свойства: **rangeCrosshairStroke**

Тип свойства: **Data Table**

ОКРАСКА ПЕРЕКРЕСТИЯ НА ОСИ ЗНАЧЕНИЙ

Цвет 1279, используемый для отображения перекрестия на оси значений, если оно видимое.

Имя свойства: **rangeCrosshairPaint**

Тип свойства: **Data Table**

ЗАХВАТ ДАННЫХ ПЕРЕКРЕСТИЕМ НА ОСИ ЗНАЧЕНИЙ

Флажок, контролирующий, будет ли перекрестие на оси значений захватывать ближайшее значение данных, когда оно устанавливается по нажатию мыши на графике.

Имя свойства: **rangeCrosshairLockedOnData**

Тип свойства: **Boolean**

Линии сетки

По умолчанию Категорийная область будет отображать линии сетки только напротив основной оси измерений. Обычные линии сетки отображаются около каждого деления оси, Малые линии сетки отображаются около каждого Малого деления.

Имейте в виду, что линии сетки параметров и значений контролируются независимо друг от друга.

ВИДИМЫЕ ЛИНИИ СЕТКИ ПАРАМЕТРОВ

Флажок, контролирующий видимость линий сетки напротив оси определений.

Имя свойства: **domainGridlinesVisible**

Тип свойства: **Boolean**

ПОЗИЦИЯ ЛИНИЙ СЕТКИ ПАРАМЕТРОВ

Позиция линий сетки определений напротив оси определений: **В начале**, **В середине** или **В конце** каждой категории.

Имя свойства: **domainGridlinePosition**

Тип свойства: **String**

ШТРИХ ЛИНИЙ СЕТКИ ПАРАМЕТРОВ

Штрих 1282 линий сетки параметров.

Имя свойства: **domainGridlineStroke**

Тип свойства: **Data Table**

ОКРАСКА ЛИНИЙ СЕТКИ ПАРАМЕТРОВ

Цвет 1279 линий сетки параметров.

Имя свойства: **domainGridlinePaint**

Тип свойства: **Data Table**

ВИДИМОСТЬ ЛИНИЙ СЕТКИ ЗНАЧЕНИЙ

Флажок, контролирующий видимость линий сетки напротив оси значений.

Имя свойства: **rangeGridlinesVisible**

Тип свойства: **Boolean**

ШТРИХ ЛИНИЙ СЕТКИ ЗНАЧЕНИЙ

Штрих 1282 линий сетки значений.

Имя свойства: **rangeGridlineStroke**

Тип свойства: **Data Table**

ОКРАСКА ЛИНИЙ СЕТКИ ЗНАЧЕНИЙ

Цвет 1279 линий сетки измерений.

Имя свойства: **rangeGridlinePaint**

Тип свойства: **Data Table**

ВИДИМЫЕ МАЛЫЕ ЛИНИИ СЕТКИ ЗНАЧЕНИЙ

Флажок, контролирующий, будут ли отображаться линии сетки около значений малых делений на основной оси значений.

Имя свойства: **rangeMinorGridlinesVisible**

Тип свойства: **Boolean**

ШТРИХ МАЛЫХ ЛИНИЙ СЕТКИ ЗНАЧЕНИЙ

Штрих 1282 малых линий сетки значений.

Имя свойства: **rangeMinorGridlineStroke**

Тип свойства: **Data Table**

ОКРАСКА МАЛЫХ ЛИНИЙ СЕТКИ ЗНАЧЕНИЙ

Цвет 1279 малых линий сетки значений.

Имя свойства: **rangeMinorGridlinePaint**

Тип свойства: **Data Table**

Другие свойства

ОРИЕНТАЦИЯ

Ориентация области построения (Вертикальная или Горизонтальная). По умолчанию установлена на вертикальную.

Имя свойства: **orientation**

Тип свойства: **String**

ПРОКРУТКА ЗНАЧЕНИЙ

Флажок, контролирующий, можно ли выполнять прокрутку области построения на оси значений.

Имя свойства: **rangePannable**

Тип свойства: **Boolean**

ФИКСИРОВАННЫЕ ЭЛЕМЕНТЫ ЛЕГЕНДЫ

Все элементы легенды 1175 области построения, используемые для замещения автоматически сформированного набора элементов легенды, если он не является пустым.

Имя свойства: **fixedLegendItems**

Тип свойства: **Data Table**

13.4.11.9.1.1 Линейная аннотация

Аннотация, которая чертит линию между двумя точками в категорийной области построения (начиная с **Категории 1, Значения 1**, и заканчивая **Категорией 2, Значением 2**).

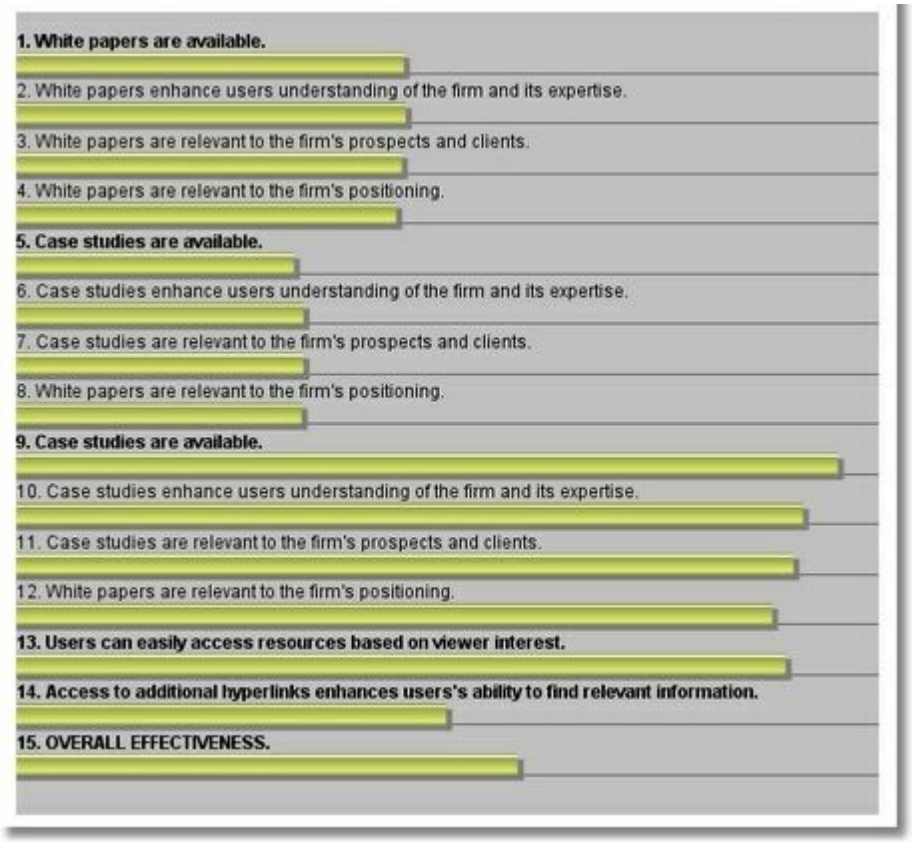
Свойства

Свойство	Имя	Тип	Описание
Категория 1	category1	строка	Категория начала линии.
Категория 2	category2	строка	Категория окончания линии.
Значение 1	value1	плавающее	Значение начала линии.
Значение 2	value2	плавающее	Значение окончания линии.
Цвет	paint	таблица данных	Цвет 1279 линии.
Штрих	stroke	таблица данных	Штрих 1282 линии.

13.4.11.9.1.2 Текстовая аннотация

Аннотация, используемая для отображения элемента текста в каком-либо месте на области построения (определяется парой **Категория, Значение**).

Пример текстовых аннотаций:



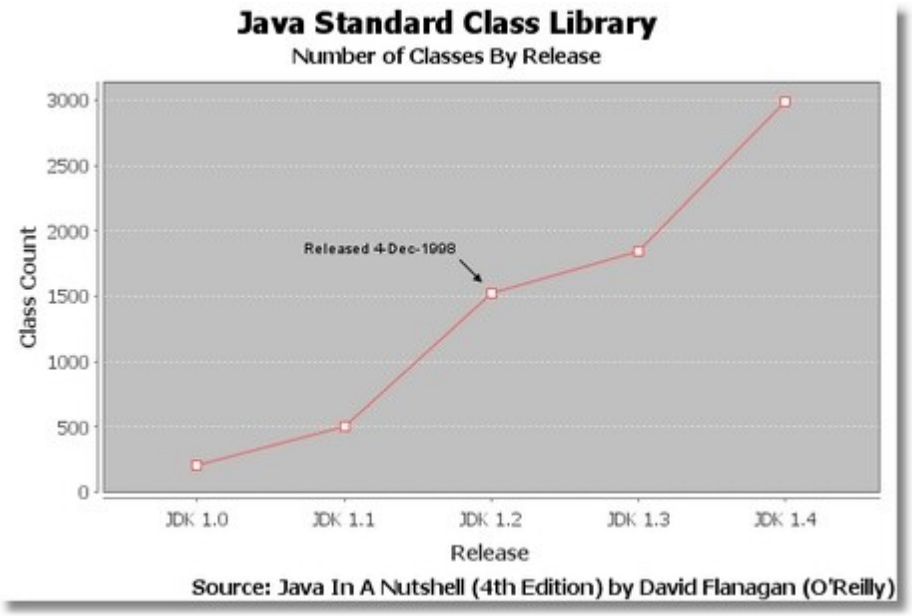
Свойства

Свойство	Имя	Тип	Описание
Категория	category	строка	Ключ категории для аннотации.
Точка привязки категории	categoryAnchor	строка	Точка привязки категории, которая помогает определить позицию точки выравнивания для аннотации.
Шрифт	font	таблица данных	Шрифт[1278] текста.
Цвет	paint	таблица данных	Цвет[1279] текста.
Точка привязки вращения	rorationAnchor	строка	Точка привязки текста, вокруг которой выполняется вращение.
Угол вращения	rorationAngle	плавающее	Угол вращения (в градусах).
Текст	text	строка	Текст, отображаемый аннотацией.
Точка привязки текста	textAnchor	строка	Точка привязки текста. Выравнивается по точке **Категория, Значение** на графике.
Значение	value	плавающее	Значение, которое определяет точку выравнивания для аннотации.

13.4.11.9.1.3 Аннотация с указателем

Аннотация для категорийной области построения, которая отображает текстовый элемент и стрелку, указывающую на точку графика, определяемую парой **Категория, Значение**.

Пример аннотации с указателем:



Свойства

Свойство	Имя	Тип	Описание
Категория	category	строка	Ключ категории для аннотации.
Точка привязки категории	categoryAnchor	строка	Точка привязки категории, которая помогает определить позицию точки выравнивания для аннотации.
Шрифт	font	таблица данных	Шрифт[1278] текста.
Цвет	paint	таблица данных	Цвет[1279] текста.
Точка привязки вращения	rorationAnchor	строка	Точка привязки текста, вокруг которой выполняется вращение.
Угол вращения	rorationAngle	плавающее	Угол вращения (в градусах).
Текст	text	строка	Текст, отображаемый аннотацией.
Точка привязки текста	textAnchor	строка	Точка привязки текста. Выравнивается по точке **Категория, Значение** на графике.
Значение	value	плавающее	Значение, которое определяет точку выравнивания для аннотации.
Угол	angle	плавающее	Угол указателя (в градусах).
Длина стрелки	arrowLength	плавающее	Длина острия стрелки.

Цвет стрелки	arrowPaint	таблица данных	Цвет 1279 стрелки.
Штрих стрелки	arrowStroke	таблица данных	Штрих 1282 стрелки.
Ширина стрелки	arrowWidth	плавающее	Ширина острия стрелки.
Радиус основания	baseRadius	плавающее	Расстояние между точкой привязки и основанием стрелки. Разница между Радиусом основания и Радиусом окончания стрелки составляет длину стрелки.
Смещение метки	labelOffset	плавающее	Смещение от основания стрелки к метке.
Радиус окончания стрелки	tipRadius	плавающее	Расстояние между точкой привязки и окончанием стрелки. Так как окончание стрелки указывает на точку привязки, данное значение должно быть меньше **Радиуса основания**.

13.4.11.9.1.4 Генератор меток элементов

Генератор меток элементов отвечает за создание текстовых строк, которые будут использоваться в качестве меток элементов в категорийном графике.

Свойства

Свойство	Имя	Тип	Описание
Формат меток	labelFormat	строка	Форматная строка, содержащая следующие символы: • {0} - имя серии • {1} - категория • {2} - строковое представление значения элемента
Тип форматера	formatType	целое	**Числовой форматер** или **Форматер дат.**
Числовой формат	numberFormat	строка	Шаблон числового форматера 2147 (только для Числового форматера).
Процентный формат	percentFormat	строка	Шаблон числового форматера 2147, используемый для процентных значений (только для Числового форматера).
Формат даты	dateFormat	строка	Шаблон формата Даты/времени 2146 (Только для Форматера дат).

13.4.11.9.1.5 Генератор всплывающих подсказок

Генератор всплывающих подсказок отвечает за создание текстовых строк, используемых в качестве всплывающих подсказок в категорийном графике.

Свойства

Свойство	Имя	Тип	Описание
Формат метки	labelFormat	строка	Форматная строка, содержащая следующие символы: • {0} - имя серии

			• {1} - категория • {2} - строковое представление значения элемента
Тип форматера	formatType	целое	**Числовой форматер** или **Форматер дат.**
Числовой формат	numberFormat	строка	Шаблон числового формата 2147 (только для Числового форматера).
Процентный формат	percentFormat	строка	Шаблон числового формата 2147, используемый для процентных значений (только для Числового форматера).
Формат дат	dateFormat	строка	Шаблон формата Даты/времени 2146 (Только для Форматера дат).

13.4.11.9.2 Координатная область построения

Координатная область построения является представлением пар значений (X, Y), где ось определений измеряет X-значения, а ось измерений - Y-значения.

Данная область построения поддерживает прокрутку и масштабирование по всем осям.

Координатная область построения обычно отображается с применением вертикальной ориентации, однако, можно ее изменить на горизонтальную, если это необходимо для некоторых приложений.

МАКЕТ ОБЛАСТИ ПОСТРОЕНИЯ

Оси отображаются слева и снизу области построения. Пространство, выделенное для осей, определяется автоматически. Следующий график показывает разделение данной области:

Определение размеров данных областей является довольно проблематичным. Размер области построения может быть изменен произвольно, но труднее изменить размеры вертикальной и горизонтальной осей. Имейте в виду, что высота вертикальной оси относительна высоте горизонтальной оси, и таким же образом ширина вертикальной оси относительна ширине горизонтальной оси. Это приводит к проблеме "курицы и яйца", потому что изменение ширины оси может повлиять на ее высоту (особенно, если изменяются деления во время изменений размера), а изменение ее высоты влияет на ширину (по той же причине).

ПРИБЛИЖЕНИЕ ПАРАМЕТРОВ

Контролирует, включено ли приближение мыши вдоль оси параметров.

Имя свойства: **domainZoomable**

Тип свойства: **Boolean**

ПРИБЛИЖЕНИЕ ЗНАЧЕНИЙ

Контролирует, включено ли приближение мыши вдоль оси значений.

Имя свойства: **rangeZoomable**

Тип свойства: **Boolean**

Квадранты

Координатная область построения обладает дополнительной функцией определения фонового цвета ее каждого квадранта.

ИСХОДНАЯ ТОЧКА КАДРАНТА

Исходная точка квадранта в пространстве данных графика, определяемая координатами X и Y. По умолчанию координатами исходной точки являются (0, 0).

Имя свойства: **quadrantOrigin**

Тип свойства: **Data Table**

ЦВЕТ КВАДРАНТОВ

Свойство	Имя	Тип	Описание
Квадрант	quadrant	целое	Определение квадранта. Существуют четыре квадранта: • Отрицательный домен, положительный диапазон • Положительны домен, положительный диапазон • Отрицательный домен, отрицательный диапазон • Положительны домен, отрицательный диапазон
Цвет	paint	таблица данных	Цвет[1279] заливки квадранта. Если является null, квадрант не заливается цветом.

Имя свойства: **quadrantsPaint**

Тип свойства: **Data Table**

Маркеры

Маркеры используются для выделения различных X-значений (маркеры доменов) или Y-значений (маркеры диапазонов).

Каждый маркер может быть расположен в одном из двух *слоев*:

- **Слой переднего плана**: маркер будет отображаться поверх серий данных
- **Фоновый слой**: маркер будет отображаться под сериями данных

МАРКЕРЫ ПАРАМЕТРОВ

Таблица, содержащая маркеры[1176] X-значений и их слои.

Имя свойства: **domainMarkers**

Тип свойства: **Data Table**

МАРКЕРЫ ЗНАЧЕНИЙ

Таблица, содержащая маркеры[1176] Y-значений и их слои.

Имя свойства: **rangeMarkers**

Тип свойства: **Data Table**

Аннотации области построения

Аннотации могут быть добавлены в область построения для выделения интересующих вас элементов данных. Доступны следующие типы стандартных аннотаций:

- Блочная аннотация[1205]

- Аннотация-изображение в координатах графика 1206
- Аннотация-изображение 1206
- Аннотация-легенда 1208
- Аннотация-линия 1208
- Аннотация -указатель 1209
- Аннотация-многоугольник 1211
- Аннотация-фигура 1212
- Текстовая аннотация 1213
- Аннотация-заголовок 1214

Аннотации отрисовщиков имеют несколько преимуществ по сравнению с аннотациями области построения.

АННОТАЦИИ

Список аннотаций.

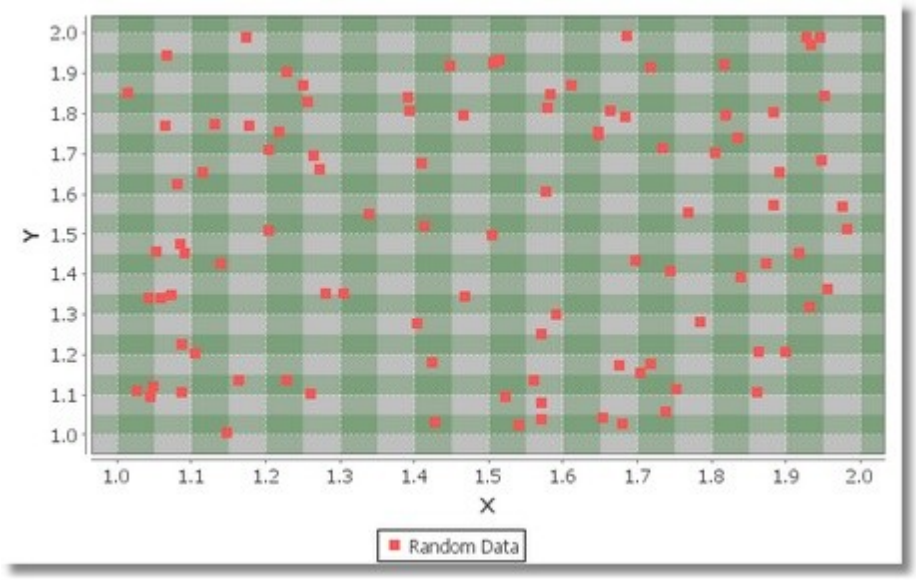
Имя свойства: **annotations**

Тип свойства: **Data Table**

Полосы делений

Координатная область построения может окрашивать различные полосы делений между делениями в области построения осей.

Пример Полос делений:



Полосы делений контролируются двумя свойствами:

ОКРАСКА ПОЛОС ДЕЛЕНИЙ ПАРАМЕТРОВ

Цвет 1279 заливки различных полос между значениями делений по оси параметров. Если значение цвета является пустым, полосы не заливаются цветом.

Имя свойства: **domainTickBandPaint**

Тип свойства: **Data Table**

ОКРАСКА ПОЛОС ДЕЛЕНИЙ ЗНАЧЕНИЙ

Цвет 1279 заливки различных полос между значениями делений по оси значений. Если значение цвета является пустым, полосы не заливаются цветом.

Имя свойства: **rangeTickBandPaint**

Тип свойства: **Data Table**

Свойства осей

Координатная область построения имеет одну ось определений и одну ось измерений. Однако в нее могут быть добавлены дополнительные оси.

Оси области построения могу появляться в верхней, нижней, левой или правой ее части. Положение оси включает два варианта. Применяется вариант, который зависит от ориентации области построения (горизонтальной или вертикальной). Для "вертикальной" области (обычно значение по умолчанию) ось определений будет отображена в верхней или нижней части области построения, а ось измерений будет отображена в ее левой или правой части. Для "горизонтальной" области построения ось определений появится в левой или правой части, а ось измерений -- в верхней или нижней.

СМЕЩЕНИЕ ОСЕЙ

Оси при необходимости могут быть слегка смещены от краев области построения. Данное свойство контролирует смещение между областью построения и осями. См. Прямоугольные вставки 1180.

Имя свойства: **axisOffset**

Тип свойства: **Data Table**

ОСИ ПАРАМЕТРОВ

Таблица, содержащая оси параметров (X-значения) 1165 графика и их позиции.

Существует четыре варианта положения оси параметров:

- **Сверху или Слева**: Сверху, если ориентация области построения является вертикальной, и слева, если ориентация области построения является горизонтальной
- **Сверху или Справа**: Сверху, если ориентация области построения является вертикальной, и справа, если ориентация области построения является горизонтальной
- **Снизу или Слева**: Снизу, если ориентация области построения является вертикальной, и слева, если ориентация области построения является горизонтальной
- **Снизу или Справа**: Снизу, если ориентация области построения является вертикальной, и справа, если ориентация области построения является горизонтальной

Имя свойства: **domainAxes**

Тип свойства: **Data Table**

ОСИ ЗНАЧЕНИЙ

Таблица, содержащая оси значений (Y-значения) 1165 графика и их позиции.

Существуют четыре варианта положения оси значений:

- **Сверху или Слева**: Сверху, если ориентация области построения является горизонтальной, и слева, если ориентация области построения является вертикальной
- **Сверху или Справа**: Сверху, если ориентация области построения является горизонтальной, и справа, если ориентация области построения является вертикальной
- **Снизу или Слева**: Снизу, если ориентация области построения является горизонтальной, и слева, если ориентация области построения является вертикальной
- **Снизу или Справа**: Снизу, если ориентация области построения является горизонтальной, и справа, если ориентация области построения является вертикальной

Имя свойства: **rangeAxes**

Тип свойства: **Data Table**

ВИДИМОСТЬ БАЗОВОЙ ЛИНИИ ПАРАМЕТРОВ

Флажок, контролирующий, будет ли отображаться нулевая линия напротив оси параметров.

Нулевая линия на оси определений является базовой линией напротив оси определений у нулевого значения.

Имя свойства: **domainZeroBaselineVisible**

Тип свойства: **Data Table**

ШТРИХ БАЗОВОЙ ЛИНИИ ПАРАМЕТРОВ

Штрих 1282 нулевой линии напротив оси параметров.

Имя свойства: **domainZeroBaselineStroke**

Тип свойства: **Data Table**

ОКРАСКА БАЗОВОЙ ЛИНИИ ПАРАМЕТРОВ

Цвет 1279 нулевой линии напротив оси параметров.

Имя свойства: **domainZeroBaselinePaint**

Тип свойства: **Data Table**

ВИДИМОСТЬ БАЗОВОЙ ЛИНИИ ЗНАЧЕНИЙ

Флажок, контролирующий, будет ли отображаться нулевая линия для оси значений.

Нулевая линия по оси значений является базовой линией для оси значений у нулевого значения.

Имя свойства: **rangeZeroBaselineVisible**

Тип свойства: **Data Table**

ШТРИХ БАЗОВОЙ ЛИНИИ ЗНАЧЕНИЙ

Штрих 1282, используемый для отображения нулевой линии для оси значений.

Имя свойства: **rangeZeroBaselineStroke**

Тип свойства: **Data Table**

ОКРАСКА БАЗОВОЙ ЛИНИИ ЗНАЧЕНИЙ

Цвет 1279, используемый для отображения нулевой линии для оси значений.

Имя свойства: **rangeZeroBaselinePaint**

Тип свойства: **Data Table**

Фиксированные размеры оси

Ширина и высота осей обычно определяются автоматически для использования необходимого пространства, т.е. ни больше, ни меньше. В некоторых случаях вам может понадобиться изменить данное поведение и задать определенное пространство для расположения осей. Это послужит упрощению выравнивания содержимого множества графиков.

Таблица фиксированного пространства осей содержит четыре значения (Сверху, Снизу, Слева, Справа), которые определяют пространство для расположения осей в верхней, нижней, левой и правой частях области построения соответственно. Так как область построения может включать в себя несколько осей, данные знаечния сопоставляют требования к пространству всех осей.

ФИКСИРОВАННОЕ ПРОСТРАНСТВО ОСИ ПАРАМЕТРОВ

Задает фиксированное пространство для отображения оси параметров.

Имя свойства: **fixedDomainAxisSpace**

Тип свойства: **Data Table**

ФИКСИРОВАННОЕ ПРОСТРАНСТВО ОСИ ЗНАЧЕНИЙ

Задает фиксированное пространство для отображения оси значений.

Имя свойства: **fixedRangeAxisSpace**

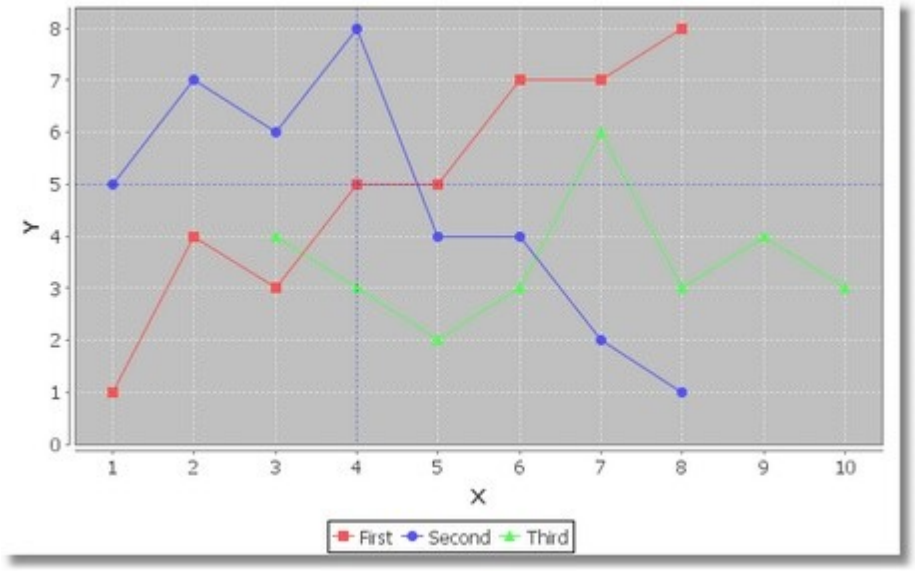
Тип свойства: **Data Table**

Перекрестия

Координатная область построения поддерживает перекрестия напротив основных осей параметров и значений.

Перекрестия устанавливаются нажатием мыши на графике в работающем виджете.

Пример графика с перекрестиями по оси параметров и оси значений.



ВИДИМОСТЬ ПЕРЕКРЕСТИЯ НА ОСИ ПАРАМЕТРОВ

Флажок, контролирующий видимость перекрестия на оси параметров.

Имя свойства: **domainCrosshairVisible**

Тип свойства: **Boolean**

ЗНАЧЕНИЕ ПЕРЕКРЕСТИЯ НА ОСИ ЗНАЧЕНИЙ

Значение для точки перекрестия на оси значений.

Имя свойства: **domainCrosshairValue**

Тип свойства: **Float**

ШТРИХ ПЕРЕКРЕСТИЯ НА ОСИ ПАРАМЕТРОВ

Штрих 1282 перекрестия на оси параметров, если оно видимое.

Имя свойства: **domainCrosshairStroke**

Тип свойства: **Data Table**

ОКРАСКА ПЕРЕКРЕСТИЯ НА ОСИ ПАРАМЕТРОВ

Цвет 1279 перекрестия на оси параметров, если оно видимое.

Имя свойства: **domainCrosshairPaint**

Тип свойства: **Data Table**

ЗАХВАТ ДАННЫХ ПЕРЕКРЕСТИЕМ НА ОСИ ПАРАМЕТРОВ

Флажок, контролирующий, будет ли точка перекрестия захватывать ближайшие значения данных в момент его образования нажатием мыши на графике.

Имя свойства: **domainCrosshairLockedOnData**

Тип свойства: **Boolean**

ВИДИМОЕ ПЕРЕКРЕСТИЕ НА ОСИ ЗНАЧЕНИЙ

Флажок, контролирующий видимость перекрестия на оси значений.

Имя свойства: **rangeCrosshairVisible**

Тип свойства: **Boolean**

ЗНАЧЕНИЕ ПЕРЕКРЕСТИЯ НА ОСИ ЗНАЧЕНИЙ

Значение точки перекрестия на оси значений.

Имя свойства: **rangeCrosshairValue**

Тип свойства: **Float**

ШТРИХ ПЕРЕКРЕСТИЯ НА ОСИ ЗНАЧЕНИЙ

Штрих 1282, используемый для отображения перекрестия на оси значений, если оно видимое.

Имя свойства: **rangeCrosshairStroke**

Тип свойства: **Data Table**

ОКРАСКА ПЕРЕКРЕСТИЯ НА ОСИ ЗНАЧЕНИЙ

Цвет 1279, используемый для отображения перекрестия на оси значений, если оно видимое.

Имя свойства: **rangeCrosshairPaint**

Тип свойства: **Data Table**

ЗАХВАТ ДАННЫХ ПЕРЕКРЕСТИЕМ НА ОСИ ЗНАЧЕНИЙ

Флажок, контролирующий, будет ли перекрестие на оси значений захватывать ближайшее значение данных, когда оно устанавливается по нажатию мыши на графике.

Имя свойства: **rangeCrosshairLockedOnData**

Тип свойства: **Boolean**

Линии сетки

Координатная область построения поддерживает отображение линий сетки напротив основных осей определений и измерений. Для каждой оси существует флажок, контролирующий видимость линий сетки. Для видимых линий можно задать стиль линии (Штрих) и цвет (Цвет).

ВИДИМЫЕ ЛИНИИ СЕТКИ ПО ОСИ ПАРАМЕТРОВ

Флажок, контролирующий видимость линий сетки напротив оси параметров.

Имя свойства: **domainGridlinesVisible**

Тип свойства: **Boolean**

ШТРИХ ЛИНИЙ СЕТКИ ПО ОСИ ПАРАМЕТРОВ

Штрих 1282 линий сетки по оси параметров.

Имя свойства: **domainGridlineStroke**

Тип свойства: **Data Table**

ЦВЕТ ЛИНИЙ СЕТКИ ПО ОСИ ПАРАМЕТРОВ

Цвет 1279 линий сетки по оси параметров.

Имя свойства: **domainGridlinePaint**

Тип свойства: **Data Table**

ВИДИМОСТЬ ЛИНИЙ СЕТКИ ПО ОСИ ЗНАЧЕНИЙ

Флажок, контролирующий видимость линий сетки по оси значений.

Имя свойства: **rangeGridlinesVisible**

Тип свойства: **Boolean**

ШТРИХ ЛИНИЙ СЕТКИ ПО ОСИ ЗНАЧЕНИЙ

Штрих 1282 линий сетки по оси значений.

Имя свойства: **rangeGridlineStroke**

Тип свойства: **Data Table**

ОКРАСКА ЛИНИЙ СЕТКИ ПО ОСИ ЗНАЧЕНИЙ

Цвет 1279 линий сетки по оси значений.

Имя свойства: **rangeGridlinePaint**

Тип свойства: **Data Table**

ВИДИМОСТЬ МАЛЫХ ЛИНИЙ СЕТКИ ПО ОСИ ПАРАМЕТРОВ

Флажок, контролирующий видимость линий сетки для значений малых делений основной оси параметров.

Имя свойства: **domainMinorGridlinesVisible**

Тип свойства: **Boolean**

ШТРИХ МАЛЫХ ЛИНИЙ СЕТКИ ПО ОСИ ПАРАМЕТРОВ

Штрих 1282 малых линий сетки по оси параметров.

Имя свойства: **domainMinorGridlineStroke**

Тип свойства: **Data Table**

ЦВЕТ МАЛЫХ ЛИНИЙ СЕТКИ ПО ОСИ ПАРАМЕТРОВ

Цвет 1279 малых линий сетки по оси параметров.

Имя свойства: **domainMinorGridlinePaint**

Тип свойства: **Data Table**

ВИДИМОСТЬ МАЛЫХ ЛИНИЙ СЕТКИ ПО ОСИ ЗНАЧЕНИЙ

Флажок, контролирующий видимость линий сетки для значений малых делений основной оси значений.

Имя свойства: **rangeMinorGridlinesVisible**

Тип свойства: **Boolean**

ШТРИХ МАЛЫХ ЛИНИЙ СЕТКИ ПО ОСИ ЗНАЧЕНИЙ

Штрих 1282 малых линий сетки по оси значений.

Имя свойства: **rangeMinorGridlineStroke**

Тип свойства: **Data Table**

ОКРАСКА МАЛЫХ ЛИНИЙ СЕТКИ ПО ОСИ ЗНАЧЕНИЙ

Цвет 1279 малых линий сетки по оси значений.

Имя свойства: **rangeMinorGridlinePaint**

Тип свойства: **Data Table**

Другие свойства

ОРИЕНТАЦИЯ

Ориентация области построения (Вертикальная или Горизонтальная). По умолчанию установлена на Вертикальную.

Имя свойства: **orientation**

Тип свойства: **String**

ПРОРКУТКА ПО ОСИ ПАРАМЕТРОВ

Флажок, контролирующий, может ли осуществляться прокрутка по оси/осям параметров.

Имя свойства: **domainPannable**

Тип свойства: **Boolean**

ПРОКРУТКА ПО ОСИ ЗНАЧЕНИЙ

Флажок, контролирующий, может ли осуществляться прокрутка по оси/осям значений.

Имя свойства: **rangePannable**

Тип свойства: **Boolean**

ФИКСИРОВАННЫЕ ЭЛЕМЕНТЫ ЛЕГЕНДЫ

Подборка элементов легенды[1175] для области построения, используемые для замещения автоматически сформированного набора элементов легенды, если он не является пустым.

Имя свойства: **fixedLegendItems**

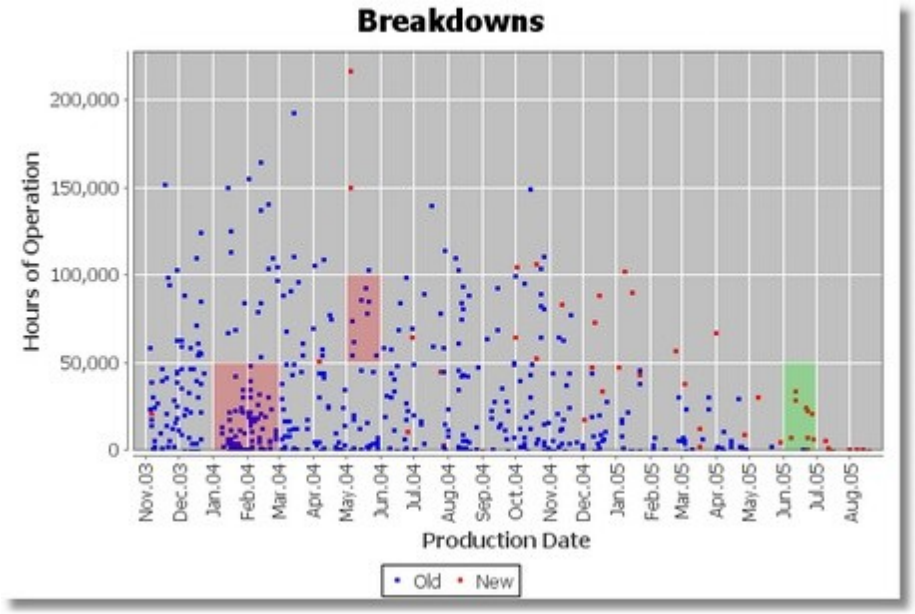
Тип свойства: **Data Table**

13.4.11.9.2.1 Блочная аннотация

Аннотация, выделяющая прямоугольную часть в области данных. Аннотация будет отображаться только тогда, когда она попадает в пределы границ осей области построения.

Данную аннотацию можно использовать с Осями дат. Для этого необходимо задать соответствующие координаты в миллисекундах с 1 января 1970.

Пример блочной аннотации:



Свойства

Свойство	Имя	Тип	Описание
X	x	плавающее	Нижняя x-координата блока.
Y	y	плавающее	Нижняя y-координата блока.
Ширина	w	плавающее	Ширина блока.
Выоста	h	плавающее	Высота блока.
Цвет заливки	fillPaint	таблица данных	Цвет[1279], используемый для заливки блока.
Штрих	stroke	таблица данных	Штрих[1282] контура блока.

Цвет контура	outlinePaint	таблица данных	Цвет[1279] контура блока.
Текст всплывающей подсказки	toolTipText	строка	Текст всплывающей подсказки для аннотации.

13.4.11.9.2.2 Аннотация-изображение в координатах графика

Аннотация, которая позволяет отображать рисунок в Координатной области построения в прямоугольнике, заданном в координатах данных. Возможным применением данной аннотации является поддержка построения базовых географических графиков, отображая карты в качестве фона графиков.

Свойства

Свойство	Имя	Тип	Описание
X	x	плавающее	Нижняя x-координата блока.
Y	y	плавающее	Нижняя y-координата блока.
Ширина	w	плавающее	Ширина изображения, в единицах данных графика.
Высота	h	плавающее	Высота изображения, в единицах данных графика.
Изображение	image	блок данных	Отображаемое изображение.
Включить в границы данных	includeInDataBounds	логическое	Флажок, указывающий, будет ли аннотация включена в диапазон данных для области построения/отрисовщика.
Текст всплывающей подсказки	toolTipText	строка	Текст всплывающей подсказки для аннотации.

13.4.11.9.2.3 Аннотация-изображение

Аннотация, позволяющая отображать рисунок в произвольном местоположении (x, y).

Свойства

Свойство	Имя	Тип	Описание
X	x	плавающее	Нижняя x-координата блока.
Y	y	плавающее	Нижняя y-координата блока.

Точка привязки	anchor	строка	Точка привязки изображения, которая выравнивается по позиции (x, y) на графике во время отображения аннотации.
Изображение	image	блок данных	Отображаемое изображение.
Текст всплывающей подсказки	toolTipText	строка	Текст всплывающей подсказки для аннотации.

13.4.11.9.2.4 Генератор меток элементов

Координатный генератор меток элементов отвечает за создание текстовых строк, которые будут использоваться в качестве меток элементов в координатном графике.

Свойства

Свойство	Имя	Тип	Описание
Форматная строка	formatString	строка	Форматная строка, содержащая следующие символы: • {0} - имя серии • {1} - строковое представление X-значения • {2} - строковое представление Y-значения
Тип форматера X Значений	xFormatType	целое	Форматер, используемый для X-значений: **Числовой форматер** или **Форматер дат**.
Формат X значений	xFormat	строка	Шаблон формата X-значений: шаблон числового формата 2147 для Числового форматера или шаблон формата Даты/времени 2146 для Форматера дат.
Тип форматера Y Значений	yFormatType	целое	Форматер, используемый для Y-значений: **Числовой форматер** или **Форматер дат**.
Формат Y значений	yFormat	строка	Шаблон формата Y-значений: шаблон числового формата 2147 для Числового форматера или шаблон формата Даты/времени 2146 для Форматера дат.

13.4.11.9.2.5 Аннотация-легенда

Аннотация, которая может помещать Легенду 1173 в график.

Пример аннотации легенды:

Свойства

Свойство	Имя	Тип	Описание
X	x	плавающее	Координата X заголовка **в диапазоне от 0.0 до 1.0**.
Y	y	плавающее	Координата Y заголовка **в диапазоне от 0.0 до 1.0**.
Максимальная ширина	maxWidth	плавающее	Маскимальная ширина аннотации (задается в процентах от ширины области построения, т.е. 0.05 составляет 5%).
Максимальная высота	maxHeight	плавающее	Максимальная высота аннотации (задается в процентах от ширины области построения, т.е. 0.05 составляет 5%).
Легенда	legend	таблица данных	Отображаемая Легенда 1173.
Точка привязки	anchor	строка	Точка привязки заголовка, который выравнивается по позиции (x, y) на графике во время отображения аннотации.
Текст всплывающей подсказки	toolTipText	строка	Текст всплывающей подсказки для аннотации.

13.4.11.9.2.6 Линейная аннотация

Простая аннотация, которая чертит линию между начальной точкой (x0, y0) и конечной (x1, y1).

Свойства

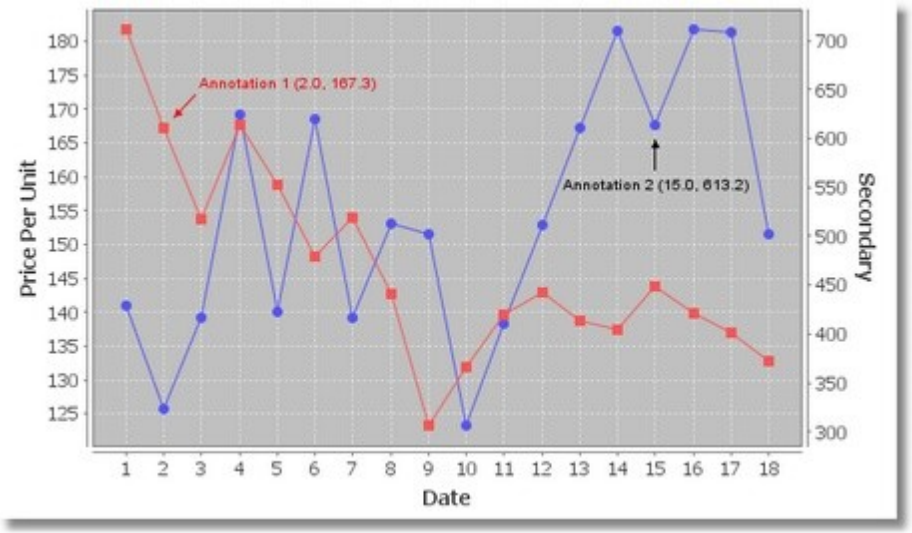
Свойство	Имя	Тип	Описание

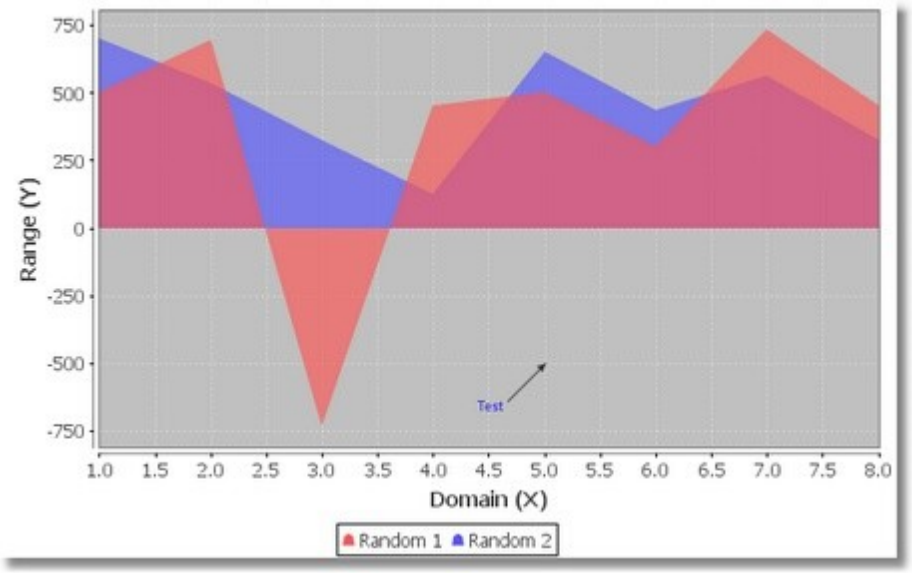
X	x	плавающее	Нижняя x-координата линии.
Y	y	плавающее	Нижняя y-координата линии.
Ширина	w	плавающее	Длина линии вдоль оси X, т.е. X1 - X0.
Высота	h	плавающее	Длина линии вдоль оси Y, т.е. Y1 - Y0.
Цвет	paint	таблица данных	Цвет[1279] линии.
Штрих	stroke	таблица данных	Штрих[1282] линии.
Текст всплывающей подсказки	toolTipText	строка	Текст всплывающей подсказки для аннотации.

13.4.11.9.2.7 Аннотация-указатель

Аннотация, отображающая стрелку, которая указывает на определенную позицию (x, y). У окончания стрелки может находиться метка.

Примеры аннотаций с указателями:





Свойства

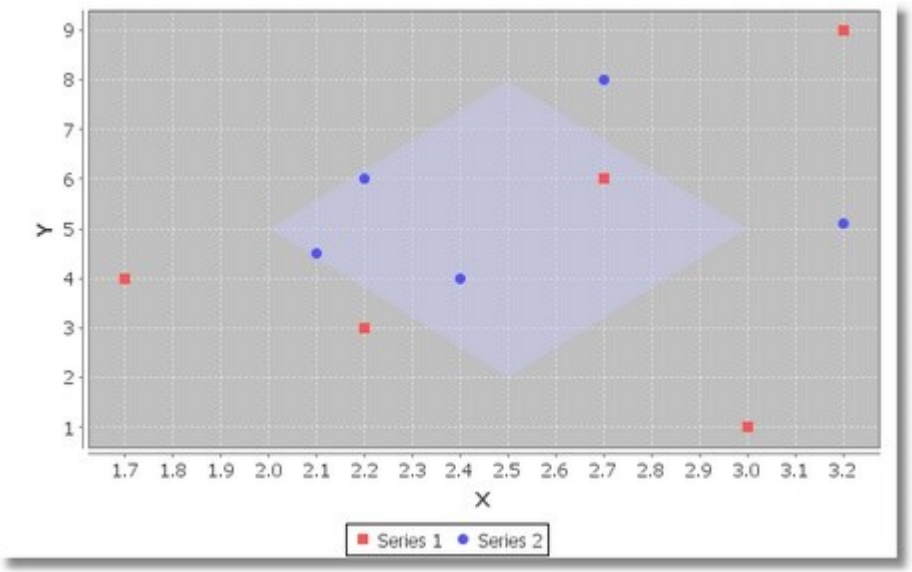
Свойство	Имя	Тип	Описание
X	x	плавающее	X-значение для аннотации.
Y	y	плавающее	Y-значение для аннотации.
Текст	text	строка	Текст, отображаемый аннотацией.
Шрифт	font	таблица данных	Шрифт[1278] текста.
Точка привязки текста	textAnchor	строка	Точка привязки текста. Выравнивается по точке **Категория, Значение** на графике.
Цвет	paint	таблица данных	Цвет[1279] текста.
Цвет фона	backgroundPaint	таблица данных	Цвет фона метки (или null, в этом случае фон метки будет бесцветным).
Точка привязки вращения	rorationAnchor	строка	Точка привязки текста, вокруг которой выполняется вращение.
Угол вращения	rorationAngle	плавающее	Угол вращения (в градусах).
Видимость контура	outlineVisible	логическое	Флажок, контролирующий видимость контура метки.
Цвет контура	outlinePaint	таблица данных	Цвет[1279] контура метки.
Штрих контура	outlinePaint	таблица данных	Штрих[1282] контура метки.

Угол	angle	плавающее	Угол указателя (в градусах).
Длина стрелки	arrowLength	плавающее	Длина острия стрелки.
Цвет стрелки	arrowPaint	таблица данных	Цвет[1279] стрелки.
Штрих стрелки	arrowStroke	таблица данных	Штрих[1282] стрелки.
Ширина стрелки	arrowWidth	плавающее	Ширина острия стрелки.
Радиус основания	baseRadius	плавающее	Расстояние между точкой привязки и основанием стрелки. Разница между Радиусом основания и Радиусом окончания стрелки составляет длину стрелки.
Смещение метки	labelOffset	плавающее	Смещение от основания стрелки к метке.
Радиус окончания стрелки	tipRadius	плавающее	Расстояние между точкой привязки и окончанием стрелки. Так как окончание стрелки указывает на точку привязки, данное значение должно быть меньше **Радиуса основания**.
Текст всплывающей подсказки	toolTipText	строка	Текст всплывающей подсказки для аннотации.

13.4.11.9.2.8 Аннотация-многоугольник

Простая аннотация, которая отображает многоугольник на координатной области построения. Координаты многоугольника определяются в пространстве данных (т.е. в системе координат, которую образуют оси области построения).

Пример аннотации-многоугольника:



Свойства

Свойство	Имя	Тип	Описание
Цвет заливки	fillPaint	таблица	Цвет[1279] заливки многоугольника.

		данных	
Штрих	stroke	таблица данных	Штрих 1282 контура многоугольника.
Цвет контура	outlinePaint	таблица данных	Цвет 1279 контура многоугольника.
Многоугольник	polygon	таблица данных	Список точек (X, Y), определяющих вершины многоугольника.
Текст всплывающей подсказки	toolTipText	строка	Текст всплывающей подсказки для аннотации.

13.4.11.9.2.9 Аннотация-форма

Простая аннотация, которая отображает форму в Координатной области построения. Координаты формы задаются в "пространстве данных" (т.е. в системе координат, которую образуют оси области построения).

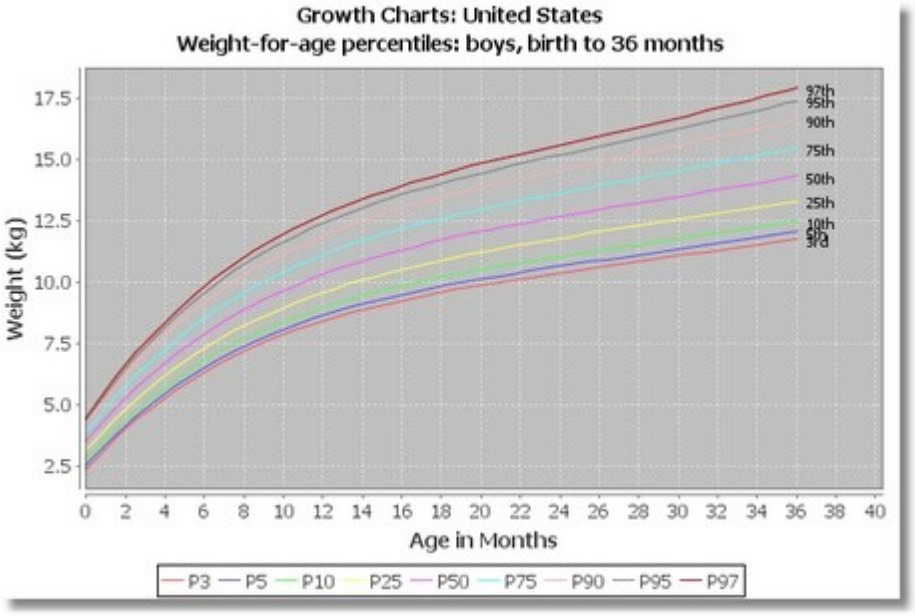
Свойства

Свойство	Имя	Тип	Описание
Штрих	stroke	таблица данных	Штрих 1282, используемый для отображения контура формы.
Фигура	shape	таблица данных	Отображаемая Форма 1283. Координаты вершин задаются в "пространстве данных".
Цвет заливки	fillPaint	таблица данных	Цвет 1279 заливки формы.
Текст всплывающей подсказки	toolTipText	строка	Текст всплывающей подсказки для аннотации.

13.4.11.9.2.10 Текстовая аннотация

Текстовая аннотация отображает на графике небольшую текстовую метку в позиции (x, y).

Пример текстовой аннотации:



Свойства

Свойство	Имя	Тип	Описание
X	x	плавающее	X-значение для аннотации.
Y	y	плавающее	Y-значение для аннотации.
Цвет	paint	таблица данных	Цвет[1279] текста.
Цвет фона	backgroundPaint	таблица данных	Цвет[1279] фона аннотации.
Цвет контура	outlinePaint	таблица данных	Цвет[1279] контура аннотации.
Текст	text	строка	Текст, отображаемый аннотацией.
Шрифт	font	таблица данных	Шрифт[1278] текста.
Точка привязки текста	textAnchor	строка	Точка привязки текста. Выравнивается по точке (X, Y) на графике.
Точка привязки вращения	rorationAnchor	строка	Точка привязки текста, вокруг которой выполняется вращение.
Угол вращения	rorationAngle	плавающее	Угол вращения (в градусах).

Видимость контура	outlineVisible	таблица данных	Флажок, контролирующий, будет ли отображаться контур для аннотации.
Штрих контура	outlineStroke	таблица данных	Штрих 1282 контура аннотации.
Текст всплывающей подсказки	toolTipText	строка	Текст всплывающей подсказки для аннотации.

13.4.11.9.2.11 Аннотация-заголовок

Аннотация, которая помещает Заголовок 1181 на график.

Свойства

Свойство	Имя	Тип	Описание
X	x	плавающее	Координата X заголовка **в диапазоне от 0.0 до 1.0**.
Y	y	плавающее	Координата Y заголовка **в диапазоне от 0.0 до 1.0**.
Маскимальная ширина	maxWidth	плавающее	Максимальная ширина аннотации (задается в процентах от ширины области построения, т.е. 0.05 составляет 5%).
Максимальная высота	maxHeight	плавающее	Максимальная высота аннотации (задается в процентах от ширины области построения, т.е. 0.05 составляет 5%).
Заголовок	title	таблица данных	Отображаемый Заголовок 1181.
Точка привязки	anchor	строка	Точка привязки заголовка, который выравнивается на графике по позиции (x, y) во время отображения аннотации.
Текст всплывающей подсказки	toolTipText	строка	Текст всплывающей подсказки для аннотации.

13.4.11.9.2.12 Генератор всплывающих подсказок

Координатный генератор всплывающих подсказок отвечает за создание текстовых строк, которые будут использоваться в качестве всплывающих подсказок на координатном графике.

Свойства

Свойство	Имя	Тип	Описание
Форматная строка	formatString	строка	Форматная строка, содержащая следующие символы: • {0} - имя серии • {1} - строковое представление X-значения • {2} - строковое представление Y-значения

Тип форматера X значений	xFormatType	целое	Форматер, используемый для X-значений: **Числовой форматер** или **Форматер дат**.
Формат X значений	xFormat	строка	Шаблон формата X-значений: шаблон числового формата 2147 для Числового форматера или шаблон формата Даты/времени 2146 для Форматера дат.
Тип форматера Y значений	yFormatType	целое	Форматер, используемый для Y-значений: **Числовой форматер** или **Форматер дат**.
Формат Y значений	yFormat	строка	Шаблон формата Y-значений: шаблон числового формата 2147 для Числового форматера или шаблон формата Даты/времени 2146 для Форматера дат.

13.4.11.9.3 Секторная область построения

Данная область построения используется Круговым 1156 и Кольцевым 1158 графиками.

Общие свойства

НАЧАЛЬНЫЙ УГОЛ

Угол (в градусах), с которого начинается первый сектор. Ноль расположен в позиции 3 часов, и по мере увеличения угла, сектор растягивается против часовой стрелки по окружности графика (так, что 90 градусов приходится на позицию 12 часов).

Имя свойства: **startAngle**

Тип свойства: **Float**

НАПРАВЛЕНИЕ

Направление секторов графика: **По часовой стрелке** (по умолчанию) или **Против часовой** стрелки.

Имя свойства: **direction**

Тип свойства: **String**

ВНУТРЕННИЙ ПРОМЕЖУТОК

Интервал во внутренней части секторного графика (область, где отображаются метки) в процентах от ширины и высоты области построения. Значением по умолчанию является 0.08 (8%).

Имя свойства: **interiorGap**

Тип свойства: **Float**

КРУГ

Флажок, контролирующий, будет график круглой или эллиптической формы.

Имя свойства: **circular**

Тип свойства: **Boolean**

Работа с пустыми и нулевыми значениями

Массив данных может содержать пустые, нулевые или отрицательные значения, которые не могут быть отображены на секторном графике. Для их отображения в секторном графике применяется специальная обработка. Если в массиве данных обнаружены пустые или нулевые значения, Круговая диаграмма по умолчанию помещает метку в позицию, где отображался бы сектор графика, если бы он имел положительные значения, и также добавляет элемент в легенду графика. Если вы хотите, чтобы нулевые и пустые значения игнорировались, используйте следующие свойства:

ИГНОРИРОВАТЬ ЗНАЧЕНИЯ NULL

Флажок, контролирующий, будут ли игнорироваться пустые значения в массиве данных.

Имя свойства: **ignoreNullValues**

Тип свойства: **Boolean**

ИГНОРИРОВАТЬ НУЛЕВЫЕ ЗНАЧЕНИЯ

Флажок, контролирующий, будут ли игнорироваться нулевые значения в массиве данных.

Имя свойства: **ignoreZeroValues**

Тип свойства: **Boolean**

Эффект тени

Область построения отобразит эффект "тени", контролируемый несколькими свойствами:

ОКРАСКА ТЕНИ

Цвет 1279, используемый для отображения эффекта "тени". Если отключено, тень не отображается.

Имя свойства: **shadowPaint**

Тип свойства: **Data Table**

СМЕЩЕНИЕ ТЕНИ ПО X

Смещение эффекта тени по оси X.

Имя свойства: **shadowXOffset**

Тип свойства: **Float**

СМЕЩЕНИЕ ТЕНИ ПО Y

Смещение эффекта тени по оси Y.

Имя свойства: **shadowXOffset**

Тип свойства: **Float**

Метки и всплывающие подсказки секторов

ГЕНЕРАТОР МЕТОК

Метки секторов формируются согласно следующим свойствам:

Свойство	Имя	Тип	Описание
Отключено	isNull	логическое	Флажок, включающий/отключающий метки элементов.
Формат меток	labelFormat	строка	Форматная строка, содержащая следующие символы: • {0} - имя серии • {1} - строковое представление значения • {2} - строковое представление значения в процентах • {3} - сумма всех значений
Числовой формат	numberFormat	строка	Шаблон числового формата 2147.
Процентный формат	percentFormat	строка	Шаблон числового формата 2147, используемый для процентных значений.

Имя свойства: **labelGenerator**

Тип свойства: **Data Table**

ГЕНЕРАТОР ПОДСКАЗОК

Всплывающие подсказки секторов формируются согласно следующим свойствам:

Свойство	Имя	Type	Тип

Отключено	isNull	логическое	Флажок, который включает/отключает всплывающие подсказки.
Формат меток	labelFormat	строка	Форматная строка, содержащая следующие символы: • {0} - имя серии • {1} - строковое представление значения • {2} - строковое представление значения в процентах • {3} - сумма всех значений
Числовой формат	numberFormat	строка	Шаблон числового формата 2147.
Процентный формат	percentFormat	строка	Шаблон числового формата 2147, используемый для процентных значений.

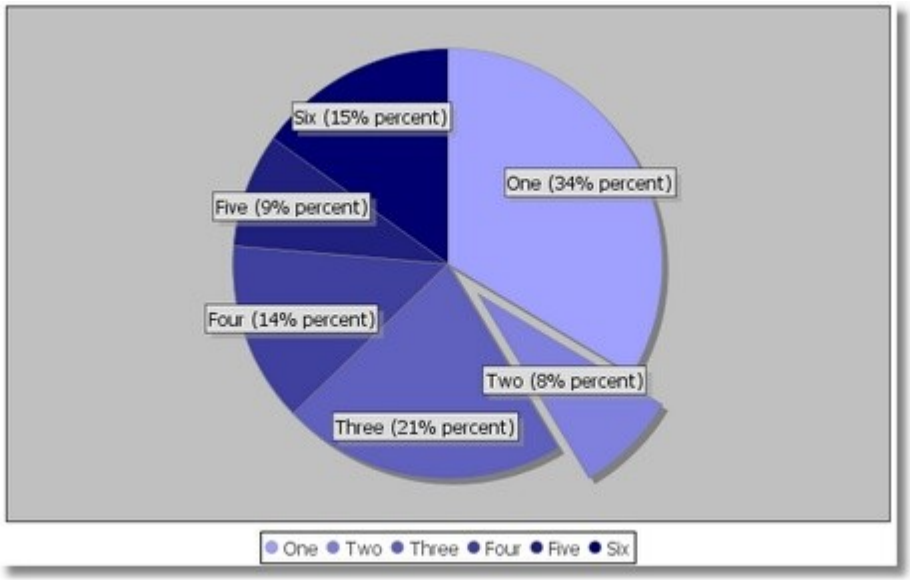
Имя свойства: **toolTipGenerator**

Тип свойства: **Data Table**

Отделяемые секторы

Секторная область построения поддерживает отображение "отделяемых" секторов, когда сектор графика смещается от центра для выделения.

Пример отделяемого сектора:



Чтобы предоставить достаточное пространство для секторов, которые смещаются от центра графика, радиус основного сектора убавляется, поэтому график с отделяемыми секторами будет казаться меньше обычного секторного графика.

ВЫДЕЛЕНИЕ СЕКЦИЙ

Таблица, контролирующая "отделяемость" секторов:

Свойство	Имя	Тип	Описание
Серия	series	строка	Имя отделяемой серии данных.
Проценты	explodePercents	плавающее	Величина, на которую смещается сектор графика, выраженная в процентах от радиуса сектора.

Имя свойства: **explodedPercents**

Тип свойства: **Data Table**

Цвета секторов

Цвет, используемый для заливки каждого сектора на графике, по умолчанию устанавливается автоматически.

БАЗОВАЯ ОКРАСКА СЕКЦИЙ

Цвет[1279] сектора по умолчанию, используемый, если для серий нет заданных пользовательских настроек.

Имя свойства: **baseSectionPaint**

Тип свойства: **Data Table**

АВТОЗАПОЛНЕНИЕ ОКРАСОК СЕКЦИЙ

Флажок, контролирующий, будут ли использованы различные, определенные заранее цвета для заливки секторов.

Имя свойства: **autoPopulateSectionPaints**

Тип свойства: **Boolean**

ОКРАСКА СЕКЦИЙ

Таблица, определяющая цвет для каждого сектора.

Свойство	Имя	Тип	Описание
Серия	series	строка	Имя серии данных.
Цвет	paint	таблица данных	Цвет[1279] заливки серии (сектора).

Имя свойства: **sectionPaints**

Тип свойства: **Data Table**

Контуры секторов

Контуры секторов отображаются по умолчанию в виде тонкой серой линии. Секторная область построения предлагает следующие опции:

- Полностью отключить отображение контуров;
- Изменить контуры всех секторов, отредактировав значение по умолчанию;
- Контролировать контур определенного сектора в отдельности.

ВИДИМОСТЬ ОКАНТОВКИ СЕКЦИЙ

Флажок, контролирующий, будут ли отображаться контуры секторов.

Имя свойства: **sectionOutlinesVisible**

Тип свойства: **Boolean**

БАЗОВАЯ ОКРАСКА ОКАНТОВКИ СЕКЦИЙ

Цвет[1279] контура по умолчанию, используемый, если для серии нет заданных пользовательских настроек.

Имя свойства: **baseSectionOutlinePaint**

Тип свойства: **Data Table**

АВТОЗАПОЛНЕНИЕ ОКРАСОК ОКАНТОВКИ СЕКЦИЙ

Флажок, контролирующий, будут ли использованы различные, определенные заранее цвета контуров секторов.

Имя свойства: **autoPopulateSectionOutlinePaints**

Тип свойства: **Boolean**

ОКРАСКА ОКАНТОВКИ СЕКЦИЙ

Таблица, определяющая цвет контура для каждого сектора.

Свойство	Имя	Тип	Описание
Серия	series	строка	Имя серии данных.
Цвет	paint	таблица данных	Цвет[1279] контура для серии (сектора).

Имя свойства: **sectionOutlinePaints**

Тип свойства: **Data Table**

БАЗОВЫЙ ШТРИХ ОКАНТОВКИ СЕКЦИЙ

Штрих[1282] контура секторов по умолчанию, используемый, если для серии нет заданных пользовательских настроек.

Имя свойства: **baseSectionOutlineStroke**

Тип свойства: **Data Table**

АВТОЗАПОЛНЕНИЕ ШТРИХОВ ОКАНТОВКИ СЕКЦИЙ

Флажок, контролирующий, будут ли использованы различные, определенные заранее цвета штриха контуров секторов.

Имя свойства: **autoPopulateSectionOutlineStrokes**

Тип свойства: **Boolean**

ШТРИХИ ОКАНТОВОК СЕКЦИЙ

Таблица, определяющая штрих контура для каждого сектора.

Свойство	Имя	Тип	Описание
Серия	series	строка	Имя серии.
Штрих	stroke	таблица данных	Штрих[1282] контура для серии (сектора).

Имя свойства: **sectionOutlineStrokes**

Тип свойства: **Data Table**

Метки секторов

ШРИФТ МЕТОК

Шрифт[1278], используемый для отображения меток секторов.

Имя свойства: **labelFont**

Тип свойства: **Data Table**

ОКРАСКА МЕТОК

Цвет[1279], используемый для отображения меток секторов.

Имя свойства: **labelPaint**

Тип свойства: **Data Table**

ОКРАСКА ФОНА МЕТОК

Цвет[1279] заливки блоков меток. Если отключено, блоки меток будут прозрачными (будет просматриваться цвет фона графика).

Имя свойства: **labelBackgroundPaint**

Тип свойства: **Data Table**

ОКРАСКА ОКАНТОВКИ МЕТОК

Цвет[1279] контура вокруг меток секторов. Если отключено, блоки меток будут отображаться без контура.

Имя свойства: **labelOutlinePaint**

Тип свойства: **Data Table**

ШТРИХ ОКАНТОВКИ МЕТОК

Штрих[1282] контура вокруг меток секторов. Если отключено, блоки меток будут отображаться без контура.

Имя свойства: **labelOutlineStroke**

Тип свойства: **Data Table**

ОКРАСКА ТЕНИ МЕТОК

Цвет[1279] тени под метками секторов. Если отключено, тень не отображается.

Имя свойства: **labelShadowPaint**

Тип свойства: **Data Table**

ОТСТУПЫ МЕТОК

Отступ для меток, т.е. пустое пространство вокруг текста и внутри контура. См. Прямоугольные вставки[1180].

Имя свойства: **labelPadding**

Тип свойства: **Data Table**

Соединения с метками

Для регулярных (непростых, см. далее) меток секторов отображается линия, соединяющая сектор графика с соответствующей ему меткой. Данные соединения имеют следующие свойства:

ВИДИМОСТЬ ЛИНИЙ МЕТОК

Флажок, контролирующий, будут ли отображаться линии соединения с метками.

Имя свойства: **labelLinksVisible**

Тип свойства: **Boolean**

СТИЛЬ ЛИНИЙ МЕТОК

Стиль соединений с метками: **Стандартный, Квадратический** или **Кубический**

Имя свойства: **labelLinkStyle**

Тип свойства: **String**

ГРАНИЦА ЛИНИЙ МЕТОК

Линия соединений с метками имеет сгиб или "локоть" в точке, расположенной немного за пределами секторного графика. Расстояние до точки от края графика выражается в процентах от радиуса сектора и является Границей соединений с метками. Значением по умолчанию является 0.025 (два с половиной процента).

Имя свойства: **labelLinkMargin**

Тип свойства: **Float**

ОКРАСКА ЛИНИЙ МЕТОК

Цвет[1279] линий соединения частей секторов с соответствующими метками.

Имя свойства: **labelLinkPaint**

Тип свойства: **Data Table**

ШТРИХ ЛИНИЙ МЕТОК

Штрих[1282] линий соединения частей секторов с соответствующими метками.

Имя свойства: **labelLinkStroke**

Тип свойства: **Data Table**

ГРАНИЦА ЛИНИЙ МЕТОК

Интервал между краями сектора и областями меток по левому и правому краю сектора в процентах от общей ширины графика. Значением по умолчанию является 0.025 (2.5%).

Имя свойства: **labelGap**

Тип свойства: **Float**

МАКСИМАЛЬНАЯ ШИРИНА МЕТОК

Максимальная ширина метки в процентах от ширины области построения. Значением по умолчанию является 0.14 (четырнадцать процентов).

Имя свойства: **maximumLabelWidth**

Тип свойства: **Float**

Простые метки

Использование "простых" меток позволяет вам быть уверенным в том, что метки будут отображены в центре каждого сектора. Не предпринимается никаких попыток избежать наложения меток в случае, когда несколько маленьких секторов отображаются рядом друг с другом, поэтому используйте простые метки, когда их наложение не будет представлять проблему для вас.

ПРОСТЫЕ МЕТКИ

Флажок, контролирующий, будут ли метки секторов отображаться в "простом" формате.

Имя свойства: **simpleLabels**

Тип свойства: **Boolean**

СМЕЩЕНИЕ ПРОСТЫХ МЕТОК

Вставки[1180], используемые для расчета точек привязки простых меток относительно ограничительного прямоугольника секторной области построения.

Имя свойства: **simpleLabelOffset**

Тип свойства: **Data Table**

Другие свойства

МИНИМАЛЬНЫЙ УГОЛ СЕКТОРА ДЛЯ ОТРИСОВКИ

Минимальный угол дуги для сектора графика.

Имя свойства: **minimumArcAngleToDraw**

Тип свойства: **Float**

ФОРМА ЭЛЕМЕНТОВ ЛЕГЕНДЫ

Форма[1283], отображаемая для каждого элемента легенды.

Имя свойства: **legendItemShape**

Тип свойства: **Data Table**

13.4.11.10 Отрисовщики графиков

Отрисовщик графика отвечает за представление отдельных элементов массива данных в области построения[1185] графика.

Все типы отрисовщиков графика поддерживают следующие общие характеристики:

1) Цвета, стили линий и формы для каждой серии:

a) Окраска
b) Окраска заливки
c) Окраска контуров
d) Штрих
e) Штрих контуров
f) Форма

2) Видимость серий
3) Видимость серий в легенде
4) Метки элементов
a) Видимость
b) Шрифт
c) Цвет
d) Положительные позиции меток элементов
e) Отрицательные позиции меток элементов

Схема действия свойств для серии данных

Многие свойства отрисовщиков должны иметь значение, определенное для каждой серии в массиве, которое предназначено для отрисовщика. Помимо свойств для серии, обычно существует значение свойства по умолчанию ("базовое"), которое будет использоваться в случае, если для определенной серии не задано значение свойства.

НАСТРОЙКА ЦВЕТА СЕРИИ

Отрисовщики отвечают за отображение элементов данных внутри области построения, поэтому они предоставляют свойства для контролирования цветов, которые будут использоваться. Цвета обычно определяются на основе "для серии" и сохраняются в поисковой таблице.

Существует механизм по умолчанию для автоматического заполнения поисковой таблицы цветами по умолчанию. Однако вы можете вручную обновить список в любое время.

Базовые свойства

ТИП ОТРИСОВЩИКА

Тип используемого отрисовщика для графика.

Имя свойства: **rendererType**

Тип свойства: **Integer**

Свойства серии по умолчанию

БАЗОВАЯ ВИДИМОСТЬ СЕРИЙ

Видимость серий по умолчанию.

Имя свойства: **baseSeriesVisible**

Тип свойства: **Boolean**

БАЗОВАЯ ВИДИМОСТЬ СЕРИЙ В ЛЕГЕНДЕ

Видимость легенды по умолчанию для серии.

Имя свойства: **baseSeriesVisibleInLegend**

Тип свойства: **Boolean**

БАЗОВАЯ ОКРАСКА

Цвет[1279] по умолчанию, используемый, если для серии нет задана пользовательская настройка.

Имя свойства: **basePaint**

Тип свойства: **Data Table**

БАЗОВЫЙ ШТРИХ

Штрих[1282] по умолчанию, используемый, если для серии не задана пользовательская настройка.

Имя свойства: **baseStroke**

Тип свойства: **Data Table**

БАЗОВАЯ ФОРМА

Форма[1283], используемая по умолчанию, если для серии не задана пользовательская настройка.

Имя свойства: **baseShape**

Тип свойства: **Data Table**

БАЗОВЫЙ ЦВЕТ ЗАЛИВКИ

Цвет[1279] заливки, используемый по умолчанию, если для серии не задана пользовательская настройка.

Имя свойства: **baseFillPaint**

Тип свойства: **Data Table**

БАЗОВАЯ ОКРАСКА ОКАНТОВКИ

Цвет[1279] контура по умолчанию, используемый, если для серии не задана пользовательская настройка.

Имя свойства: **baseOutlinePaint**

Тип свойства: **Data Table**

БАЗОВАЯ ВИДИМОСТЬ МЕТОК ЭЛЕМЕНТОВ

Видимость меток элементов по умолчанию.

Имя свойства: **baseItemLabelsVisible**

Тип свойства: **Boolean**

БАЗОВЫЙ ШРИФТ МЕТОК ЭЛЕМЕНТОВ

Шрифт[1278] меток элементов по умолчанию, если для серии не задана пользовательская настройка.

Имя свойства: **baseItemLabelFont**

Тип свойства: **Data Table**

БАЗОВАЯ ОКРАСКА МЕТОК ЭЛЕМЕНТОВ

Цвет[1279] меток элементов по умолчанию, если для серии не задана пользовательская настройка.

Имя свойства: **baseItemLabelPaint**

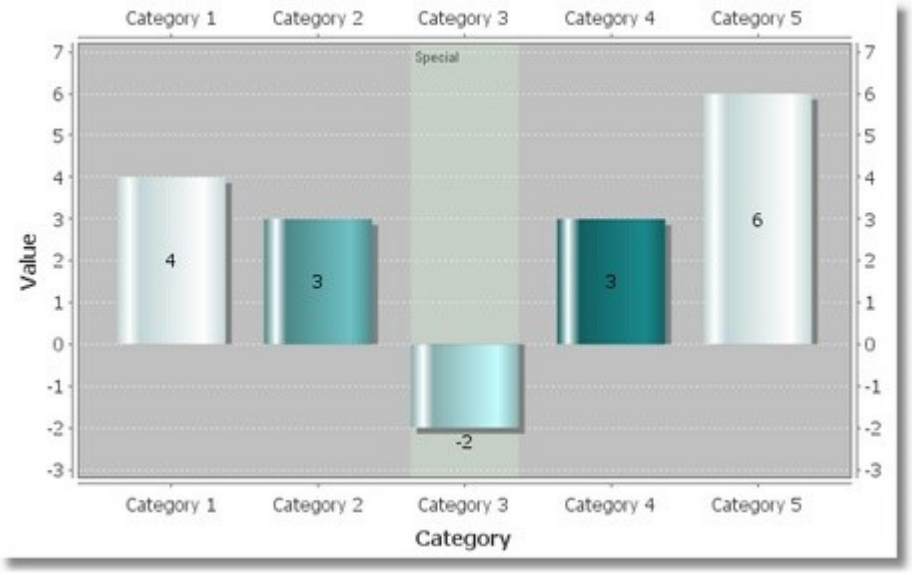
Тип свойства: **Data Table**

БАЗОВОЕ ПОЛОЖЕНИЕ МЕТОК ЭЛЕМЕНТОВ

Данное свойство включает два значения:

- **Положительная позиция меток элементов**[1172]: настройка по умолчанию для положительной позиции меток элементов используется, когда не задана настройка для серии или настройка, заменяющая ее.
- **Отрицательная позиция меток элементов**: настройка по умолчанию для отрицательной позиции меток элементов используется, когда не задана настройка для серии или настройка, заменяющая ее.

Пример категорийного столбчатого графика[1074], в котором метки элементов для положительных значений данных отображаются внутри столбцов, в то время как метки элементов для отрицательных значений данных отображаются под столбцами:



Приведенный выше график также имеет маркер категории[1178].

Имя свойства: **baseItemLabelPosition**

Тип свойства: **Data Table**

СМЕЩЕНИЕ ТОЧКИ ПРИВЯЗКИ МЕТОК ЭЛЕМЕНТОВ

Смещение точки привязки меток элементов позволяет управлять позициями меток элементов, контролируя расстояние, на которое точка привязки переносится с ее первоначальной позиции.

Имя свойства: **itemLabelAnchorOffset**

Тип свойства: **Float**

БАЗОВАЯ ФОРМА ЛЕГЕНДЫ

Фигура[1283] по умолчанию, используемая в легенде (легендах)[1173] графика.

Имя свойства: **baseLegendShape**

Тип свойства: **Data Table**

БАЗОВЫЙ ШРИФТ ТЕКСТА ЛЕГЕНДЫ

Шрифт[1278] по умолчанию, используемый в легенде (легендах)[1173] графика.

Имя свойства: **baseLegendTextFont**

Тип свойства: **Data Table**

БАЗОВАЯ ОКРАСКА ТЕКСТА ЛЕГЕНДЫ

Цвет[1279] текста по умолчанию, используемый в легенде (легендах)[1173] графика.

Имя свойства: **baseLegendTextPaint**

Тип свойства: **Data Table**

БАЗОВЫЙ ШТРИХ ОКАНТОВКИ

Штрих[1282] контура по умолчанию, используемый, если для серии недоступен другой штрих.

Имя свойства: **baseOutlineStroke**

Тип свойства: **Data Table**

Свойства для серии

ВИДИМОСТЬ СЕРИИ

Таблица, определяющая видимость отдельных серий данных.

Свойство	Имя	Тип	Описание
Индекс	index	целое	Индекс серии.
Видимость серии	seriesVisible	логическое	Флажок, указывающий, будет ли серия данных видимой.

Имя свойства: **seriesVisible**

Тип свойства: **Data Table**

АВТОЗАПОЛНЕНИЕ ОКРАСОК СЕРИЙ

Флажок, указывающий, что различные определенные заранее цвета будут использоваться вместо цвета по умолчанию.

Имя свойства: **autoPopulateSeriesPaints**

Тип свойства: **Boolean**

ОКРАСКА СЕРИЙ

Таблица, определяющая цвета для отдельных серий данных.

Свойство	Имя	Тип	Описание
Индекс	index	целое	Индекс серии.
Цвет серии	seriesPaint	таблица данных	Цвет 1279 серии.

Имя свойства: **seriesPaints**

Тип свойства: **Data Table**

АВТОЗАПОЛНЕНИЕ ШТРИХОВ СЕРИЙ

Флажок, указывающий, что различные определенные заранее штрихи будут использоваться вместо штриха по умолчанию.

Имя свойства: **autoPopulateSeriesStrokes**

Тип свойства: **Boolean**

ШТРИХ СЕРИЙ

Таблица, определяющая штрихи для отдельных серий данных.

Свойство	Имя	Тип	Описание
Индекс	index	целое	Индекс серии.
Штрих серии	seriesStroke	таблица данных	Штрих 1282 серии.

Имя свойства: **seriesStrokes**

Тип свойства: **Data Table**

АВТОЗАПОЛНЕНИЕ ФОРМ СЕРИЙ

Флажок, указывающий, что различные определенные заранее формы будут использоваться вместо формы по умолчанию.

Имя свойства: **autoPopulateSeriesShapes**

Тип свойства: **Boolean**

ФОРМА СЕРИЙ

Таблица, определяющая формы для отдельных серий данных.

Свойство	Имя	Тип	Описание
Индекс	index	целое	Индекс серии.
Форма серии	seriesShape	таблица данных	Форма 1283 серии.

Имя свойства: **seriesShapes**

Тип свойства: **Data Table**

АВТОЗАПОЛНЕНИЕ ЦВЕТОВ ЗАЛИВКИ

Флажок, указывающий, что различные заранее определенные цвета заливки будут использоваться вместо цвета заливки по умолчанию.

Имя свойства: **autoPopulateSeriesFillPaints**

Тип свойства: **Boolean**

ЦВЕТА ЗАЛИВКИ СЕРИЙ

Таблица, определяющая цвета заливки для отдельных серий данных.

Свойство	Имя	Тип	Описание
индекс	index	целое	Индекс серии.
Цвет заливки серии	seriesFillPaint	таблица данных	Цвет 1279 заливки серии.

Имя свойства: **seriesFillPaints**

Тип свойства: **Data Table**

АВТОЗАПОЛНЕНИЕ ЦВЕТОВ ОКАНТОВКИ СЕРИЙ

Флажок, указывающий, что различные определенные заранее цвета контуров будут использоваться вместо цвета контуров по умолчанию.

Имя свойства: **autoPopulateSeriesOutlinePaints**

Тип свойства: **Boolean**

ОКРАСКА ОКАНТОВКИ СЕРИЙ

Таблица, определяющая цвета контура для отдельных серий данных.

Свойство	Имя	Тип	Описание
Индекс	index	целое	Индекс серии.
Цвет контура серии	seriesOutlinePaint	таблица данных	Цвет 1279 контура серии.

Имя свойства: **seriesOutlinePaints**

Тип свойства: **Data Table**

АВТОЗАПОЛНЕНИЕ ШТРИХОВ ОКАНТОВКИ СЕРИЙ

Флажок, указывающий, что различные определенные заранее штрихи контуров будут использоваться вместо Штриха контура по умолчанию.

Имя свойства: **autoPopulateSeriesOutlineStrokes**

Тип свойства: **Boolean**

ШТРИХ ОКАНТОВКИ СЕРИЙ

Таблица, определяющая штрихи контура для отдельной серии данных.

Свойство	Имя	Тип	Описание
Индекс	index	целое	Индекс серии.
Штрих контура серии	seriesOutlineStroke	таблица данных	Штрих 1282 контура серии.

Имя свойства: **seriesOutlineStrokes**

Тип свойства: **Data Table**

ВИДИМОСТЬ СЕРИЙ В ЛЕГЕНДЕ

Таблица, определяющая видимость легенды для отдельных серий данных.

Свойство	Имя	Тип	Описание
Индекс	index	целое	Индекс серии.
Видимость серии в легенде	seriesVisibleInLegend	логическое	Флажок, указывающий, будет ли видима серия данных в легенде.

Имя свойства: **seriesVisibleInLegend**

Тип свойства: **Data Table**

СВОЙСТВА ЛЕГЕНДЫ

Таблица, определяющая представление отдельных серий в легенде.

Свойство	Имя	Тип	Описание
Индекс	index	целое	Индекс серии.
Форма	shape	таблица данных	Форма 1283, используемая для представления серии в легенде.
Цвет текста	paint	таблица данных	Цвет 1279, используемый для представления серии в легенде.
Шрифт текста	font	таблица данных	Шрифт 1278, используемый для представления серии в легенде.

Имя свойства: **legendProperties**

Тип свойства: **Data Table**

СВОЙСТВА МЕТОК ЭЛЕМЕНТОВ СЕРИИ

Таблица, определяющая представление меток элементов для отдельных серий.

Свойство	Имя	Тип	Описание
Индекс	index	целое	Индекс серий.
Видимость	visible	логическое	Флажок, контролирующий видимость меток элементов.
Шрифт	font	таблица	Шрифт 1278 меток элементов.

		данных	
Цвет	paint	таблица данных	Цвет[1279] меток элементов.
Положительная позиция элементов	positiveItemsPosition	таблица данных	Позиция меток[1172] для положительных элементов в серии.
Отрицательная позиция элементов	negativeItemsPosition	таблица данных	Позиция меток[1172] для отрицательных элементов в серии.

Имя свойства: **seriesItemLabelProperties**

Тип свойства: **Data Table**

13.4.11.10.1 Категорийный отрисовщик

Отрисовщик категорий является основным отрисовщиком для элементов данных в области построения[1187]. Он наследует свойства базового отрисовщика[1221] и добавляет их к своим свойствам.

Свойства

БАЗОВЫЙ ГЕНЕРАТОР МЕТОК ЭЛЕМЕНТОВ

Генератор меток элементов[1196] по умолчанию, используемый для создания меток элементов, когда не задан пользовательский генератор для серий.

Имя свойства: **baseItemLabelGenerator**

Тип свойства: **Data Table**

БАЗОВЫЙ ГЕНЕРАТОР ПОДСКАЗОК

Генератор всплывающих подсказок[1196] по умолчанию, используемый для создания всплывающих подсказок, когда не задан пользовательский генератор для серий.

Имя свойства: **baseToolTipGenerator**

Тип свойства: **Data Table**

ГЕНЕРАТОР МЕТОК ЭЛЕМЕНТОВ ЛЕГЕНДЫ

Генератор для имени серии в легенде.

Форматная строка может содержать символ {0}, представляющий имя серии.

Имя свойства: **legentItemLabelGenerator**

Тип свойства: **Data Table**

ГЕНЕРАТОР ПОДСКАЗОК ЭЛЕМЕНТОВ ЛЕГЕНДЫ

Генератор всплывающих подсказок для легенды.

Форматная строка может содержать символ {0}, представляющий имя серии.

Имя свойства: **legentItemToolTipGenerator**

Тип свойства: **Data Table**

ГЕНЕРАТОРЫ МЕТОК ЭЛЕМЕНТОВ СЕРИЙ

Таблица, содержащая генераторы меток элементов для определенных серий данных.

Свойство	Имя	Тип	Описание
Индекс	index	целое	Индекс серии.

Генератор меток элементов серии	seriesItemLabelGenerator	таблица данных	Генератор меток элементов 1196 для серии.

Имя свойства: **seriesItemLabelGenerators**

Тип свойства: **Data Table**

ГЕНЕРАТОРЫ ВЫСПЛЫВАЮЩИХ ПОДСКАЗОК ЭЛЕМЕНТОВ СЕРИЙ

Таблица, содержащая генераторы всплывающих подсказок для определенных серий данных.

Свойство	Имя	Тип	Описание
Индекс	index	целое	Индекс серии.
Генератор всплывающих подсказок серии	seriesToolTipGenerator	таблица данных	Генератор всплывающих подсказок 1196 для серии.

Имя свойства: **seriesToolTipGenerators**

Тип свойства: **Data Table**

13.4.11.10.1.1 Отрисовщик столбцов категорий

Данный отрисовщик является подтипом Отрисовщика категорий 1228 и используется вместе с Категорийной областью построения 1187 для создания столбчатых графиков. Он унаследует все свойства Категорийного отрисовщика и имеет свои собственные свойства.

Контролирование ширины столбцов

Отрисовщик автоматически рассчитывает ширину столбцов так, чтобы они занимали все доступное пространство области построения, поэтому вы не можете определять ширину столбцов напрямую. Однако ширина столбцов является функцией следующих свойств:

ГРАНИЦА ЭЛЕМЕНТОВ

Граница элементов в процентах от общей длины оси категорий (по умолчанию, 0.20 или двадцать процентов). Данное свойство контролирует пространство, предназначенное для интервалов между столбцами в пределах одной категории.

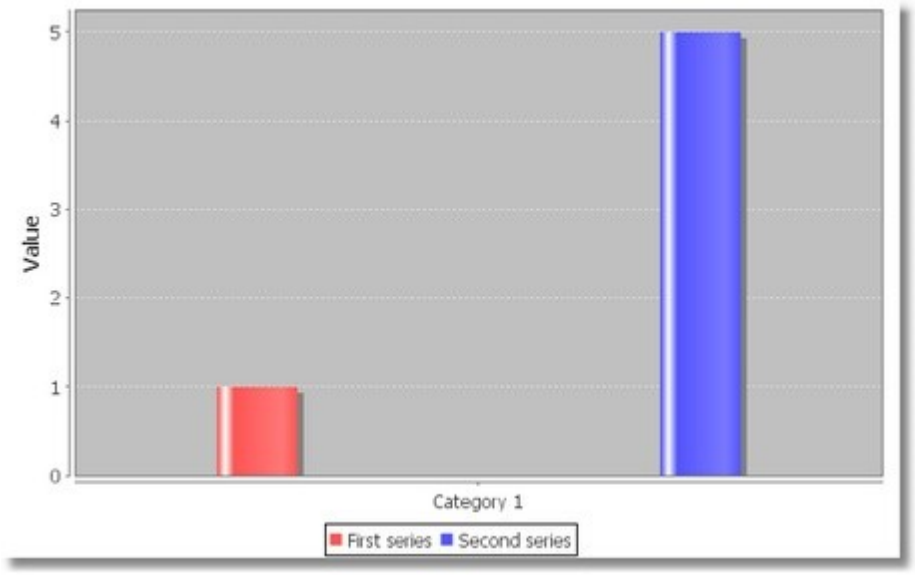
Имя свойства: **itemMargin**

Тип свойства: **Float**

МАКСИМАЛЬНАЯ ШИРИНА СТОЛБЦА

Максимальная ширина столбца в процентах от длины оси. Например, значение, равное 0.05, определяет, что ширина столбцов никогда не будет превышать пяти процентов от длины оси. Это позволяет улучшить внешний вид графиков, в которых отображаются один или два столбца.

Пример графика с ограниченной максимальной шириной столбцов:



Имя свойства: **maximumBarWidth**

Тип свойства: **Float**

Базовое значение

По умолчанию отрисовщик отображает столбец между нулем (базовым значением) и значением данных отображаемого элемента. Некоторые графики требуют ненулевое базовое значение.

БАЗОВОЕ ЗНАЧЕНИЕ

Базовое значение для столбцов.

Имя свойства: **base**

Тип свойства: **Float**

ВКЛЮЧАТЬ БАЗОВОЕ ЗНАЧЕНИЕ В ДИАПАЗОН

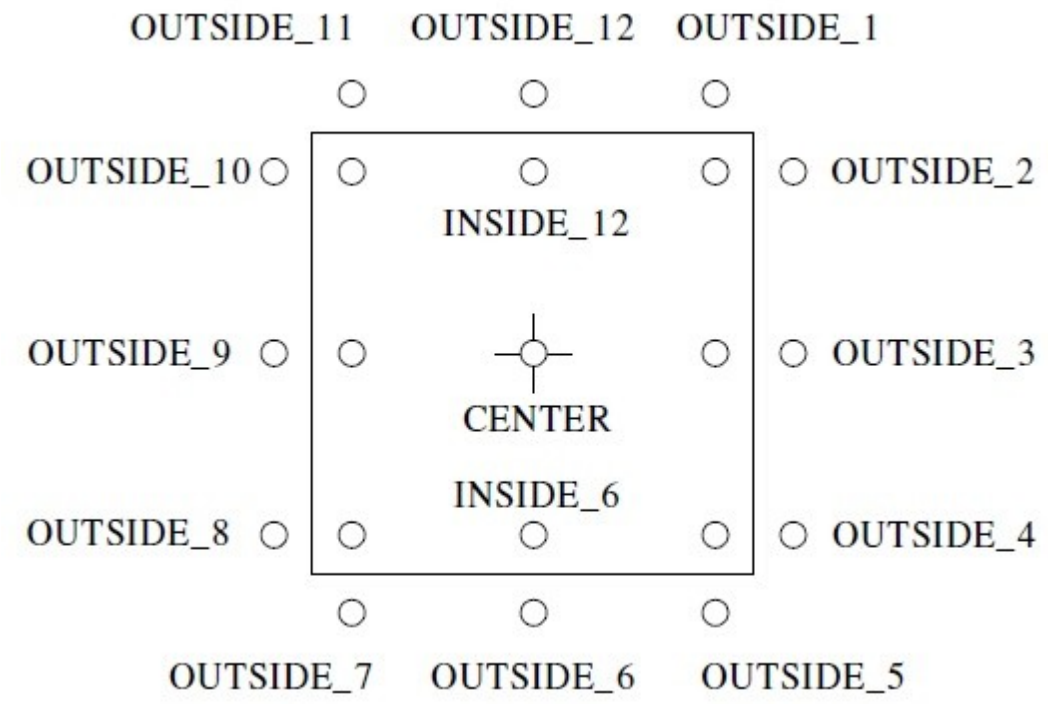
Флажок, контролирующий, будет ли учитываться базовое значение при расчете автодиапазона для оси измерений.

Имя свойства: **includeBaseInRange**

Тип свойства: **Boolean**

Метки элементов

Так как столбцы имеют прямоугольную форму, отрисовщик рассчитывает точки привязки так, как указано на рисунке далее. Имейте в виду, что позиции чисел соответствуют (приблизительно) положению часов на циферблате.



Когда метка элемента отображается внутри столбца, отрисовщик также рассчитывает, является ли размер столбца достаточным для вмещения текста. Если нет, отрисовщик проверяет, задана ли для нее альтернативная позиция. Если таковая имеется, метка отображается в ней, если нет, метка не отображается вообще. Могут быть заданы две альтернативные позиции, одна для положительных значений, вторая -- для отрицательных (например, в случае, когда метки положительных значений, непоместившиеся внутри столбца, отображаются выше столбцов, а метки отрицательных значений, непоместившиеся внутри столбца, отображаются ниже).

АЛЬТЕРНАТИВНОЕ ПОЛОЖЕНИЕ МЕТОК ЭЛЕМЕНТОВ

Данное свойство включает два значения:

- **Позиция меток положительных элементов**[1172]: альтернативная позиция для меток положительных значений. Установите значение на null, если вы хотите скрыть значения, которые не помещаются внутри столбца.
- **Позиция меток отрицательных элементов**: альтернативная позиция для меток отрицательных значений. Установите значение на null, если вы хотите скрыть значения, которые не помещаются внутри столбца.

Имя свойства: **itemLabelPositionFallback**

Тип свойства: **Data Table**

Тени столбцов

ВИДИМОСТЬ ТЕНИ

Флажок, контролирующий отображение теней столбцов.

Имя свойства: **shadowsVisible**

Тип свойства: **Boolean**

ОКРАСКА ТЕНИ

Цвет[1279] заливки теней столбцов.

Имя свойства: **shadowPaint**

Тип свойства: **Data Table**

СМЕЩЕНИЕ ТЕНИ ПО X

Смещение теней столбцов по оси X.

Имя свойства: **shadowXOffset**

Тип свойства: **Float**

СМЕЩЕНИЕ ТЕНИ ПО Y

Смещение теней столбцов по оси Y.

Имя свойства: **shadowYOffset**

Тип свойства: **Float**

Другие свойства

ОКРАСКА СТОЛБЦОВ

Отрисовщик столбцов отвечает за фактическое отображение отдельного столбца. Он обладает следующими свойствами:

Свойство	Имя	Тип	Описание
Тип	type	целое	Тип редактора: **Стандартный** или **Градиентный**: • Стандартный редактор столбцов использует для заливки один Цвет[1279] (текущий цвет серии данных). • Градиентный редактор столбцов применяет эффект нескольких областей градиента, чтобы придать столбцам оригинальный вид.
G1	g1	плавающее	Точка деления между первой и второй областями градиента (больше 0.0).
G2	g2	плавающее	Точка деления между второй и третьей градиентными областями (больше G1).
G3	g3	плавающее	Точка деления между третьей и четвертой градиентными областями (больше G2 и меньше 1.0).

Имя свойства: **barPainter**

Тип свойства: **Data Table**

ОКАНТОВКА СТОЛБЦОВ

Флажок, контролирующий, будет ли отображаться контур для каждого столбца. Цвет и штрих контура определяются при помощи свойств основного отрисовщика графиков.

Имя свойства: **drawBarOutline**

Тип свойства: **Boolean**

МАКСИМАЛЬНАЯ ДЛИНА СТОЛБЦА

Устанавливает минимальную длину, применимую к столбцам. Вы можете задать малое значение (например, 1.0), чтобы предотвратить отображение слишком коротких столбцов.

Имя свойства: **minimumBarLength**

Тип свойства: **Float**

13.4.11.10.1.2 Отрисовщик линейных категорий

Данный отрисовщик является подтипом Отрисовщика категорий[1228] и используется вместе с Категорийной областью построения[1187] для создания столбчатых графиков. Он унаследует все свойства Категорийного отрисовщика и имеет свои собственные свойства.

Свойства

БАЗОВАЯ ВИДИМОСТЬ СЕРИЙ

Таблица, определяющая видимость линии между элементами для индивидуальных серий данных.

Свойство	Имя	Тип	Описание
Индекс	index	целое	Индекс серии.

Видимость линий серий	seriesLinesVisible	логическое	Флажок, контролирующий видимость линий между элементами для серий данных.

Имя свойства: **seriesLinesVisible**

Тип свойства: **Data Table**

ВИДИМОСТЬ ФОРМ СЕРИЙ

Таблица, определяющая видимость форм элементов для индивидуальных серий данных.

Свойство	Имя	Тип	Описание
Индекс	index	целое	Индекс серии.
Видимость форм серий	seriesShapesVisible	логическое	Флажок, контролирующий видимость форм элементов серий данных.

Имя свойства: **seriesShapesVisible**

Тип свойства: **Data Table**

ЗАЛИВКА ФОРМ СЕРИЙ

Таблица, контролирующая заливку форм для индивидуальных серий данных.

Свойство	Имя	Тип	Описание
Флажок, контролирующий видимость	index	целое	Индекс серии.
Заливка форм серий	seriesShapesFilled	логическое	Флажок, контролирующий заливку форм элементов для серии данных.

Имя свойства: **seriesShapesFilled**

Тип свойства: **Data Table**

БАЗОВАЯ ВИДИМОСТЬ ЛИНИЙ

Значение по умолчанию для видимости линий между элементами данных.

Имя свойства: **baseLinesVisible**

Тип свойства: **Boolean**

ВИДИМОСТЬ ЛИНИЙ СЕРИЙ

Значение по умолчанию для видимости форм элементов.

Имя свойства: **baseShapesVisible**

Тип свойства: **Boolean**

БАЗОВАЯ ЗАЛИВКА ФОРМ

Настройка по умолчанию для заливки форм элементов.

Имя свойства: **baseShapesFilled**

Тип свойства: **Boolean**

ОТРИСОВКА ОКАНТОВОК

Флажок, контролирующий, будут ли отображаться контуры форм элементов.

Имя свойства: **drawOutlines**

Тип свойства: **Boolean**

ИСПОЛЬЗОВАТЬ ЗАЛИВКУ

Флажок, контролирующий, будет ли использоваться цвет заливки для серии или же обычный цвет серий для заливки форм элементов.

Имя свойства: **useFillPaint**

Тип свойства: **Boolean**

ИСПОЛЬЗОВАТЬ ОКАНТОВКУ

Флажок, определяющий какой цвет будет использоваться для контуров форм. Отрисовщик использует один из двух возможных цветов:

- Цвет контура для текущей серии.
- Обычный цвет текущей серии.

Данное свойство пригодно для Линейного и Линейного 3D отрисовщиков.

Имя свойства: **useOutlinePaint**

Тип свойства: **Boolean**

СМЕЩЕНИЕ СЕРИЙ

Флажок, контролирующий, будут ли ступенчатые линии серий данных смещены относительно друг друга.

Имя свойства: **useSeriesOffset**

Тип свойства: **Boolean**

13.4.11.10.2 Координатный отрисовщик

Координатный отрисовщик является основным отрисовщиком для элементов данных в Координатной области построения 1197. Он унаследует свойства основного отрисовщика 1221 и добавляет их к своим собственным.

Свойства

БАЗОВЫЙ ГЕНЕРАТОР МЕТОК ЭЛЕМЕНТОВ

Координатный генератор меток 1207 по умолчанию, используемый для создания меток элементов, если для серии не задан пользовательский генератор.

Имя свойства: **baseItemLabelGenerator**

Тип свойства: **Data Table**

БАЗОВЫЙ ГЕНЕРАТОР ПОДСКАЗОК

Координатный генератор всплывающих подсказок 1214 по умолчанию, используемый для создания всплывающих подсказок, если для серии не задан пользовательский генератор.

Имя свойства: **baseToolTipGenerator**

Тип свойства: **Data Table**

ГЕНЕРАТОР МЕТОК ЭЛЕМЕНТОВ ЛЕГЕНДЫ

Генератор для имен серий в легенде.

Форматная строка может содержать символ {0} для представления имени серии.

Имя свойства: **legentItemLabelGenerator**

Тип свойства: **Data Table**

ГЕНЕРАТОР ПОДСКАЗОК ЭЛЕМЕНТОВ ЛЕГЕНДЫ

Генератор всплывающих подсказок для элементов легенды.

Форматная строка может содержать символ {0} для представления имени серии.

Имя свойства: **legentItemToolTipGenerator**

Тип свойства: **Data Table**

ГЕНЕРАТОРЫ МЕТОК ЭЛЕМЕНТОВ СЕРИЙ

Таблица, определяющая генераторы меток элементов для отдельной серии данных.

Свойство	Имя	Тип	Описание
Индекс	index	целое	Индекс серии.
Генератор меток элементов серии	seriesItemLabelGenerator	таблица данных	Генератор меток элементов 1196 для серии.

Имя свойства: **seriesItemLabelGenerators**

Тип свойства: **Data Table**

ГЕНЕРАТОРЫ ВЫСПЛЫВАЮЩИХ ПОДСКАЗОК ЭЛЕМЕНТОВ СЕРИЙ

Таблица, определяющая генераторы всплывающих подсказок для отдельной серии данных.

Свойство	Имя	Тип	Описание
Индекс	index	целое	Индекс серии.
Генератор всплывающих подсказок элементов серии	seriesToolTipGenerator	таблица данных	Генератор всплывающих подсказок 1196 серии.

Имя свойства: **seriesToolTipGenerators**

Тип свойства: **Data Table**

ПОРЯДОК СОРТИРОВКИ СЕРИЙ

Порядок отображения серий в массиве данных: **Прямой** или **Обратный.**

Серии по умолчанию отображаются в обратном порядке (так, чтобы первая серия появлялась впереди всех других в массиве данных).

Имя свойства: **seriesRenderingOrder**

Тип свойства: **String**

АННОТАЦИИ ОТРИСОВКИ

Аннотации отрисовки имеют два отличия от аннотаций области построения 1198:

1. Возможно задать *Слой* для каждой аннотации, в итоге чего она появится снизу или сверху данных.
2. Когда координатный график добавляется в качестве подграфика к смешанному координатному графику 1139, аннотации области построения определяются внутри графика-родителя и привязаны к его основным осям, в то время как аннотации отрисовщиков привязаны к осям подграфика.

Свойства аннотаций отрисовщиков:

Свойство	Имя	Тип	Описание
Аннотация	annotation	таблица данных	Тип аннотации и таблица свойств. Доступны следующие стандартные типы аннотаций: • Бокс 1205 • Изображение на данных 1206 • Изображение 1206 • Легенда 1208 • Линия 1208 • Указатель 1209 • Многоугольник 1211

			• Форма 1212 • Текст 1213 • Заголовок 1214
Слой	layer	строка	Слой, на который накладывается аннотация: **Передний план** или **Фон**.

Имя свойства: **rendererAnnotations**

Тип свойства: **Data Table**

13.4.11.10.2.1 Координатный линейный отрисовщик

Данный отрисовщик является подтипом Координатного отрисовщика 1234 и используется вместе с Координатной областью построения 1197 для создания XY-графиков. Он отображает данные в виде форм и/или соединяя точки данных прямыми линиями.

Свойства

ВИДИМОСТЬ ЛИНИЙ СЕРИИ

Таблица, определяющая видимость линии между элементами для отдельных серий данных.

Свойство	Имя	Тип	Описание
Индекс	index	целое	Индекс серии.
Видимость линий серии	seriesLinesVisible	логическое	Флажок, контролирующий видимость линий между элементами для отдельных серий данных.

Имя свойства: **seriesLinesVisible**

Тип свойства: **Data Table**

ВИДИМОСТЬ ФОРМ СЕРИЙ

Таблица, определяющая видимость форм элементов для отдельных серий данных..

Свойство	Имя	Тип	Описание
Индекс	index	целое	Индекс серии.
Видимость форм серий	seriesShapesVisible	логическое	Флажок, контролирующий видимость форм элементов серий данных.

Имя свойства: **seriesShapesVisible**

Тип свойства: **Data Table**

ЗАЛИВКА ФОРМ СЕРИЙ

Таблица, контролирующая заливку форм для отдельных серий данных.

Свойство	Имя	Тип	Описание
Индекс	index	целое	Индекс серии.
Заливка форм серий	seriesShapesFilled	логическое	Флажок, контролирующий заливку форм элементов для серии данных.

Имя свойства: **seriesShapesFilled**

Тип свойства: **Data Table**

БАЗОВАЯ ВИДИМОСТЬ ЛИНИЙ

Значение по умолчанию для видимости линий между элементами данных.

Имя свойства: **baseLinesVisible**

Тип свойства: **Boolean**

ВИДИМОСТЬ ФОРМ СЕРИЙ

Значение по умолчанию для видимости форм элементов.

Имя свойства: **baseShapesVisible**

Тип свойства: **Boolean**

ЗАЛИВКА ФОРМ СЕРИЙ

Настройка по умолчанию для заливки форм элементов.

Имя свойства: **baseShapesFilled**

Тип свойства: **Boolean**

ЛИНИЯ ЛЕГЕНДЫ

Данный отрисовщик позволяет настроить отображение легенды таким образом, чтобы вы могли задать форму (обычно линию, хотя можно использовать любые фигуры), которые будут представлять каждую серию в легенде.

Имя свойства: **legendLine**

Тип свойства: **Data Table**

ОТРИСОВКА ОКАНТОВОК

Флажок, контролирующий, будут ли отображаться контуры форм элементов.

Имя свойства: **drawOutlines**

Тип свойства: **Boolean**

НЕПРЕРЫВНАЯ ЛИНИЯ СЕРИИ

Очень часто используется пунктирная линия для отображения линий соединений между элементами серии данных. Проблема возникает, когда элементы в серии расположены близко друг к другу на графике, потому что по умолчанию отрисовщик отображает линию соединений отдельно для каждого элемента данных (соединяя текущий элемент с предыдущим). Штрих для линии устанавливается для каждого сегмента, что может привести к тому, что штрих будет не виден для серии. Возможно также отображение соединительных линий целой серии в виде сплошной линии.

Флажок **Отображать линию серии в виде пути** контролирует, будут ли отображаться линии соединений между элементами в виде одной линии ("пути") или каждая в отдельности.

Имя свойства: **drawSeriesLineAsPath**

Тип свойства: **Boolean**

ИСПОЛЬЗОВАТЬ ЗАЛИВКУ

Флажок, контролирующий, будет ли использоваться цвет заливки для серии или же обычный цвет серии для заливки форм элементов.

Имя свойства: **useFillPaint**

Тип свойства: **Boolean**

ИСПОЛЬЗОВАТЬ ОКАНТОВКУ

Флажок, определяющий, какой цвет будет использоваться для контуров форм. Отрисовщик использует один из двух возможных цветов:

- Цвет контура для текущей серии.
- Обычный цвет текущей серии.

Имя свойства: **useOutlinePaint**

Тип свойства: **Boolean**

13.4.12 Индикаторы

Индикаторы представляют собой пригодную для чтения визуализацию числовых значений. Поддерживаются следующие типы индикаторов:


Простой индикатор 1238


Простой дуговой индикатор 1244


Круговой индикатор 1249


Круговой гистограммный индикатор 1250


Дуговой индикатор 1251


Полукруглый индикатор 1252


Четвертной индикатор 1252


Линейный индикатор 1253


Линейный гистограммный индикатор 1254


Индикатор-счетчик 1255

13.4.12.1 Простой индикатор

Этот компонент отображает простой индикатор с одним или несколькими указателями и шкалами.

Простой индикатор выглядит следующим образом:

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Граница 1275, Курсор 1276, Всплывающее меню 1276

Пользовательские свойства

ЗНАЧЕНИЯ ДАННЫХ (Свойство по умолчанию 1274)

Список значений данных, которые указываются индикатором. Это индексированное свойство 1274.

Имя свойства: **datasets**

Тип свойства: **Data Table**

ФОН

Фон индикатора (может быть сплошной цвет, линейный/радиальный градиент или текстура).

Поля фона индикатора:

Поле	Тип	Описание
Видимый	логическое	Флажок, определяющий, изображен ли фон.
Цвет	таблица данных	Цвет 1279, используемый для заливки фона.

Имя свойства: **gaugeBackground**

Тип свойства: **Data Table**

РАМКА

Свойства рамки кругового типа.

Поля рамки индикатора:

Поле	Тип	Описание
Видимый	логическое	Флажок, определяющий, изображена ли шкала.
Цвет фона	таблица данных	Цвет 1279, используемый для заливки пространства между двойными линиями вокруг внешней границы круговой рамки.
Цвет переднего фона	таблица данных	Цвет 1279, используемый для изображения двойных линий вокруг внешней границы индикатора.
Радиус	плавающее	Радиус, выраженный в виде процента, относящегося к треугольной рамке.
Штрих	таблица данных	Штрих 1282, используемый для изображения контура рамки.

Имя свойства: **frame**

Тип свойства: **Data Table**

ОБЗОР

Размеры поля построения индикатора рассчитываются относительно прямоугольной рамки, но в некоторых случаях требуется установить только подмножество поля в области построения. Параметр **Обзор** позволяет сделать это при помощи установки обозреваемого прямоугольника относительно прямоугольной рамки области построения.

Поля обзора:

Поле	Тип	Описание
X	плавающее	Относительная позиция обзора x-координаты прямоугольника с ссылкой на прямоугольную рамку. Значение может варьироваться от 0.0 до 1.0.
Y	плавающее	Относительная позиция обзора y-координаты прямоугольника с ссылкой на прямоугольную рамку. Значение может варьироваться от 0.0 до 1.0.
Ширина	плавающее	Относительная ширина обозреваемого прямоугольника с ссылкой на прямоугольную рамку. Значение может варьироваться от 0.0 до 1.0.
Высота	плавающее	Относительная высота обозреваемого прямоугольника с ссылкой на прямоугольную рамку. Значение может варьироваться от 0.0 до 1.0.

Имя свойства: **view**

Тип свойства: **Data Table**

КРЫШКА

Крышка индикатора. Это кружок в середине индикатора (к которому присоединяются указатели).

Поля крышки индикатора:

Поле	Тип	Описание
Видимый	логическое	Флажок, определяющий, изображена ли крышка.
Цвет заливки	таблица данных	Цвет 1279 заливки крышки.
Цвет контура	таблица данных	Цвет 1279 контура крышки.
Штрих контура	таблица данных	Штрих 1282 контура крышки.
Радиус	плавающее	Радиус для крышки в виде процента от размера прямоугольной рамки индикатора.

Имя свойства: **cap**

Тип свойства: **Data Table**

ШКАЛЫ

Таблица шкал индикатора и их ассоциации со значениями данных 1239. Каждая запись имеет поле **Индекс Значений Данных**, которое определяет, какое значение данных обозначено на шкале.

Шкала может отображать основные и дополнительные деления с определенным пользователем интервалом и числовые значения для основных меток деления. Чтобы контролировать дугу, на которой изображена шкала, вы можете определить стартовый угол и диапазон (в градусах).

Поля таблицы шкал:

Поле	Тип	Описание
Индекс значения данных	целое	Индекс значения данных, указанных на шкале.
Шкала	таблица данных	Свойства шкалы.

Поля шкалы индикатора:

Поле	Описание
Видимый	Флажок, определяющий, отображена ли шкала.
Нижняя граница	Нижняя граница шкалы.
Верхняя граница	Верхняя граница шкалы.
Стартовый угол	Угол (в градусах) стартовой точки шкалы.
Диапазон	Диапазон (в градусах) шкалы.
Радиус деления	Радиус деления определяет, насколько далеко на индикаторе изображены деления (обычно они появляются рядом с внешним краем индикатора, но для индикатора, который показывает больше одной шкалы, вы, наверняка, захотите поместить шкалу ближе к центру). Радиус деления определяет точки нахождения основных и дополнительных делений, равно как и метки основных делений. Он представляется в виде процента от прямоугольной рамки индикатора.
Интервал основных делений	Интервал между основными делениями.
Длина основных делений	Длина основных делений, выраженная в количестве, которое нужно вычесть из радиуса делений.
Цвет основных делений	Цвет 1279, используемый для изображения основных делений.
Штрих основных делений	Штрих 1282, используемый для изображения основных делений.

Количество дополнительных делений	Количество дополнительных делений, изображаемое между основными делениями.
Длина дополнительных делений	Длина дополнительных делений в виде радиуса, обозначенного процентом от прямоугольной рамки индикатора.
Цвет дополнительных делений	Цвет1279, используемый для изображения дополнительных делений.
Штрих дополнительных делений	Штрих1279, используемый для изображения дополнительных делений.
Шрифт метки деления	Шрифт1278, используемый для изображения меток делений.
Форматтер метки деления	Числовой форматтер2147, используемый для конвертирования значений меток делений в строки.
Смещение метки деления	Точка привязки для каждой метки определяется использованием радиуса метки, настроенного при помощи значения смещения. Как только определяется точка привязки, метка значения центрируется на точке привязки.
Цвет метки деления	Цвет1279, используемый для изображения меток деления.
Видимая область меток деления	Флажок, определяющий, видимы ли метки деления.
Видимая область метки первого деления	Флажок, определяющий, видима ли метка первого деления.

Имя свойства: **scales**

Тип свойства: **Data Table**

ОБЛАСТЬ ЗНАЧЕНИЙ

Область значений индикатора. Каждая область представлена в виде цветной дуги. Она используется для выделения области значений, например, "обычное", "высокое" и "предельное". Область значений в основном определяется ее нижними и верхними границами. Выделения области значений появляются на определенном радиусе в пределах индикатора.

Поля области значений:

Поле	Тип	Описание
Видимый	логическое	Флажок, определяющий, изображается ли область значения.
Индекс шкалы	целое	Область значений измеряется относительно шкалы со специальным индексом.
Нижняя граница	плавающее	Нижняя граница области значений, в единицах данных.
Верхняя граница	плавающее	Верхняя граница области значений, в единицах данных.
Внутренний радиус	плавающее	Радиус внутреннего выделения области значений в виде процента от размера прямоугольной рамки индикатора.
Внешний радиус	плавающее	Радиус внешнего выделения области значений в виде процента от размера прямоугольной рамки индикатора.
Цвет	таблица данных	Цвет1279, используемый для выделения области значений.

Имя свойства: **ranges**

Тип свойства: **Data Table**

ОБЫЧНЫЕ УКАЗАТЕЛИ

Список обычных указателей, имеющих длинную треугольную форму. Указатели используют для обозначения текущего значения на индикаторе.

Поля обычных указателей:

Поле	Тип	Описание
Индекс значения данных	целое	Индекс значения данных 1239, обозначенный указателем.
Видимый	логическое	Флажок, определяющий, отображен ли указатель.
Цвет заливки	таблица данных	Цвет 1279 указателя.
Цвет контура	таблица данных	Цвет 1279 контура указателя.
Радиус	плавающее	Длина указателя в виде процента от размера прямоугольной рамки индикатора.
Ширина радиуса	плавающее	Ширина основания указателя определена как радиус в процентах относительно размера прямоугольной рамки индикатора.

Имя свойства: **normalPointers**

Тип свойства: **Data Table**

БУЛАВОЧНЫЕ УКАЗАТЕЛИ

Список булавочных указателей. Булавочный указатель, изображенный как тонкая прямая линия. Указатели используются для указания текущего значения на индикаторе.

Поля булавочных указателей:

Поле	Тип	Описание
Индекс значения данных	целое	Индекс значения данных 1239, обозначенный указателем.
Видимый	логическое	Флажок, определяющий, отображен ли указатель.
Цвет	таблица данных	Цвет 1279 указателя.
Радиус	плавающее	Длина указателя в виде процента от размера прямоугольной рамки индикатора.
Штрих	таблица данных	Штрих 1282, используемый для изображения указателя.

Имя свойства: **pinPointers**

Тип свойства: **Data Table**

ИНДИКАТОРЫ ЗНАЧЕНИЙ

Список индикаторо значений, показывающий текущие значения данных 1239.

Поля индикатора значений:

Поле	Тип	Описание
Индекс значения данных	целое	Индекс значения данных 1239, изображаемый индикатором.
Видимый	логическое	Флажок, определяющий, изображен ли индикатор.
Угол	плавающее	Точка привязки для позиционирования индикатора определяется при помощи угла и радиуса. Данное свойство - это угол, в градусах, для расчета точки привязки.
Радиус	плавающее	Радиус относительно прямоугольной рамки индикатора.

Шрифт	таблица данных	Шрифт1278, используемый для изображения значения индикатора.
Рамочная точка привязки	строка	Значение, которое индикатор отображает в виде текста в рамке. Точка привязки рамки определяет точку на рамке, которая будет выравнена относительно точки привязки (рассчитывается исходя из угла и радиуса).
Отступы	таблица данных	Отступы (сверху, слева, снизу, справа), которые определяют размер белого пространства вокруг значения индикатора.
Числовой формат	строка	Числовой форматтер2147, используемый для форматирования значения.
Цвет контура	таблица данных	Цвет1279, используемый для отрисовки контура вокруг индикатора.
Штрих контура	таблица данных	Штрих1282, используемый для отрисовки контура вокруг индикатора.
Цвет	таблица данных	Цвет1279, используемый для отображения значения индикатора.
Цвет фона	таблица данных	Цвет1279, используемый для заливки фона индикатора.
Эталон значения	плавающее	Чтобы убедиться, что индикатор значения имеет фиксированную ширину независимо от текущего значения в пакете данных, используется эталонное значение для расчета размеров индикатора. Это эталонное значение форматируется с использованием текущего числового форматтера, затем используются размеры строки для определения размеров значения индикатора.
Значение точки привязки	строка	Значение точки привязки определяет точку, относительно которой будет выравниваться текст значения.
Текст точки привязки	строка	Точка текста, выравниваемая относительно точки привязки значения.

Имя свойства: **indicators**

Тип свойства: **Data Table**

ТЕКСТОВЫЕ АННОТАЦИИ

Список текстовых аннотаций. Текстовая аннотация - это строка текста, отображаемая в произвольной области индикатора.

Поля текстовой аннтоации:

Поле	**Тип**	**Описание**
Метка	строка	Текст, отображаемый аннотацией.
Видимый	логическое	Флажок, определяющий, отображена ли аннотация.
Угол	плавающее	Точка привязки метки текста определяется углом и радиусом. Данное свойство - это угол (в градусах), который определяет точку привязки текста.
Радиус	плавающее	Радиус, используемый для определения точки привязки текста в виде процента относительно прямоугольной рамки индикатора.
Точка привязки	строка	Точка привязки определяет, как текст выравнивается относительно точки привязки.
Цвет	таблица данных	Цвет1279 метки.
Шрифт	таблица данных	Шрифт1278 метки.

Имя свойства: **annotations**

Тип свойства: **Data Table**

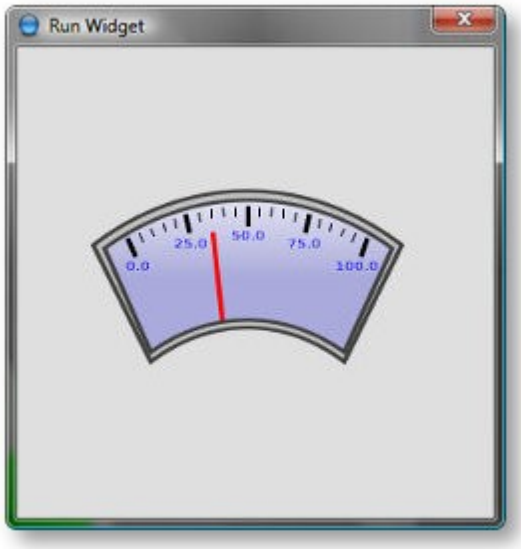
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.12.2 Простой дуговой индикатор

Этот компонент отображает простой дуговой индикатор с одним или несколькими указателями и шкалами.

Простой дуговой индикатор выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Граница 1275, Курсор 1276, Всплывающее меню 1276

Пользовательские свойства

ЗНАЧЕНИЯ ДАННЫХ (Свойство по умолчанию 1274)

Список значений данных, которые указываются индикатором. Это индексированное свойство 1274.

Имя свойства: **datasets**

Тип свойства: **Data Table**

ФОН

Фон индикатора (может быть сплошной цвет, линейный/радиальный градиент или текстура).

Поля фона индикатора:

Поле	Тип	Описание
Видимый	логическое	Флажок, определяющий, изображен ли фон.
Цвет	таблица данных	Цвет 1279, используемый для заливки фона.

Имя свойства: **gaugeBackground**

Тип свойства: **Data Table**

РАМКА

Поля рамки индикатора:

Поле	Тип	Описание
Видимый	логическое	Флажок, определяющий, изображается ли рамка.
Цвет переднего фона	таблица данных	Цвет1279, используемый для изображения двойных линий вокруг внешней стороны индикатора.
Стартовый угол	плавающее	Стартовый угол дуги, в градусах.
Размер	плавающее	Размер угла дуги, в градусах.
Внутренний радиус	плавающее	Внутренний радиус дуги, выраженный в процентах относительно прямоугольной рамки.
Внешний радиус	плавающее	Внешний радиус дуги, выраженный в процентах относительно прямоугольной рамки.
Штрих	таблица данных	Штрих1282, используемый для изображения контура рамки.

Имя свойства: **arcFrame**

Тип свойства: **Data Table**

ОБЗОР

Размеры поля построения индикатора рассчитываются относительно прямоугольной рамки, но в некоторых случаях требуется установить только подмножество поля в области построения. Параметр **Обзор** позволяет сделать это при помощи установки обозреваемого прямоугольника относительно прямоугольной рамки области построения.

Поля обзора:

Поле	Тип	Описание
X	плавающее	Относительная позиция обзора x-координаты прямоугольника с ссылкой на прямоугольную рамку. Значение может варьироваться от 0.0 до 1.0.
Y	плавающее	Относительная позиция обзора y-координаты прямоугольника с ссылкой на прямоугольную рамку. Значение может варьироваться от 0.0 до 1.0.
Ширина	плавающее	Относительная ширина обозреваемого прямоугольника с ссылкой на прямоугольную рамку. Значение может варьироваться от 0.0 до 1.0.
Высота	плавающее	Относительная высота обозреваемого прямоугольника с ссылкой на прямоугольную рамку. Значение может варьироваться от 0.0 до 1.0.

Имя свойства: **view**

Тип свойства: **Data Table**

КРЫШКА

Крышка индикатора. Это кружок в середине индикатора (к которому присоединяются указатели).

Поля крышки индикатора:

Поле	Тип	Описание
Видимый	логическое	Флажок, определяющий, изображена ли крышка.
Цвет заливки	таблица данных	Цвет1279 заливки крышки.
Цвет контура	таблица данных	Цвет1279 контура крышки.
Штрих контура	таблица данных	Штрих1282 контура крышки.
Радиус	плавающее	Радиус для крышки в виде процента от размера прямоугольной рамки индикатора.

Имя свойства: **cap**

Тип свойства: **Data Table**

ШКАЛЫ

Таблица шкал индикатора и их ассоциации со значениями данных[1239]. Каждая запись имеет поле **Индекс Значений Данных**, которое определяет, какое значение данных обозначено на шкале.

Шкала может отображать основные и дополнительные деления с определенным пользователем интервалом и числовые значения для основных меток деления. Чтобы контролировать дугу, на которой изображена шкала, вы можете определить стартовый угол и диапазон (в градусах).

Поля таблицы шкал:

Поле	Тип	Описание
Индекс значения данных	целое	Индекс значения данных, указанных на шкале.
Шкала	таблица данных	Свойства шкалы.

Поля шкалы индикатора:

Поле	Описание
Видимый	Флажок, определяющий, отображена ли шкала.
Нижняя граница	Нижняя граница шкалы.
Верхняя граница	Верхняя граница шкалы.
Стартовый угол	Угол (в градусах) стартовой точки шкалы.
Диапазон	Диапазон (в градусах) шкалы.
Радиус отметки	Радиус отметки определяет, насколько далеко на индикаторе изображены отметки (обычно они появляются рядом с внешним краем индикатора, но для индикатора, который показывает больше одной шкалы, вы, наверняка, захотите поместить шкалу ближе к центру). Радиус отметки определяет точки нахождения основных и дополнительных отметок, равно как и метки основных делений. Он представляется в виде процента от прямоугольной рамки индикатора.
Интервал основных отметок	Интервал между основными отметками.
Длина основных отметок	Длина основных отметок, выраженная в количестве, которое нужно вычесть из радиуса отметки.
Цвет основных отметок	Цвет[1279], используемый для изображения основных отметок.
Штрих основных отметок	Штрих[1282], используемый для изображения основных отметок.
Количество дополнительных отметок	Количество дополнительных отметок, изображаемое между основными отметками.
Длина дополнительных отметок	Длина дополнительных отметок в виде радиуса, обозначенного процентом от прямоугольной рамки индикатора.
Цвет дополнительных отметок	Цвет[1279], используемый для изображения дополнительных отметок.
Штрих дополнительных отметок	Штрих[1279], используемый для изображения дополнительных отметок.
Шрифт метки отметки	Шрифт[1278], используемый для изображения меток отметок.
Форматтер метки отметки	Числовой форматтер[2147], используемый для конвертирования значений меток отметок в строки.
Смещение метки отметки	Точка привязки для каждой метки определяется использованием радиуса метки, настроенного при помощи значения смещения. Как только определяется точка привязки, метка значения центрируется на точке привязки.

Цвет метки отметки	Цвет[1279], используемый для изображения меток отметки.
Видимая область меток отметки	Флажок, определяющий, видимы ли метки отметки.
Видимая область метки первой отметки	Флажок, определяющий, видима ли метка первой отметки.

Имя свойства: **scales**

Тип свойства: **Data Table**

ОБЛАСТЬ ЗНАЧЕНИЙ

Область значений индикатора. Каждая область представлена в виде цветной дуги. Она используется для выделения области значений, например, "обычное", "высокое" и "предельное". Область значений в основном определяется ее нижними и верхними границами. Выделения области значений появляются на определенном радиусе в пределах индикатора.

Поля области значений:

Поле	Тип	Описание
Видимый	логическое	Флажок, определяющий, изображается ли область значения.
Индекс шкалы	целое	Область значений измеряется относительно шкалы со специальным индексом.
Нижняя граница	плавающее	Нижняя граница области значений, в единицах данных.
Верхняя граница	плавающее	Верхняя граница области значений, в единицах данных.
Внутренний радиус	плавающее	Радиус внутреннего выделения области значений в виде процента от размера прямоугольной рамки индикатора.
Внешний радиус	плавающее	Радиус внешнего выделения области значений в виде процента от размера прямоугольной рамки индикатора.
Цвет	таблица данных	Цвет[1279], используемый для выделения области значений.

Имя свойства: **ranges**

Тип свойства: **Data Table**

ОБЫЧНЫЕ УКАЗАТЕЛИ

Список обычных указателей, имеющих длинную треугольную форму. Указатели используют для обозначения текущего значения на индикаторе.

Поля обычных указателей:

Поле	Тип	Описание
Индекс значения данных	целое	Индекс значения данных[1239], обозначенный указателем.
Видимый	логическое	Флажок, определяющий, отображен ли указатель.
Цвет заливки	таблица данных	Цвет[1279] указателя.
Цвет контура	таблица данных	Цвет[1279] контура указателя.
Радиус	плавающее	Длина указателя в виде процента от размера прямоугольной рамки индикатора.
Ширина радиуса	плавающее	Ширина основания указателя определена как радиус в процентах относительно размера прямоугольной рамки индикатора.

Имя свойства: **normalPointers**

Тип свойства: **Data Table**

БУЛАВОЧНЫЕ УКАЗАТЕЛИ

Список булавочных указателей. Булавочный указатель, изображенный как тонкая прямая линия. Указатели используются для указания текущего значения на индикаторе.

Поля булавочных указателей:

Поле	Тип	Описание
Индекс значения данных	целое	Индекс значения данных 1239, обозначенный указателем.
Видимый	логическое	Флажок, определяющий, отображен ли указатель.
Цвет	таблица данных	Цвет 1279 указателя.
Радиус	плавающее	Длина указателя в виде процента от размера прямоугольной рамки индикатора.
Штрих	таблица данных	Штрих 1282, используемый для изображения указателя.

Имя свойства: **pinPointers**

Тип свойства: **Data Table**

ИНДИКАТОРЫ ЗНАЧЕНИЙ

Список индикаторо значений, показывающий текущие значения данных 1239.

Поля индикатора значений:

Поле	Тип	Описание
Индекс значения данных	целое	Индекс значения данных 1239, изображаемый индикатором.
Видимый	логическое	Флажок, определяющий, изображен ли индикатор.
Угол	плавающее	Точка привязки для позиционирования индикатора определяется при помощи угла и радиуса. Данное свойство - это угол, в градусах, для расчета точки привязки.
Радиус	плавающее	Радиус относительно прямоугольной рамки индикатора.
Шрифт	таблица данных	Шрифт 1278, используемый для изображения значения индикатора.
Рамочная точка привязки	строка	Значение, которое индикатор отображает в виде текста в рамке. Точка привязки рамки определяет точку на рамке, которая будет выравнена относительно точки привязки (рассчитывается исходя из угла и радиуса).
Отступы	таблица данных	Отступы (сверху, слева, снизу, справа), которые определяют размер белого пространства вокруг значения индикатора.
Числовой формат	строка	Числовой форматтер 2147, используемый для форматирования значения.
Цвет контура	таблица данных	Цвет 1279, используемый для отрисовки контура вокруг индикатора.
Штрих контура	таблица данных	Штрих 1282, используемый для отрисовки контура вокруг индикатора.
Цвет	таблица данных	Цвет 1279, используемый для отображения значения индикатора.
Цвет фона	таблица данных	Цвет 1279, используемый для заливки фона индикатора.
Эталон значения	плавающее	Чтобы убедиться, что индикатор значения имеет фиксированную ширину независимо от текущего значения в пакете данных, используется эталонное

		значение для расчета размеров индикатора. Это эталонное значение форматируется с использованием текущего числового форматтера, затем используются размеры строки для определения размеров значения индикатора.
Значение точки привязки	строка	Значение точки привязки определяет точку, относительно которой будет выравниваться текст значения.
Текст точки привязки	строка	Точка текста, выравниваемая относительно точки привязки значения.

Имя свойства: **indicators**

Тип свойства: **Data Table**

ТЕКСТОВЫЕ АННОТАЦИИ

Список текстовых аннотаций. Текстовая аннотация - это строка текста, отображаемая в произвольной области индикатора.

Поля текстовой аннтоации:

Поле	**Тип**	**Описание**
Метка	строка	Текст, отображаемый аннотацией.
Видимый	логическое	Флажок, определяющий, отображена ли аннотация.
Угол	плавающее	Точка привязки метки текста определяется углом и радиусом. Данное свойство - это угол (в градусах), который определяет точку привязки текста.
Радиус	плавающее	Радиус, используемый для определения точки привязки текста в виде процента относительно прямоугольной рамки индикатора.
Точка привязки	строка	Точка привязки определяет, как текст выравнивается относительно точки привязки.
Цвет	таблица данных	Цвет[1279] метки.
Шрифт	таблица данных	Шрифт[1278] метки.

Имя свойства: **annotations**

Тип свойства: **Data Table**

Общие события

Скрытие[1285], Показ[1285], Перемещение[1285], Изменение размеров[1285], Клик мыши[1286], Нажатие кнопки мыши[1286], Отпускание кнопки мыши[1286], Вход мыши[1286], Выход мыши[1286], Перемещение мыши[1286], Вращение колесика мыши[1286], Печать клавиши[1287], Нажатие клавиши[1287], Отпускание клавиши[1287], Получение фокуса[1287], Потеря фокуса[1287]

13.4.12.3 Круговой индикатор

Этот компонент отображает легкоконфигурируемый круговой индикатор.

Компонент "Круговой индикатор" выглядит следующим образом:

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Граница 1275, Курсор 1276, Всплывающее меню 1276

Все Общие свойства индикатора 1255.

Все Общие свойства кругового индикатора 1266.

Пользовательские свойства

Не определены.

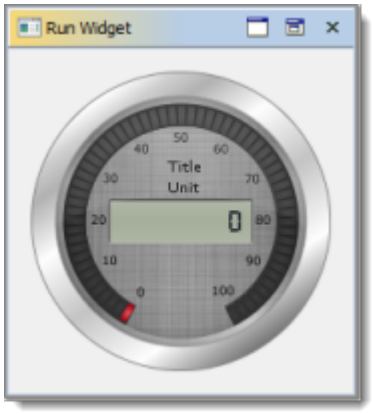
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.12.4 Круговой гистограммный индикатор

Этот компонент отображает легкоконфигурируемый гистограммный круговой индикатор.

Компонент "Круговой гистограммный индикатор" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Граница 1275, Курсор 1276, Всплывающее меню 1276

Все Общие свойства индикатора 1255.

Все Общие свойства кругового индикатора 1266.

Пользовательские свойства

ЦВЕТ ГИСТОГРАММЫ

Цвет гистограммы (одно из заранее определенных значений).

Имя свойства: **barGraphColor**

Тип свойства: **Color**

ВИДИМОЕ МАКСИМАЛЬНОЕ ЗНАЧЕНИЕ

Контролирует видимость максимального значения, которое является последним измеряемым значением.

Имя свойства: **peakValueVisible**

Тип свойства: **Boolean**

МАКСИМАЛЬНОЕ ЗНАЧЕНИЕ

Максимальное значение индикатора. Используется гистограммным индикатором для визуализации последнего отображаемого значения.

Имя свойства: **peakValue**

Тип свойства: **Double**

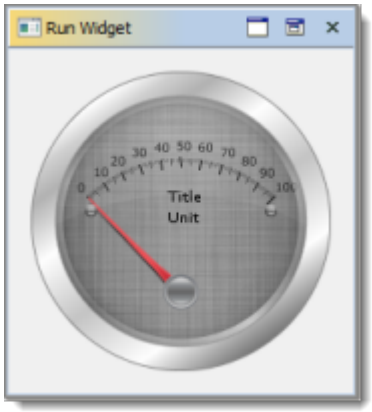
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.12.5 Дуговой индикатор

Этот компонент отображает легкоконфигурируемый дуговой индикатор.

Компонент "Дуговой индикатор" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Граница 1275, Курсор 1276, Всплывающее меню 1276

Все Общие свойства индикатора 1255.

Все Общие свойства кругового индикатора 1266

Пользовательские свойства

Не определены.

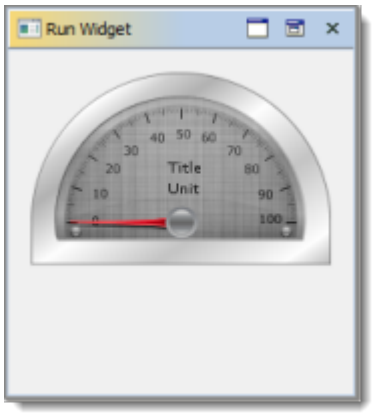
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.12.6 Полукруглый индикатор

Этот компонент отображает легкоконфигурируемый полукруглый индикатор.

Компонент "Полукруглый индикатор" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Граница 1275, Курсор 1276, Всплывающее меню 1276

Все Общие свойства индикатора 1255.

Все Общие свойства кругового индикатора 1266

Пользовательские свойства

Не определены.

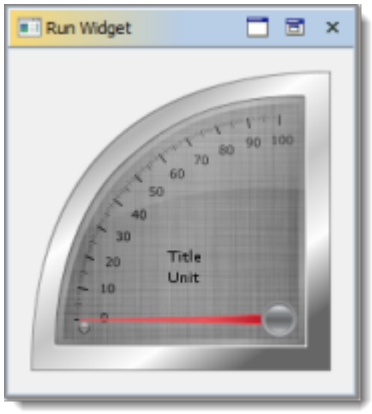
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.12.7 Четвертной индикатор

Этот компонент отображает легкоконфигурируемый четвертной индикатор.

Компонент "Четвертной индикатор" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Граница 1275, Курсор 1276, Всплывающее меню 1276

Все Общие свойства индикатора 1255.

Все Общие свойства кругового индикатора 1266.

Пользовательские свойства

Не определены.

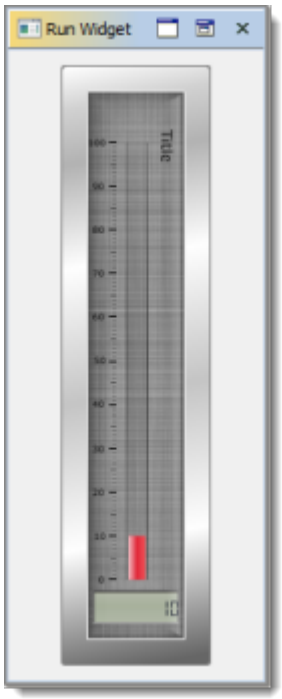
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.12.8 Линейный индикатор

Этот компонент отображает легкоконфигурируемый линейный индикатор.

Компонент "Линейный индикатор" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Граница 1275, Курсор 1276, Всплывающее меню 1276

Все Общие свойства индикатора 1255.

Пользовательские свойства

Не определены.

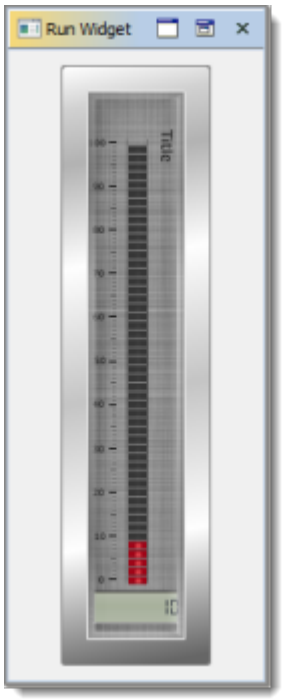
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.12.9 Линейный гистограммный индикатор

Этот компонент отображает легкоконфигурируемый линейный гистограммный индикатор.

Компонент "Линейный гистограммный индикатор" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Граница 1275, Курсор 1276, Всплывающее меню 1276

Все Общие свойства индикатора 1255.

Пользовательские свойства

Не определены.

Общие события

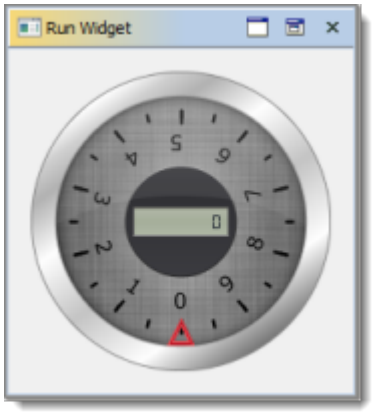
Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика

мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.12.10 Индикатор-счетчик

Этот компонент отображает легкоконфигурируемый индикатор-счетчик.

Компонент "Индикатор-счетчик" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Граница 1275, Курсор 1276, Всплывающее меню 1276

Все Общие свойства индикатора 1255.

Все Общие свойства кругового индикатора 1266.

Пользовательские свойства

Не определены.

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.12.11 Общие свойства индикатора

Эта статья описывает свойства, доступные для всех типов индикаторов.

Общие свойства

ВИДИМОСТЬ ФОНА

Определяет, видима и имеет ли цвет фоновая картинка.

Имя свойства: **backgroundVisible**

Тип свойства: **Boolean**

ЦВЕТ ФОНА

Цвет фона индикатора (одно из заранее определяемых значений).

Имя свойства: **backgroundColor**

Тип свойства: **String**

ПОЛЬЗОВАТЕЛЬСКИЙ ФОН

Пользовательский цвет[1279] рамки. Используется, если свойство **Цвет Фона** утсановлено на **Пользовательский**.

Имя свойства: **customBackground**

Тип свойства: **Data Table**

ВИДИМОСТЬ ПЕРЕДНЕГО ФОНА

Включает/Отключает видимость картинки переднего фона. Если включено, картинка переднего фона окрашивается.

Имя свойства: **foregroundVisible**

Тип свойства: **Boolean**

ОРИЕНТАЦИЯ

Ориентация индикатора.

Имя свойства: **orientation**

Тип свойства: **String**

СВЯЗАННЫЕ ЗНАЧЕНИЯ

Включает/Отключает автоматическую настройку значения светодиода индикатора, когда меняется основное значение индикатора.

Имя свойства: **valueCouples**

Тип свойства: **Boolean**

АВТОМАТИЧЕСКИЙ СБРОС НА НОЛЬ

Включает/Отключает режим, при котором индикатор возвращается к нулю после установки значения. То есть, если вы устанавливаете значение, указатель перемещается на это значение и после достижения данного значения взвращается обратно к нулю.

Имя свойства: **autoResetToZero**

Тип свойства: **Boolean**

СТАНДАРТНОЕ ВРЕМЯ УСТАНОВКИ СТРЕЛКИ НА ЗАДАННОЕ ЗНАЧЕНИЕ

Время в миллисекундах, которое необходимо указателю/столбцу/дисплею для перехода с минимального значения индикатора на максимальное в стандартном режиме. Минимальное время 250 мс, максимальное - 5000 мс.

Имя свойства: **stdTimeToValue**

Тип свойства: **Long**

ВОЗВРАТ ВРЕМЕНИ НА НОЛЬ

Время в миллисекундах, которое необходимо указателю/столбцу/дисплею для перехода от значения обратно к нулю при самовозврате к нулевому режиму. Минимальное время 250 мс, максимальное - 5000 мс.

Имя свойства: **rtzTimeBackToZero**

Тип свойства: **Long**

ВРЕМЯ УСТАНОВКИ СТРЕЛКИ НА ЗАДАННОЕ ЗНАЧЕНИЕ

Время в миллисекундах, которое необходимо указателю/столбцу/дисплею для перехода от минимального значения индикатора до максимального при самовозврате к нулевому режиму. Минимальное время 250 мс, максимальное - 5000 мс.

Имя свойства: **rtzTimeToValue**

Тип свойства: **Long**

МИНИМАЛЬНОЕ ВИДИМОЕ ИЗМЕРЯЕМОЕ ЗНАЧЕНИЕ

Видимость индикатора Минимального Измеряемого Значения.

Имя свойства: **minMeasuredValueVisible**

Тип свойства: **Boolean**

МАКСИМАЛЬНОЕ ВИДИМОЕ ИЗМЕРЯЕМОЕ ЗНАЧЕНИЕ

Видимость индикатора Максимального Измеряемого Значения.

Имя свойства: **maxMeasuredValueVisible**

Тип свойства: **Boolean**

ЗНАЧЕНИЕ

Значение, отображаемое на индикаторе. То же значение будет отображаться на светодиоде индикатора, если включена опция **Связывание значений**.

Имя свойства: **value**

Тип свойства: **Double**

МАКСИМАЛЬНОЕ ЗНАЧЕНИЕ

Устанавливает максимальное значение области значений индикатора. Это значение определяет максимальное значение, отображаемое индикатором.

Имя свойства: **maxValue**

Тип свойства: **Double**

МИНИМАЛЬНОЕ ЗНАЧЕНИЕ

Устанавливает минимальное значение области значений индикатора. Это значение определяет минимальное значение, отображаемое индикатором.

Имя свойства: **minValue**

Тип свойства: **Double**

Свойства рамки

Этот раздел описывает свойства рамки индикатора.

ВИДИМОСТЬ РАМКИ

Включает/Отключает видимость рамки.

Имя свойства: **frameVisible**

Тип свойства: **Boolean**

СТИЛЬ РАМКИ

Стиль рамки индикатора (одно из заранее определяемых значений). Стиль рамки - это своего рода схема цветов для рамки компонента.

Имя свойства: **frameDesign**

Тип свойства: **String**

ЭФФЕКТ РАМКИ

Псевдо 3D эффект рамки.

Имя свойства: **frameEffect**

Тип свойства: **String**

ВКЛЮЧЕНИЕ ОСНОВНОГО ЦВЕТА РАМКИ

Включает/Отключает использование **Основного Цвета Рамки** для окрашивания **Стиля Рамки** Блестящий Металл.

Имя свойства: **frameBaseColorEnabled**

Тип свойства: **Boolean**

ОСНОВНОЙ ЦВЕТ РАМКИ

Цвет, используемый для окрашивания **Стиля Рамки** Блестящий Металл.

Имя свойства: **frameBaseColor**

Тип свойства: **Color**

ПОЛЬЗОВАТЕЛЬСКИЙ СТИЛЬ РАМКИ

Используемый пользователем цвет1279, если свойство **Стиль Рамки** установлено на Пользовательское.

Имя свойства: **customFrameDesign**

Тип свойства: **Data Table**

Свойства подсветки

Этот раздел описывает свойства эффекта подсветки индикатора.

ВИДИМОСТЬ ПОДСВЕТКИ

Включает/Отключает индикатор подсветки.

Имя свойства: **glowVisible**

Тип свойства: **Boolean**

ПУЛЬСАЦИЯ ПОДСВЕТКИ

Включает/Отключает пульсацию эффекта подсветки.

Имя свойства: **glowPulsating**

Тип свойства: **Boolean**

ПОДСВЕТКА

Включает/Отключает подсвечивание индикатора подсветки.

Имя свойства: **glowing**

Тип свойства: **Boolean**

ЦВЕТ ПОДСВЕТКИ

Цвет, который будет использоваться для индикатора подсветки.

Имя свойства: **glowColor**

Тип свойства: **Color**

Свойства светодиода

Этот раздел описывает свойства светодиодных индикаторов. Каждый индикатор имеет до двух светодиодов: пороговый светодиод и светодиод, управляемый пользователем.

ВИДИМОСТЬ СВЕТОДИОДА

Включает/Отключает видимость порогового светодиода (одно из заранее определяемых значений).

Имя свойства: **ledVisible**

Тип свойства: **Boolean**

ЦВЕТ СВЕТОДИОДА

Цвет порогового светодиода.

Имя свойства: **ledColor**

Тип свойства: **String**

ВИДИМОСТЬ ПОЛЬЗОВАТЕЛЬСКОГО СВЕТОДИОДА

Контролирует видимость пользовательского светодиода.

Имя свойства: **userLedVisible**

Тип свойства: **Boolean**

ВКЛЮЧЕНИЕ ПОЛЬЗОВАТЕЛЬСКОГО СВЕТОДИОДА

Включает и выключает пользовательский светодиод.

Имя свойства: **userLedOn**

Тип свойства: **Boolean**

МЕРЦАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО СВЕТОДИОДА

Включает/Отключает мерцание пользовательского светодиода.

Имя свойства: **userLedBlinking**

Тип свойства: **Boolean**

ЦВЕТ ПОЛЬЗОВАТЕЛЬСКОГО СВЕТОДИОДА

Цвет пользовательского светодиода (одно из заранее определяемых значений).

Имя свойства: **userLedColor**

Тип свойства: **String**

ПОЗИЦИЯ СВЕТОДИОДА

Позиция порогового светодиода индикатора, определенная в виде значений X, Y. X и Y должны находиться в диапазоне 0...1, поскольку они определяют позицию светодиода в пределах индикатора.

Имя свойства: **ledPosition**

Тип свойства: **Data Table**

ПОЗИЦИЯ ПОЛЬЗОВАТЕЛЬСКОГО СВЕТОДИОДА

Позиция пользовательского светодиода индикатора, определенная в виде значений X, Y. X и Y должны находиться в диапазоне 0...1, поскольку они определяют позицию светодиода в пределах индикатора.

Имя свойства: **userLedPosition**

Тип свойства: **Data Table**

Свойства шкалы

Этот раздел описывает свойства шкалы индикатора.

ВИДИМОСТЬ ОТМЕТОК

Включает или отключает видимость отметок.

Имя свойства: **tickmarksVisible**

Тип свойства: **Boolean**

ВИДИМОСТЬ СЕКЦИИ ОТМЕТОК

Контролирует видимость секции отметок.

Имя свойства: **tickmarkSectionsVisible**

Тип свойства: **Boolean**

СЕКЦИИ ОТМЕТОК

Список секций отметок. Секции отпределяются стартовым значением, значением остановки, цветом и цветом выделения. Это свойство может оказаться полезным, если вам нужны четко определенные области, которые вы хотели бы визуализировать цветными отметками.

Имя свойства: **tickmarkSections**

Тип свойства: **Data Table**

ВКЛЮЧЕНИЕ ЦВЕТА ОТМЕТКИ ИЗ ТЕМЫ

Включает/Отключает использование отдельного цвета для отметок.

Имя свойства: **tickmarkColorFromThemeEnabled**

Тип свойства: **Boolean**

ЦВЕТ ОТМЕТКИ

Цвет отметок и их меток.

Имя свойства: **tickmarkColor**

Тип свойства: **Color**

ВИДМОСТЬ МЕТОК

Включает и отключает видимость меток отметок.

Имя свойства: **ticklabelsVisible**

Тип свойства: **Boolean**

ЧИСЛОВОЙ ФОРМАТ МЕТОК

Числовой формат, который используется для отображения меток отметок. Доступны следующие форматы: автоматический, стандартный, фракционный и научный.

Имя свойства: **labelNumberFormat**

Тип свойства: **String**

ВИДИМОСТЬ ДОПОЛНИТЕЛЬНЫХ ОТМЕТОК

Включает/Отключает видимость дополнительных отметок.

Имя свойства: **minorTickmarkVisible**

Тип свойства: **Boolean**

ТИП ДОПОЛНИТЕЛЬНЫХ ОТМЕТОК

Текущий тип отметок, используемый для дополнительных отметок. Значение может быть Линейным (по умолчанию), Круговым, Треугольным или Квадратным.

Имя свойства: **minorTickmarkType**

Тип свойства: **String**

РАССТОЯНИЕ МЕЖДУ ДОПОЛНИТЕЛЬНЫМИ ОТМЕТКАМИ

Расстояние между дополнительными отметками, если отключено свойство **Красивая Шкала**.

Имя свойства: **minorTickSpacing**

Тип свойства: **Double**

ВИДИМОСТЬ ОСНОВНЫХ ОТМЕТОК

Включает/Отключает видимость основных отметок.

Имя свойства: **majorTickmarkVisible**

Тип свойства: **Boolean**

ТИП ОСНОВНЫХ ОТМЕТОК

Текущий тип отметок, используемый для основных отметок. Значение может быть Линейным (по умолчанию), Круговым, Треугольным или Квадратным.

Имя свойства: **majorTickmarkType**

Тип свойства: **String**

РАССТОЯНИЕ МЕЖДУ ОСНОВНЫМИ ОТМЕТКАМИ

Расстояние между основными отметками, если отключено свойство **Красивая шкала.**

Имя свойства: **majorTickSpacing**

Тип свойства: **Double**

КРАСИВАЯ ШКАЛА

Включает/Отключает расчет "красивых" значений для минимальных и максимальных значений шкалы.

Имя свойства: **niceScale**

Тип свойства: **Boolean**

ЛОГАРИФМИЧЕСКАЯ ШКАЛА

Включает/Отключает логарифмическое масштабирование для оси.

Имя свойства: **logScale**

Тип свойства: **Boolean**

МАКСИМАЛЬНОЕ КОЛИЧЕСТВО ДОПОЛНИТЕЛЬНЫХ ОТМЕТОК

Максимальное количество дополнительных отметок.

Имя свойства: **maxNoOfMinorTicks**

Тип свойства: **Integer**

МАКСИМАЛЬНОЕ КОЛИЧЕСТВО ОСНОВНЫХ ОТМЕТОК

Максимальное количество основных отметок.

Имя свойства: **maxNoOfMajorTicks**

Тип свойства: **Integer**

Свойства дорожки

Дорожка - это область, которая определяется стартовым значением, значением середины и значением остановки. Эта область окрашивается градиентом, использующим два или три данных цвета. Например, критическая область для термометра находится между 30 и 100 градусами Цельсия и имеет градиент от зеленого к желтому и красному. В этом случае стартовым значением является 30, значением остановки - 100, а середина может быть где-то между 30 и 100 градусами.

Дорожка может использоваться в дополнение к Областям и Секциям индикатора.

ВИДИМОСТЬ ДОРОЖКИ

Контролирует видимость дорожки.

Имя свойства: **trackVisible**

Тип свойства: **Boolean**

НАЧАЛО ДОРОЖКИ

Значение, где начинается дорожка.

Имя свойства: **trackStart**

Тип свойства: **Double**

ЦВЕТ НАЧАЛА ДОРОЖКИ

Цвет точки, где начинается дорожка.

Имя свойства: **trackStartColor**

Тип свойства: **Color**

СЕРЕДИНА ДОРОЖКИ

Значение средней точки между **Началом Дорожки** и **Концом Дорожки**.

Имя свойства: **trackSection**

Тип свойства: **Double**

ЦВЕТ СЕРЕДИНЫ ДОРОЖКИ

Цвет (средней) точки середины дорожки.

Имя свойства: **trackSectionColor**

Тип свойства: **Color**

КОНЕЦ ДОРОЖКИ

Значение, которым заканчивается дорожка.

Имя свойства: **trackStop**

Тип свойства: **Double**

ЦВЕТ КОНЦА ДОРОЖКИ

Цвет точки, где заканчивается дорожка.

Имя свойства: **trackStopColor**

Тип свойства: **Color**

Свойства заголовка

Это раздел описывает свойства заголовка и меток единиц индикатора.

ВКЛЮЧЕНИЕ ЗАГОЛОВКА И ШРИФТА ЕДИНИЦЫ ИЗМЕРЕНИЯ

Включает и выключает использование пользовательского заголовка и шрифта меток единицы измерения.

Имя свойства: **titleAndUnitFontEnabled**

Тип свойства: **Boolean**

ЗАГОЛОВОК

Заголовок индикатора, например, "Температура".

Имя свойства: **title**

Тип свойства: **String**

ЗАГОЛОВОК И ШРИФТ ЕДИНИЦЫ ИЗМЕРЕНИЯ

Шриф1278, используемый для заголовка и меток единиц измерения.

Имя свойства: **titleAndUnitFont**

Тип свойства: **Data Table**

СТРОКА ЕДИНИЦЫ ИЗМЕРЕНИЯ

Строка единицы измерения индикатора, например `[cm]`.

Имя свойства: **unitString**

Тип свойства: **String**

ВКЛЮЧИТЬ ЦВЕТ МЕТОК ИЗ ТЕМЫ

Включает/Отключает использование отдельного цвета для заголовка и меток единиц измерения.

Имя свойства: **labelColorFromThemeEnabled**

Тип свойства: **Boolean**

ЦВЕТ МЕТКИ

Цвет заголовка и меток единиц измерения.

Имя свойства: **labelColor**

Тип свойства: **Color**

Свойства порога

Этот раздел описывает свойства порога индикатора.

ВИДИМОСТЬ ПОРОГА

Контролирует видимость индикатора порога.

Имя свойства: **thresholdVisible**

Тип свойства: **Boolean**

ПОРОГ

Значение, при котором отображается индикатор порога.

Имя свойства: **threshold**

Тип свойства: **Double**

ТИП ПОРОГА

Тип индикатора порога (Стрелка или Треугольник).

Имя свойства: **thresholdType**

Тип свойства: **String**

ЦВЕТ ПОРОГА

Цвет индикатора порога (одно из заранее определяемых значений).

Имя свойства: **thresholdColor**

Тип свойства: **String**

ПЕРЕВЕРНУТОЕ ПОВЕДЕНИЕ ПОРОГОВ

Включает/Отключает перевернутое поведение порогов.

Имя свойства: **thresholdBehaviourInverted**

Тип свойства: **Boolean**

Свойства области

Области - это видимые секции индикатора, которые визуализируют определенный диапазон значений. Области могут использоваться в дополнение к Дорожкам и Секциям индикатора.

ВИДИМОСТЬ ОБЛАСТЕЙ

Контролирует видимость областей.

Имя свойства: **areasVisible**

Тип свойства: **Boolean**

ОБЛАСТИ

Список областей. Каждая область определяется стартовым значением, значением остановки и цветом.

Имя свойства: **areas**

Тип свойства: **Data Table**

ОБЛАСТЬ ВЫДЕЛЕНИЯ

Включает/Отключает подсветку области, которая содержит текущее значение.

Имя свойства: **highlightArea**

Тип свойства: **Boolean**

ВИДИМОСТЬ ОБЛАСТЕЙ С 3D ЭФФЕКТОМ

Контролирует внешний вид областей с 3D эффектом.

Имя свойства: **areas3DEffectVisible**

Тип свойства: **Boolean**

ВКЛЮЧЕНИЕ ПРОЗРАЧНЫХ ОБЛАСТЕЙ

Включает/Отключает использование прозрачного цвета для заливки областей.

Имя свойства: **transparentAreasEnabled**

Тип свойства: **Boolean**

Свойства секций

Секции - это окрашенные цветом части шкалы индикатора. Секции могут использоваться в дополнение к Дорожке и Областям индикатора.

ВИДИМОСТЬ СЕКЦИЙ

Контролирует видимость секций.

Имя свойства: **sectionsVisible**

Тип свойства: **Boolean**

СЕКЦИИ

Список секций. Каждая секция определяется стартовым значением, значением остановки, цветом и цветом выделения.

Имя свойства: **sections**

Тип свойства: **Data Table**

ТОЛЬКО ОТМЕТКИ СЕКЦИЙ

Контролирует, видны ли только отметки секции.

Имя свойства: **sectionTickmarksOnly**

Тип свойства: **Boolean**

СЕКЦИЯ ВЫДЕЛЕНИЯ

Включает/Отключает выделение секции, содержащей текущее значение.

Имя свойства: **highlightSection**

Тип свойства: **Boolean**

ВИДИМОСТЬ СЕКЦИЙ С 3D ЭФФЕКТОМ

Контролирует внешний вид секции с 3D эффектом.

Имя свойства: **section3DEffectVisible**

Тип свойства: **Boolean**

ВКЛЮЧЕНИЕ ПРОЗРАЧНЫХ СЕКЦИЙ

Включает/Отключает использование прозрачного цвета для заливки секций.

Имя свойства: **transparentSectionsEnabled**

Тип свойства: **Boolean**

Свойства ЖК дисплея

Каждый индикатор может отображать дисплей типа ЖК с числовой презентацией значения индикатора. Этот раздел описывает свойства ЖК дисплея.

ВИДИМОСТЬ ЖК ДИСПЛЕЯ

Включает или отключает видимость ЖК дисплея.

Имя свойства: **lcdVisible**

Тип свойства: **Boolean**

ЗНАЧЕНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ЖК

Если отключен флажок индикатора **Связывание значений**, ЖК дисплей будет отображать отдельное числовое значение, определенное этим свойством.

Имя свойства: **lcdValue**

Тип свойства: **Double**

ЦВЕТ

Цвет фона ЖК экрана.

Имя свойства: **lcdColor**

Тип свойства: **String**

ДЕСЯТИЧНЫЕ

Количество десятичных знаков, используемых для визуализации значений на ЖК экране.

Имя свойства: **lcdDecimals**

Тип свойства: **Integer**

ИНФОРМАЦИОННАЯ СТРОКА

Произвольный текст, отображаемый в верхней части ЖК экрана.

Имя свойства: **lcdInfoString**

Тип свойства: **String**

ИНФОРМАЦИОННЫЙ ШРИФТ

Шрифт1278, используемый для отображения **Информационной Строки**.

Имя свойства: **lcdInfoFont**

Тип свойства: **Data Table**

ЧИСЛОВАЯ СИСТЕМА

Числовая система, которая будет использоваться для отображения текущего ЖК значения (десятичное, шестнадцатиричное или восьмеричное).

Имя свойства: **lcdNumberSystem**

Тип свойства: **String**

НАУЧНЫЙ ФОРМАТ

Включает/Отключает научный формат ЖК значения.

Имя свойства: **lcdScientificFormat**

Тип свойства: **Boolean**

ВИДИМОСТЬ ПОРОГА

Включает/Отключает видимость индикатора порога ЖК.

Имя свойства: **lcdThresholdVisible**

Тип свойства: **Boolean**

ПОРОГ

Значение порога ЖК индикатора.

Имя свойства: **lcdThreshold**

Тип свойства: **Double**

ПЕРЕВЕРНУТОЕ ПОВЕДЕНИЕ ПОРОГА

Включает/Отключает перевернутое поведение ЖК порога.

Имя свойства: **lcdthresholdBehaviourInverted**

Тип свойства: **Boolean**

ВИДИМОСТЬ СТРОКИ ЕДИНИЦЫ ИЗМЕРЕНИЯ

Включает/Отключает видимость строки единицы измерения на ЖК дисплее.

Имя свойства: **lcdUnitStringVisible**

Тип свойства: **Boolean**

ШРИФТ ЗНАЧЕНИЯ

Шрифт1278, используемый для визуализации значения на ЖК дисплее.

Имя свойства: **lcdValueFont**

Тип свойства: **Data Table**

СТРОКА ЕДИНИЦЫ ИЗМЕРЕНИЯ

Текст метки единицы измерения, отображаемый на ЖК дисплее.

Имя свойства: **lcdUnitString**

Тип свойства: **String**

ШРИФТ ЕДИНИЦЫ ИЗМЕРЕНИЯ

Шрифт1278, используемый для визуализации метки единицы измерения на ЖК дисплее.

Имя свойства: **lcdUnitFont**

Тип свойства: **Data Table**

ЦИФРОВОЙ ШРИФТ

Включает/Отключает использование цифрового шрифта на ЖК дисплее.

Имя свойства: **digitalFont**

Тип свойства: **Boolean**

13.4.12.11.1 Общие свойства кругового индикатора

Данная статья описывает свойства, доступные для всех индикаторов кругового типа.

ЦВЕТ УКАЗАТЕЛЯ

Цвет указателя.

Имя свойства: **pointerColor**

Тип свойства: **String**

ПОЛЬЗОВАТЕЛЬСКИЙ ЦВЕТ УКАЗАТЕЛЯ

Цвет, по которому определяется пользовательский цвет указателя.

Имя свойства: **customPointerColor**

Тип свойства: **Color**

ТИП УКАЗАТЕЛЯ

Тип указателя (одно из заранее определяемых значений).

Имя свойства: **pointerType**

Тип свойства: **String**

ТИП ПЕРЕДНЕГО ФОНА

Тип переднего фона, используемый для кругового индикатора (одно из заранее определяемых значений).

Имя свойства: **foregroundType**

Тип свойства: **String**

ТИП ИНДИКАТОРА

Тип индикатора:

- Тип 1 - индикатор в одну четверть (90 градусов)
- Тип 2 - индикатор в две четверти (180 градусов)
- Тип 3 - индикатор в три четверти (270 градусов)
- Тип 4 - индикатор в четыре четверти (300 градусов)

Имя свойства: **gaugeType**

Тип свойства: **String**

ПОКАЗАТЬ ТЕНЬ ОТ УКАЗАТЕЛЯ

Включает/Отключает тень указателя.

Имя свойства: **pointerShadowVisible**

Тип свойства: **Boolean**

ПОКАЗАТЬ ДИАПАЗОН ИЗМЕРЯЕМЫХ ЗНАЧЕНИЙ

Включает/Отключает видимость дуги, представляющей диапазон измеряемых значений.

Имя свойства: **rangeOfMeasuredValuesVisible**

Тип свойства: **Boolean**

Свойства кнопки

Этот раздел описывает свойства центральной кнопки кругового индикатора.

ТИП КНОПКИ

Тип кнопки (одно из заранее определяемых значений).

Имя свойства: **knowType**

Тип свойства: **String**

СТИЛЬ КНОПКИ

Стиль кнопки (одно из заранее определяемых значений).

Имя свойства: **knowStyle**

Тип свойства: **String**

Свойства рамки

ТИП РАМКИ

Определяет тип рамки, используемой для кругового индикатора. Может быть круглый или квадратный.

Имя свойства: **frameType**

Тип свойства: **String**

Свойства шкалы

НАПРАВЛЕНИЕ

Направление отсчета меток отметок (по часовой или против часовой стрелки).

Имя свойства: **tickmarkDirection**

Тип свойства: **String**

ОРИЕНТАЦИЯ

Ориентация метко отметок (Обычная, Горизонтальная или Тангенсная).

Имя свойства: **ticklabelOrientation**

Тип свойства: **String**

Свойства секции

ВКЛЮЧЕНИЕ РАСШИРЕННЫХ СЕКЦИЙ

Включает/Отключает использование более широких секций.

Имя свойства: **expandedSectionsEnabled**

Тип свойства: **Boolean**

13.4.13 Графические примитивы

Графические примитивы - это простые компоненты виджетов, такие как круг или многоугольник. Примитивы очень похожи на регулярные компоненты 955, хотя создаются они особенным образом ("отрисовываются").

Поддерживаются следующие типы графических примитивов:

 Ломаная линия

 Многоугольник

 Прямоугольник

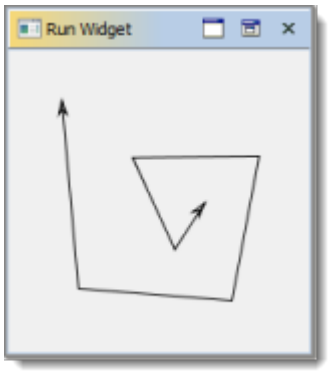
 Эллипс

 Стрелка

13.4.13.1 Ломаная линия

Этот компонент отображает линию, состоящую из одного и более прямых отрезков, корторые могут иметь стрелку(и) на конечных отрезках.

Компонент "Ломаная линия" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Передний фон 1275, Граница 1275, Шрифт 1275, Курсор 1276, Всплывающая подсказка 1276, Всплывающее меню 1276

Пользовательские свойства

ШТРИХ

Штрих1282, используемый для отрисовки ломаной линии.

Имя свойства: **stroke**

Тип свойства: **Data Table**

ТОЧКИ

Таблица точек привязки ломаной линии, каждая из которых характеризуется координатами (X, Y) в пределах внутреннего пространства компонента "Ломаная линия". Координаты варьируются от 0.0 до 1.0.

Имя свойства: **points**

Тип свойства: **Data Table**

СТРЕЛКА НАЧАЛА

Определяет, изображается ли стрелка начала.

Имя свойства: **startArrow**

Тип свойства: **Boolean**

СТРЕЛКА КОНЦА

Определяет, изображается ли стрелка конца.

Имя свойства: **endError**

Тип свойства: **Boolean**

ВНЕШНИЙ РАДИУС

Определяет внешний радиус стрелок начала и конца.

Имя свойства: **outerRadius**

Тип свойства: **Float**

ВНУТРЕННИЙ РАДИУС

Определяет внутренний радиус стрелок начала и конца.

Имя свойства: **innerRadius**

Тип свойства: **Float**

УГОЛ

Определяет угол стрелок начала и конца.

Имя свойства: **angle**

Тип свойства: **Float**

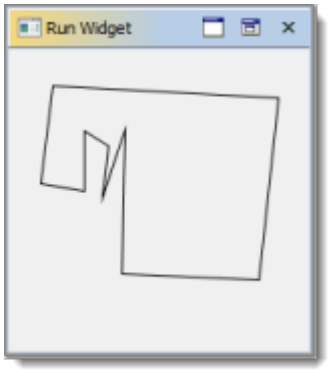
Общие события

Скрытие1285, Показ1285, Перемещение1285, Изменение размеров1285, Клик мыши1286, Нажатие кнопки мыши1286, Отпускание кнопки мыши1286, Вход мыши1286, Выход мыши1286, Перемещение мыши1286, Вращение колесика мыши1286, Печать клавиши1287, Нажатие клавиши1287, Отпускание клавиши1287, Получение фокуса1287, Потеря фокуса1287

13.4.13.2 Многоугольник

Этот компонент отображает замкнутый и, возможно, закрашенный многоугольник, состоящий из одного и более прямых отрезков.

Компонент "Многоугольник" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Передний фон 1275, Фон 1275, Непрозрачный 1275, Граница 1275, Шрифт 1275, Курсор 1276, Всплывающая подсказка 1276, Всплывающее меню 1276

Пользовательские свойства

ШТРИХ

Штрих 1282, используемый для отрисовки многоугольника.

Имя свойства: **stroke**

Тип свойства: **Data Table**

ТОЧКИ

Таблица точек привязки ломаной линии, каждая из которых характеризуется координатами (X, Y) в пределах внутреннего пространства компонента "Многоугольник". Координаты варьируются от 0.0 до 1.0.

Имя свойства: **points**

Тип свойства: **Data Table**

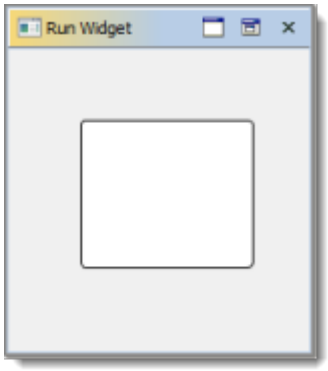
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.13.3 Прямоугольник

Этот компонент отображает прямоугольник, который может быть закрашенным.

Компонент "Прямоугольник" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Передний фон 1275, Фон 1275, Непрозрачный 1275, Граница 1275, Шрифт 1275, Курсор 1276, Всплывающая подсказка 1276, Всплывающее меню 1276

Пользовательские свойства

ШТРИХ

Штрих 1282, используемый для отрисовки прямоугольника.

Имя свойства: **stroke**

Тип свойства: **Data Table**

ШИРИНА ДУГИ

Ширина угловой дуги прямоугольника.

Имя свойства: **arcWidth**

Тип свойства: **Float**

ВЫСОТА ДУГИ

Высота угловой дуги прямоугольника.

Имя свойства: **arcHeight**

Тип свойства: **Float**

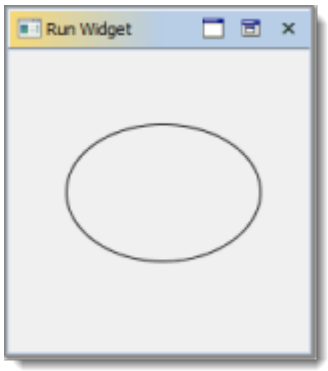
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.13.4 Эллипс

Этот компонент отображает эллипс или круг, который может быть закрашенным.

Компонент "Эллипс" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Передний фон 1275, Фон 1275, Непрозрачный 1275, Граница 1275, Шрифт 1275, Курсор 1276, Всплывающая подсказка 1276, Всплывающее меню 1276

Пользовательские свойства

ШТРИХ

Штрих 1282, используемый для отрисовки эллипса.

Имя свойства: **stroke**

Тип свойства: **Data Table**

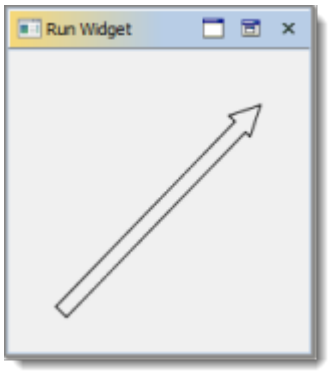
Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.13.5 Стрелка

Этот компонент отображает стрелку, которая может быть закрашенной.

Компонент "Стрелка" выглядит следующим образом:



Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Передний фон 1275, Фон 1275, Непрозрачный 1275, Граница 1275, Шрифт 1275, Курсор 1276, Всплывающая подсказка 1276, Всплывающее меню 1276

Пользовательские свойства

ШТРИХ

Штрих 1282, используемый для отрисовки стрелки.

Имя свойства: **stroke**

Тип свойства: **Data Table**

ШИРИНА СТРЕЛКИ

Определяет ширину стрелки.

Имя свойства: **arrowWidth**

Тип свойства: **Float**

ДЛИНА КОНЧИКА

Определяет длину кончика стрелки.

Имя свойства: **tipLength**

Тип свойства: **Float**

ШИРИНА КОНЧИКА

Определяет ширину кончика стрелки.

Имя свойства: **tipWidth**

Тип свойства: **Float**

Общие события

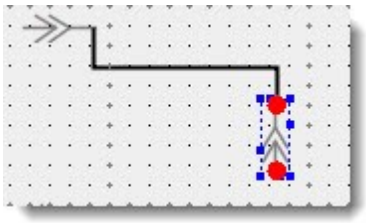
Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1286, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.14 Коннекторы

Коннекторы служат для соединения компонентов на рабочей панели. Связь является отдельным компонентом и отображается в дереве компонентов как "Коннектор".

После построения коннектора между двумя компонентами, коннектор останется неразрывным между ними до тех пор пока:

1. Будет удалена сама связь;
2. Будет удален один из связанных компонентов;
3. Будет удалена точка прикрепления 1284 коннектора;



Построение

После активации точки прикрепления на любом компоненте начинается построение коннектора. Установка промежуточной точки происходит при нажатии ЛКМ.

Если установленная точка входит в окрестность точки прикрепления 1284 другого компонента, то будет построен коннектор. Построение коннектора происходит по следующим правилам:

1. Если при построении линии не было задано ни одной промежуточной точки, коннектор будет построен по кратчайшему пути.
2. Если при построении коннектора были заданы промежуточные точки, коннектор будет построен по точкам. Точки можно изменять, корректируя коннектор.

Общие свойства

Ширина 1274, Высота 1274, Привязки 1275, Активный 1275, Видимый 1275, Основной цвет 1275, Рамка 1275, Шрифт 1275, Курсор 1276, Всплывающая подсказка 1276, Всплывающее меню 1276

Пользовательские свойства

Включает в себя пользовательские свойства компонента **Ломаная линия** 1268.

ПУТЬ

Задает режим постороения коннектора. Есть три режима: ломаная линия, ортогональный режим и кубическая кривая Безье.

Имя свойства: **path**

Тип свойства: **Целое**

ИМЯ НАЧАЛЬНОГО КОМПОНЕНТА

Задает начальный компонент к которому прикрепляется коннектор.

Имя свойства: **startComponentName**

Тип свойства: **Строка**

ИМЯ КОНЕЧНОГО КОМПОНЕНТА

Задает конечный компонент к которому прикрепляется коннектор.

Имя свойства: **endComponentName**

Тип свойства: **Строка**

ИМЯ НАЧАЛЬНОЙ ТОЧКИ ПРИКРЕПЛЕНИЯ

Задает начальную точку прикрепления 1274 к которому прикрепляется коннектор.

Имя свойства: **startPinName**

Тип свойства: **Строка**

ИМЯ КОНЕЧНОЙ ТОЧКИ ПРИКРЕПЛЕНИЯ

Задает конечную точку прикрепления 1274 к которому прикрепляется коннектор.

Имя свойства: **endPinName**

Тип свойства: **Строка**

Общие события

Скрытие 1285, Показ 1285, Перемещение 1285, Изменение размеров 1285, Клик мыши 1286, Нажатие кнопки мыши 1286, Отпускание кнопки мыши 1286, Вход мыши 1286, Выход мыши 1286, Перемещение мыши 1286, Вращение колесика мыши 1288, Печать клавиши 1287, Нажатие клавиши 1287, Отпускание клавиши 1287, Получение фокуса 1287, Потеря фокуса 1287

13.4.15 Свойства компонента

Каждый компонент обладает несколькими свойствами. Ссылка на них могут осуществляться из привязок 1295, которые "направляют" данные к и от устройства (например, настройки аппаратных устройств). Некторые свойства являются общими 1274 для всех компонентов виджета. Другие касаются отдельного компонента и описаны в статьях, посвященных данным компонентам.

Типы свойств

Описание каждого свойства компонента включает *тип* свойства. Тип свойства представляет собой тип значения, полученного из ссылки на свойство компонента, которое отображено в привязке виджета (строка, логическое, целое и т.д.).

Индексированные свойства

Индексированное свойство является свойством с несколькими элементами ,т.е. представляемое массивом или списком. Ссылки на элементы индексированного свойства должны осуществляться из привязок 1295 методом добавления отсчитываемого от нуля **индекса** в цель 1295 привязки.

Свойство по умолчанию

Свойство по умолчанию - свойство компонента, которое автоматически является целью создаваемой привязки 1295. Например, при перетаскивании какой-либо переменной сервера в метку 956, привязка, созданная в результате данного действия, изменяет текст метки, т.к. **Текст** является свойством по умолчанию компонента "Метка".

Общие свойства компонентов

Данный раздел описывает свойства, поддерживаемые большинством компонентов виджета. Отдельное описание каждого компонента включает в себя список поддерживаемых общих свойств.

ШИРИНА

Оптимальная ширина компонента в пикселях. Если для отображения компонента недостаточно свободного пространства, данную настройку можно игнорировать и актуальная ширина компонента будет меньше. Если значение данного свойства установлено на ноль, ширина компонента рассчитывается автоматически в зависимости от его содержания и доступной области изображения в виджете.

Имя свойства: **width**

Тип свойства: **Целое**

ВЫСОТА

Оптимальная высота компонента в пикселях. Если для отображения компонента недостаточно свободного пространства, данную настройку можно игнорировать и актуальная высота компонента будет меньше. Если значение данного свойства установлено на ноль, высота компонента рассчитывается автоматически в зависимости от его содержания и доступной области изображения в макете виджета.

Имя свойства: **height**

Тип свойства: **Целое**

ПРИВЯЗКИ

Данное свойство содержит таблицу всех привязок[1295], относящихся к данному компоненту (т.е. привязки с выражением[1296], целью[1295] или активатором[1296], ссылающимися на свойства компонента).

Имя свойства: **bindings**

Тип свойства: **Таблица данных**

АКТИВЕН

Данный флажок указывает, что компонент активен. Отключенные компоненты не отвечают на вход пользователя и в большинстве интерфейсов выделены серым цветом (недоступны для выбора).

Имя свойства: **enabled**

Тип свойства: **Логическое**

ВИДИМЫЙ

Данный флажок указывает, что компонент является видимым.

Имя свойства: **visible**

Тип свойства: **Логическое**

ПЕРЕДНИЙ ПЛАН

Данное свойство определяет цвет переднего плана компонента. Цветом переднего плана выделены элементы, относящиеся к компоненту.

Имя свойства: **foreground**

Тип свойства: **Цвет**

ФОН

Данное свойство определяет цвет фон компонента. Цветом фона выделены элементы, относящиеся к компоненту.

Имя свойства: **background**

Тип свойства: **Цвет**

НЕПРОЗРАЧНЫЙ

Определяет непрозрачность компонента. Является логическим значением (Boolean), может быть "True" (непрозрачный) и "False" (прозрачный). Фон непрозрачного компонента не отображается -- свойство цвет фона[1275] игнорируется.

Имя свойства: **opaque**

Тип свойства: **Логическое**

ГРАНИЦА

Более подробную информацию о настройках границ см. в статье Граница[1277].

Имя свойства: **border**

Тип свойства: **Таблица данных**

ШРИФТ

Определяет шрифт, используемый в компоненте. Данное свойство относится к компонентам с метками или другими текстовыми элементами (например, Текстовое поле 958). Свойства шрифта описаны здесь 1278.

Имя свойства: **font**

Тип свойства: **Таблица данных**

КУРСОР

Определяет, какой курсор мыши используется, когда он наводится на компонент.

Доступные курсоры:

- По умолчанию
- Перекрестие
- Текст
- На паузе
- Изменение размера (N, S, E, W, NE, NW, SE, SW)
- В виде руки
- Перемещение

Имя свойства: **cursor**

Тип свойства: **Целое**

Обратите внимание, что Web UI поддерживает только курсор по умолчанию.

ВСПЛЫВАЮЩАЯ ПОДСКАЗКА

Текст всплывающей подсказки компонента. Подсказки обычно появляются при наведении курсора мыши на компонент:

Имя свойства: **tooltip**

Тип свойства: **Строка**

ФОКУСИРУЕМЫЙ

Флажок определяет, может ли компонент получить фокус ввода с клавиатуры.

Имя свойства: **focusable**

Тип свойства: **Логическое**

ВСПЛЫВАЮЩЕЕ МЕНЮ

Данное свойство настраивает контекстное меню компонента, который отображается при нажатии на компоненте правой кнопкой мыши. Таблица элементов меню содержит следующие поля:

Поле	Имя	Тип	Описание
Имя	name	строка	Название элемента меню, на который происходит ссылка из активатора привязки 1296.
Описание	description	строка	Текстовое описание элемента, т.е. текст, появляющийся в меню.
Пиктограмма	icon	блок данных	Пиктограмма элемента.
Условие	condition	выражение	Если данное выражение условия определено и возвращает false, элемент всплывающего меню будет пропущен.

Имя свойства: **popupMenu**

Тип свойства: **Таблица данных**

13.4.15.1 Пользовательские свойства

Помимо станартных свойств компонента, можно определить собственные **пользовательские свойства**. Данные свойства могут использоваться для сохранения внутреннего состояния виджета, например, промежуточного результата какого-либо расчета.

Пользовательские свойства можно добавить к любому компоненту виджета, однако, они не будут влиять на поведение или визуальное представление компонента. Вместо этого, на пользовательские свойства могут ссылаться привязки виджета 1295.

Управление пользовательскими свойствами

Управление пользовательскими свойствами осуществляется через контекстное меню диалогового окна "Свойства компонента" 431 в GUI редакторе 423. В данном контекстном меню вы увидите три соответствующих пункта:

- **Добавить пользовательское свойство**. Добавляет новое свойство для компонента, свойства которого изменяются.
- **Редактировать пользовательское свойство.** Позволяет изменить параметры выбранного свойства.
- **Удалить пользовательское свойство**. Удаляет определение пользовательского свойства из компонента.

Параметры пользовательского свойства

Параметры пользовательского свойства могут быть определены во время его создания и отредактированы позже. Пользовательское свойство по сути представляет собой таблицу данных 49.

Имя свойства используется для ссылки на него из привязок виджета 1295.

Другие параметры пользовательского свойства описаны в разделе Формат таблицы данных 49.

13.4.15.2 Граница

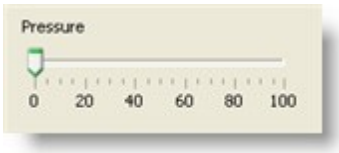
Каждый компонент имеет **внутреннюю** и **внешнюю** границу. Свойства определяются для каждой в отдельности.

Свойства границы

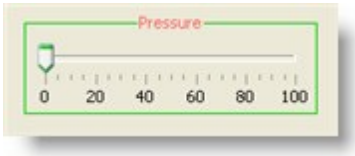
Поле	Имя	Тип	Описание
Позиция	pos	строка	**Внутренний** или **внешний**, доступно только для чтения.
Тип	type	строка	**Нет**, **Пустой**, **Линия**, **Заглубленный**, **Приподнятый** или **Вдавленный**.
Сверху	top	целое	Толщина границы верхнего края компонента. Применимо к "**Пустой**" и "**Линия**" рамкам.
Слева	left	целое	Толщина границы левого края компонента. Применимо к "**Пустой**" и "**Линия**" рамкам.
Снизу	bottom	целое	Толщина границы нижнего края компонента. Применимо к "**Пустой**" и "**Линия**" рамкам.
Справа	right	целое	Толщина границы правого края компонента. Применимо к "**Пустой**" и "**Линия**" рамкам.
Основной цвет	color	цвет	Цвет рамки или цвет основной части для двухцветных рамок. Применимо к "**Линия**", "**Заглубленный**", "**Приподнятый**" и **"Рельефной"** рамкам.
Дополнительный цвет	scolor	цвет	Цвет тени рамки или другого элемента декора. Применимо к **Вдавленной**", "**Приподнятый**" и **"Вдавленный"** рамкам.
Заголовок	title	строка	Текст заголовка рамки.
Цвет заголовка	tcolor	цвет	Цвет текста заголовка рамки
Выравнивание заголовка	justification	целое	Выравнивание заголовка рамки - **Слева**, **По центру** или **Справа**.

Примеры границ

Пустая граница невидима, но может быть использована для добавления заголовка к элементу:

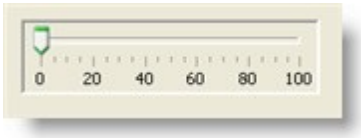


Зеленая граница с заголовком красного цвета по центру:

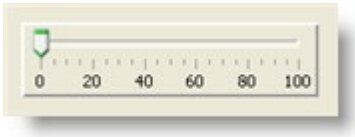


Синяя граница с различной толщиной линий с каждой стороны:

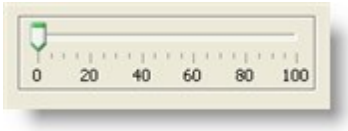
Вдавленная граница:



Выпуклая граница:



Рельефная граница:



13.4.15.3 Шрифт

Данный раздел описывает свойства шрифта. Для отображения текстовых строк компоненты виджета используют несколько шрифтов.

Свойства шрифта

Поле	Имя	Тип	Описание
Использовать особый шрифт	custom	логическое	Включите данный параметр, чтобы использовать пользовательский шрифт в компоненте.
Имя	name	строка	Имя используемого шрифта. Если вы выбираете пользовательский шрифт, вполне возможно, что он будет недоступен для всех клиентов, запускающих данный виджет. В таком случае будет использоваться шрифт по умолчанию. Используйте известные шрифты, такие как Verdana, Times New Roman и т.д.

Размер	size	целое	Размер шрифта в точках.
Полужирный	bold	логическое	**Жирный** шрифт.
Курсив	italic	логическое	Шрифт *курсив*.

13.4.15.4 Заливка

Цвет используется в компонентах виджета для заливки различных областей, начертания линий или других форм. Существуют четыре основных типа заливки:

- Цвет 1279
- Линейный градиент 1279
- Радиальный градиент 1280
- Текстура

Цвет

При таком типе заливки используется один цвет.

Линейный градиент

Линейный градиент определяет способ заливки форм, где цвет плавно переходит в другой слева направо. Пользователь может задать два и более цветов градиента, данная заливка обеспечит интерполяцию между каждым цветом. Пользователь также определяет начальные и конечные точки, которые указывают, где должен начаться и закончиться градиент в пользовательском пространстве.

Пользователь должен представить массив чисел с плавающей запятой, определяя тем самым распределение цветов по градиенту. Данные значения должны быть в пределе от 0.0 до 1.0 и действовать подобно ключевому кадру в градиенте (они отмечают, где именно градиент должен быть определенного цвета).

В случае, если пользователь не задал значение первого ключевого кадра на 0 и/или последнего ключевого кадра на 1, ключевые кадры будут созданы на данных позициях, а первый и последний цвета будут реплицированы. Поэтому, если пользователь вводит следующие значения для формирования градиента:

Относительное положение	**Цвет**
0.3	Синий
0.7	Красный

это будет конвертировано в следующие ключевые кадры:

Относительное положение	**Цвет**
0.0	Синий
0.3	Синий
0.7	Красный
1.0	Красный

Пользователь может также выбрать действие, которое должен выполнять градиент при заливке за пределами начальной и конечной точек. Если **Цикличный метод** не определен, по умолчанию будет выбрано свойство **Отключен**, которое означает, что цвет конечного ключевого кадра будет использован для заливки оставшейся области.

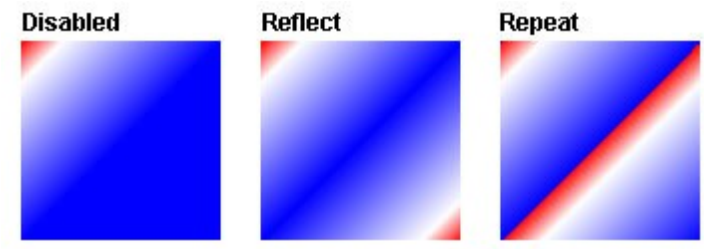
Изображение, следующее далее, представляет каждый из трех цикличных методов, применимых к линейному градиенту со следующими параметрами:

Start X, Start Y: 0.0

End X, End Y: 50.0

Цвета:

Относительное положение	Цвет
0.0	Красный
0.2	Белый
1.0	Синий



Радиальный градиент

Радиальный градиент представляет собой способ заливки формы, при котором цвет плавно переходит в другой из центра к краям. Пользователь может назначить 2 и более цветов градиента, и данная заливка обеспечит интерполяцию между каждым цветом.

Пользователь должен задать круговое контролирование шаблона градиента, которое определяется центральной точкой и радиусом. Пользователь может также назначить отдельный фокус в данном кругу, который будет контролировать положение первого цвета градиента. По умолчанию фокус установлен на центр круга.

Данная заливка назначит первый цвет градиента для фокуса и последний цвет внешней границы круга, плавная интерполяция между любыми цветами определяется пользователем. Любая линия от фокуса к окружности заполнит промежутки между всеми цветами градиента.

Определение фокусной точки вне радиуса круга приведет к тому, что фокус будет установлен в точке пересечения линии центра фокуса и внешней границы круга.

Пользователь обязан предоставить массив чисел с плавающей запятой для распределения цветов по градиенту. Данные значения должны быть в пределе от 0.0 до 1.0 и действовать подобно ключевому кадру в градиенте (они отмечают, где именно градиент должен быть определенного цвета).

В случае, если пользователь не задал значение первого ключевого кадра на 0 и/или последнего ключевого кадра на 1, ключевые кадры будут созданы на данных позициях, а первый и последний цвета будут реплицированны. Поэтому, если пользователь вводит следующие значения для формирования градиента:

Относительное положение	Цвет
0.3	Синий
0.7	Красный

это будет конвертировано в следующие ключевые кадры:

Относительное положение	Цвет
0.0	Синий
0.3	Синий
0.7	Красный
1.0	Красный

Пользователь может также выбрать действие, которое должен выполнять градиент при заливке за пределами границ радиуса круга. Если **Цикличный метод** не определен, по умолчанию будет выбрано свойство **Отключен**, которое означает, что цвет конечного ключевого кадра будет использован для заливки оставшейся области.

Изображение, следующее далее, представляет каждый из трех цикличных методов, применимых к радиальному градиенту со следующими параметрами:

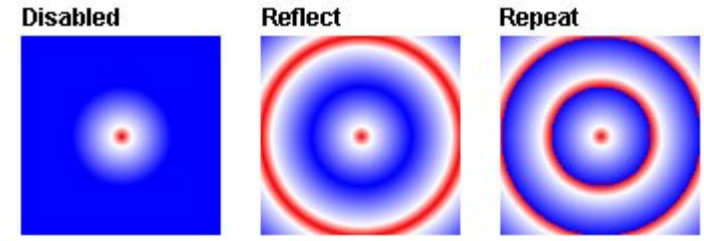
Start X, Start Y: 50.0

Radius: 25.0

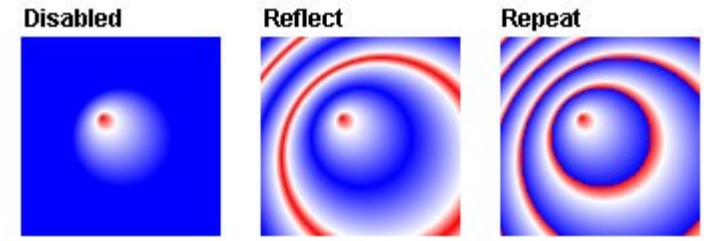
Focus X, Focus Y: 50.0

Цвета:

Относительное положение	Цвет
0.0	Красный
0.2	Белый
1.0	Синий



Другое изображение представляет каждый из трех цикличных методов, применимых к радиальному градиенту, с теми же параметрами, но с не центрированным фокусом (Focus X, Focus Y: 40.0):



Текстура

Текстура представляет собой способ заливки фигуры каким-либо изображением текстуры. Размер изображения принимает вид прямоугольника, размеры которого определяются параметрами Start X, Start Y, End X и End Y. Данный прямоугольник реплицируется во всех направлениях пространства фигуры.

Свойства заливки

Поле	Имя	Тип
Тип	type	целое
Цвет	color	цвет
Начало X	startx	плавающее
Начало Y	starty	плавающее
Конец X	endx	плавающее
Конец Y	endy	плавающее
Радиус	radius	плавающее
Фокус X	focusx	плавающее
Фокус Y	focusy	плавающее
Цвета	colors	таблица данных
Цикл	cycle	целое
Текстура	texture	блок данных

13.4.15.5 Штрих

Штрих определяет основной набор отображаемых атрибутов различных частей компонентов виджета. Атрибуты изображения, определяемые штрихом, описывают форму кисти, которой был очерчен контур какой-либо формы, оформлены её края и места соединений контура формы. Данные атрибуты включают:

Ширина линии

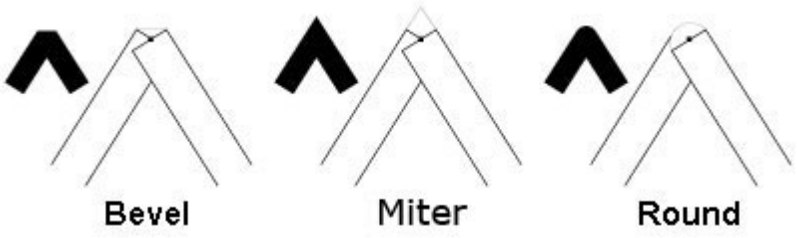
Ширина кисти, измеряемая перпендикулярно её траектории.

Форма окончания

Предмет оформления краев открытых внутренних контуров и сегментов штриха. Можно выделить три различных вида оформления: **Нет (None)**, **Круглый (Round)** и **Квадратный (Square)**.

Соединение штрихов

Оформление, применимое в точке сочленения двух отрезков. Выделяют три типа сочленений: **Срез (Bevel)**, **Угол (Miter)** и **Скругление (Round)**.



Предел угла

Коэффициент срезания линий с острым сочленением. Сочленение линий обрезается, когда пропорция длины острого сочленения относительно ширины штриха больше, чем значение коэффициент срезания. Длина острого сочленения является диагональной длиной сочленения, т.е. расстоянием между внутренней и внешней линиями. Чем меньше угол между линиями, тем больше длина сочленения и тем острее образующийся угол. Значение по умолчанию коэффициента срезания, равное 10.0, приведет к обрезке всех углов менее 11 градусов. Обрезка острого сочленения конвертирует оформление соединения линий в скошенное сочленение.

Массив штриха

Атрибуты **Массив штриха** и **Сдвиг штриха** определяют создание шаблона штриховых линий путем чередования непрозрачных и прозрачных сегментов.

Массив штриха представляет собой количество значений, передающих длину отрезков штриха. Другие записи в массиве представляют длину прозрачных и непрозрачных сегментов штриха пространства пользователя. По мере продвижения кисти во время обводки контура пространство пользователя, в котором осуществляется обводка, увеличивается. Значение расстояния используется для указания на массив штриха. Линия кисти становится непрозрачной, когда суммарное расстояние преобразуется в четный элемент массива штриха, в противном случае, она становится прозрачной.

Сдвиг штриха

Сдвиг штриха представляет собой расстояние промежутка между сегментами штриха. Иными словами, зазор определяет точку в шаблоне штриха, соответствующую началу обводки.

Изображение, приведенное далее, показывает вид обводки с различными массивами и зазором штриха:

Свойства штриха

Поле	Имя	Тип
Ширига линии	lineWidth	плавающее
Форма окончания	endCap	целое
Соединение штрихов	lineJoin	целое
Предел угла	miterLimit	плавающее
Массив штриха	dashArray	таблица данных
Сдвиг штриха	dashPhase	плавающее

13.4.15.6 Форма

Форма - это компонент простой геометрической формы, такой как прямоугольник или звезда.

Поддерживаются следующие типы форм:

- Прямоугольник
- Скругленный прямоугольник
- Эллипс/Круг
- Дуга
- Линия
- Многоугольник
- Звезда

Свойства форм

Поле	Имя	Тип	Описание
Тип	type	целое	Тип формы.
Координата X	x	плавающее	Крайняя левая точка **Прямоугольника, Скругленного прямоугольника, Эллипса/Круга, Дуги, Линии** или **Звезды**.
Координата Y	y	плавающее	Самая верхняя точка **Прямоугольника, Скругленного прямоугольника, Эллипса/Круга, Дуги, Линии** или **Звезды**.
Ширина	width	плавающее	Точка ширины **Прямоугольника, Скругленного прямоугольника, Эллипса/Круга, Дуги;** Для **Линии --** расстояние "End X - Start X".
Высота	height	плавающее	Точка высоты **Прямоугольника, Скругленного прямоугольника, Эллипса/Круга, Дуги;**

			Для **Линии --** расстояние "End Y - Start Y".
Ширина дуги	arcWidth	плавающее	Ширина дуги угла **Скругленного прямоугольника**.
Высота дуги	arcHeight	плавающее	Высота дуги угла **Скругленного прямоугольника**.
Начало	start	плавающее	Начало **Дуги** (в градусах).
Величина	extent	плавающее	Величина **Дуги** (в градусах).
Тип дуги	arcType	целое	Тип **Дуги**: • Открытая: дуга без линий, соединяющих начальную и конечную точки сегментов дуги. • Хорда: дуга, начальная и конечная точки которой соединены прямой линией. • Сектор: дуга, у которой прямая линия исходит из начальной точки сегмента к центру полного эллипса, а из него идет в конечную точку сегмента.
Точки	points	таблица данных	Точки **многоугольника,** т.е. ряд точек (X, Y). По умолчанию ряд точек многоугольника определяет форму "Крест".
Внутренний радиус	innerRadius	плавающее	Внутренний радиус **Звезды**.
Внешний радиус	outerRadius	плавающее	Внешний радиус **Звезды**.
Количество сторон	branchesCount	целое	Количество сторон **Звезды**.

13.4.15.7 Точки прикрепления

Точки прикрепления - точки, добавляемые для соединения компонентов при помощи связей[1273]. Корректная работа точек предусмотрена для абсолютной схемы компановки[954].

Каждая точка задается параметрами в таблице:

Свойства точек

Поле	Имя	Тип	Описание
Имя точки	pin name	Integer	Имя точки прикрепления. **Должно быть уникальным.**
Тип точки	pin type	Integer	1) none -1 2) input 0 3) input-output 1 4) output 2
X координат	pin X coordinate	Integer	X координата точки в процентах от размера компонента
У координат	pin У coordinate	Integer	У координата точки в процентах от размера компонента

13.4.16 События компонента

Компоненты виджета могут создавать события во время взаимодействия с пользователем. Например, событие **Действие** активируется, когда пользователь нажимает на него клавишей мыши.

Событие может определяться как активатор привязки 1296. "Активатор" приводит к тому, что посредством привязки записываются данные с виджета в контекст сервера, затем они считываются из контекста и используются в качестве содержания виджета, возможны также другие формы обработки данных.

Предположим, мы имеем текстовое поле **Имя**, которое привязано к полю какой-либо переменной контекста (т.е. имя пользователя). Его привязка также определяет событие **Активатор** -- событие **Действие** кнопки **Сохранить**, расположенной рядом. Когда пользователь нажимает на данную кнопку, возникает следующая последовательность действий:

1) Сканируются все привязки виджета в поисках событий определенного для них активатора.

2) Система видит, что привязка текстового поля **Имя** имеет событие **Действие**, заданное в качестве "активатора".

3) Далее происходит выполнение привязки совместно с другими привязками, для которых данное событие является активатором (например, **Фамилия**). Простыми словами, данные в текстовых полях теперь записаны в базу данных.

Общие события компонентов

Данный раздел описывает события, поддерживаемые большинством компонентов виджета. Отдельное описание каждого компонента включает список общих событий.

СКРЫТИЕ

Данное событие активируется, когда компонент убирается с экрана.

Имя события: **hidden**

Поля события:

Поле	Имя	Тип	Описание
идентификатор	id	целое	Идентификатор типа события.

ПОКАЗ

Данное событие активируется, когда компонент появляется на экране.

Имя события: **shown**

Поля события:

Поле	Имя	Тип	Описание
идентификатор	id	целое	Идентификатор типа события.

ПЕРЕМЕЩЕНИЕ

Данное событие активируется при перемещении компонента в контейнере-родителе.

Имя события: **moved**

Поля события:

Поле	Имя	Тип	Описание
идентификатор	id	целое	Идентификатор типа события.

ИЗМЕНЕНИЕ РАЗМЕРОВ

Данное событие активируется при изменении размера компонента.

Имя события: **resized**

Поля события:

Поле	Имя	Тип	Описание
идентификатор	id	целое	Идентификатор типа события.
Ширина	width	Integer	Новая ширина компонента.
Высота	height	Integer	Новая высота компонента.

КЛИК МЫШИ

Данное событие активируется при щелчке мыши по компоненту.

Событие не активируется, если между нажатием и отпусканием кнопки мыши было движение мыши.

Используйте это событие для реагирования на одиночные и тройные щелчки. Для двойных щелчков используйте событие Двойной клик мыши.

Имя события: **mouseClicked**

Параметры: см. событие мыши 1288.

ДВОЙНОЙ КЛИК МЫШИ

Данное событие активируется при двойном щелчке мыши по компоненту.

Имя события: **mouseDoubleClicked**

Параметры: см. событие мыши 1288.

НАЖАТИЕ КНОПКИ МЫШИ

Данное событие активируется при нажатии кнопки мыши по компоненту.

Имя события: **mousePressed**

Параметры: см. событие мыши 1288.

ОТПУСКАНИЕ КНОПКИ МЫШИ

Данное событие активируется при отпускании нажатой кнопки мыши.

Имя события: **mouseReleased**

Параметры: см. событие мыши 1288.

ВХОД МЫШИ

Данное событие активируется при появлении курсора мыши в области компонента.

Имя события: **mouseEntered**

Параметры: см. событие мыши 1288.

ВЫХОД МЫШИ

Данное событие активируется при выходе курсора мыши из области компонента.

Имя события: **mouseExited**

Параметры: см. событие мыши 1288.

ПЕРЕМЕЩЕНИЕ МЫШИ

Данное событие активируется при перемещении курсора мыши по области компонента.

Имя события: **mouseMoved**

Параметры: см. событие мыши 1288.

ВРАЩЕНИЕ КОЛЕСИКА МЫШИ

Данное событие активируется при прокрутке колесика мыши в компоненте.

Имя события: **mouseWheelMoved**

Поля события: Событие "Вращение колесика мыши" имеет все поля обычного события мыши 1288. К тому же оно определяет следующие поля:

Поле	Имя	Тип	Описание
Размер прокрутки	scrollAmount	целое	Количество элементов, которые нужно прокрутить за один клик вращения мыши. Действует только если **тип прокуртки -** это "прокуртка элементов".
Тип прокрутки	scrollType	целое	Тип прокрутки: • 0 - представляет собой прокрутку по "элементам" (как прокрутка клавишей-стрелкой) • 1 - Константа, представляющая прокрутку по "блокам" (как прокрутка при помощи клавиш "страница вверх", "страница вниз")
Вращение колесика	wheelRotation	целое	Количество "кликов", при которых вращается колесико. Отрицательные значения, если колесико мыши вращается вверх/в сторону от пользователя, и положительные значения, если колесико мыши вращается вниз/в сторону пользователя.

ПЕЧАТЬ КЛАВИШИ

Данное событие активируется при вводе символа кнопкой клавиатуры, когда компонент находится в фокусе.

Имя события: **keyTyped**

Параметры: см. событие клавиатуры 1289.

НАЖАТИЕ КЛАВИШИ

Данное событие активируется при нажатии кнопки клавиатуры, когда компонент находится в фокусе.

Имя события: **keyPressed**

Параметры: см. событие клавиатуры 1289.

ОТПУСКАНИЕ КЛАВИШИ

Данное событие активируется при отпускании кнопки, когда компонент находится в фокусе.

Имя события: **keyReleased**

Параметры: см. событие клавиатуры 1289.

ПОЛУЧЕНИЕ ФОКУСА

Данное событие активируется, когда компонент получает фокус ввода.

Имя события: **focusGained**

Поля события:

Поле	Имя	Тип	Описание
Идентификатор	id	целое	Идентификатор типа события.
Временный	temporary	логическое	Определяет фокус события как временный или постоянный.

ПОТЕРЯ ФОКУСА

Данное событие активируется, когда компонент теряет фокус ввода.

Имя события: **focusLost**

Поля событий:

Поле	Имя	Тип	Описание
Идентификатор	id	целое	Идентификатор типа события.

Временный	temporary	логическое	Определяет фокус события как временный или постоянный.

13.4.16.1 События мыши

Эта таблица описывает все поля событий компонента "мышь".

Поле	Имя	Тип	Описание
Идентификатор	id	целое	Идентификатор типа события.
Когда	when	дата	Временные метки события.
Модификаторы	modifiers	целое	Маска модификатора для данного события. Модификаторы представляют состояния для всех модальных клавиш, таких как ALT, CTRL, META, сразу после свершения события. Это значение поразрядное, ЛИБО состоит из следующих констант: • 1 - клавиша Shift • 2 - клавиша Control • 4 - клавиша Meta • 8 - клавиша Alt • 32 - клавиша Alt Graph
Нажатие Alt	altDown	логическое	True, если клавиша Alt была нажата, когда случилось событие.
Нажатие Alt Graph	altGraphDown	логическое	True, если клавиша Alt Graph была нажата, когда случилось событие.
Нажатие Control	controlDown	логическое	True, если клавиша Control была нажата, когда случилось событие.
Нажатие Shift	shiftDown	логическое	True, если клавиша Shift была нажата, когда случилось событие.
Нажатие Meta	metaDown	логическое	True, если клавиша Meta была нажата, когда случилось событие.
X	x	целое	Горизонтальная x позиция события, относящегося к компоненту-источнику.
Y	y	целое	Горизонтальная y позиция события, относящегося к компоненту-источнику.
X на Экране	xOnScreen	целое	Абсолютная горизонтальная x позиция события. В виртуальной многоэкранной среде устройства, где область экрана отображает множество экранов физических устройств, эта координата относится к виртуальной системе координат.
Y на Экране	yOnScreen	целое	Абсолютное горизонтальное положение y события. См. заметки сверху.
Кнопка	button	целое	Определяет, какая из кнопок мыши поменяла состояние: • 0 - ни одна из кнопок • 1 - кнопка #1 • 2 - кнопка #2 • 3 - кнопка #3
Количество кликов	clickCount	целое	Количество кликов мыши, ассоциируемых с этим событием.
Расширенные модификаторы	modifiersEx	целое	Расширенная маска модификатора для данного события. Расширенные модификаторы представляют состояние всех модальных клавиш, таких как ALT, CTRL, META, и кнопок мыши сразу после свершения события. Это значение поразрядное, ЛИБО состоит из следующих констант: • 64 - клавиша Shift

			• 128 - клавиша Ctrl • 256 - клавиша Meta • 512 - клавиша Alt • 1024 - кнопка мыши 1 • 2048 - кнопка мыши 2 • 4096 - кнопка мыши 3
Триггер всплывающего меню	popupTrigger	целое	Определяет, является ли данное событие событием триггера всплывающего меню для платформы.

13.4.16.2 События клавиатуры

Эта таблица описывает все поля событий компонента "клавиатура".

Поле	Имя	Тип	Описание
Идентификатор	id	целое	Идентификатор типа события.
Когда	when	дата	Временные метки события.
Модификаторы	modifiers	целое	Маска модификатора для данного события. Модификаторы представляют состояния всех модальных клавиш, таких как ALT, CTRL, META, сразу после совершения события. Это значение поразрядное, ЛИБО состоит из следующих констант: • 1 - клавиша Shift • 2 - клавиша Control • 4 - клавиша Meta • 8 - клавиша Alt • 32 - клавиша Alt Graph
Нажатие Alt	altDown	логическое	True, если клавиша Alt была нажата, когда случилось событие.
Нажатие Alt Graph	altGraphDown	логическое	True, если клавиша Alt Graph была нажата, когда случилось событие.
Нажатие Control	controlDown	логическое	True, если клавиша Control была нажата, когда случилось событие.
Нажатие Shift	shiftDown	логическое	True, если клавиша Shift была нажата, когда случилось событие.
Нажатие Meta	metaDown	логическое	True, если клавиша Meta была нажата, когда случилось событие.
Клавиша действия	actionKey	логическое	Определяет, является ли клавиша в этом событии клавишей "действия". Обычно клавиша действия не запускает символ Юникода и не является клавишей-модификатором.
Символ клавиши	keyChar	строка	Возвращает символ, ассоциируемый с клавишей в этом событии. Например, событие "Набор Клавиш" для Shift + "a" возвращает значение для "A". События "Клавиша Нажата" и "Клавиша Отпущена" не подразумевают отчета о вводе символа. Таким образом, значения, возвращаемые этим методом, гарантированно имеют значение только для событий "Набор Клавиш".
Код клавиши	keyCode	целое	Код клавиши в виде целого числа, ассоциируемый с клавишей в этом событии.
Положение клавиши	keyLocation	целое	Возвращает положение клавиши, создавшей это событие клавиши. Некоторые клавиши могут нажиматься больше одного раза, например, правая и левая клавиши Shift. Более того, некоторые клавиши

			нажимаются на цифровой клавиатуре. Это позволяет разграничивать подобные клавиши. Доступные положения: • 0 - неизвестное • 1 - стандартное • 2 - левое • 3 - правое • 4 - цифровая клавиатура

Коды клавиш

Клавиша	Код клавиши
VK_0	48
VK_1	49
VK_2	50
VK_3	51
VK_4	52
VK_5	53
VK_6	54
VK_7	55
VK_8	56
VK_9	57
VK_A	65
VK_ACCEPT	30
VK_ADD	107
VK_AGAIN	65481
VK_ALL_CANDIDATES	256
VK_ALPHANUMERIC	240
VK_ALT	18
VK_ALT_GRAPH	65406
VK_AMPERSAND	150
VK_ASTERISK	151
VK_AT	512
VK_B	66
VK_BACK_QUOTE	192
VK_BACK_SLASH	92
VK_BACK_SPACE	8
VK_BEGIN	65368
VK_BRACELEFT	161
VK_BRACERIGHT	162
VK_C	67
VK_CANCEL	3

VK_CAPS_LOCK	20
VK_CIRCUMFLEX	514
VK_CLEAR	12
VK_CLOSE_BRACKET	93
VK_CODE_INPUT	258
VK_COLON	513
VK_COMMA	44
VK_COMPOSE	65312
VK_CONTEXT_MENU	525
VK_CONTROL	17
VK_CONVERT	28
VK_COPY	65485
VK_CUT	65489
VK_D	68
VK_DEAD_ABOVEDOT	134
VK_DEAD_ABOVERING	136
VK_DEAD_ACUTE	129
VK_DEAD_BREVE	133
VK_DEAD_CARON	138
VK_DEAD_CEDILLA	139
VK_DEAD_CIRCUMFLEX	130
VK_DEAD_DIAERESIS	135
VK_DEAD_DOUBLEACUTE	137
VK_DEAD_GRAVE	128
VK_DEAD_IOTA	141
VK_DEAD_MACRON	132
VK_DEAD_OGONEK	140
VK_DEAD_SEMIVOICED_SOUND	143
VK_DEAD_TILDE	131
VK_DEAD_VOICED_SOUND	142
VK_DECIMAL	110
VK_DELETE	127
VK_DIVIDE	111
VK_DOLLAR	515
VK_DOWN	40
VK_E	69
VK_END	35
VK_ENTER	10
VK_EQUALS	61
VK_ESCAPE	27

VK_EURO_SIGN	516
VK_EXCLAMATION_MARK	517
VK_F	70
VK_F1	112
VK_F10	121
VK_F11	122
VK_F12	123
VK_F13	61440
VK_F14	61441
VK_F15	61442
VK_F16	61443
VK_F17	61444
VK_F18	61445
VK_F19	61446
VK_F2	113
VK_F20	61447
VK_F21	61448
VK_F22	61449
VK_F23	61450
VK_F24	61451
VK_F3	114
VK_F4	115
VK_F5	116
VK_F6	117
VK_F7	118
VK_F8	119
VK_F9	120
VK_FINAL	24
VK_FIND	65488
VK_FULL_WIDTH	243
VK_G	71
VK_GREATER	160
VK_H	72
VK_HALF_WIDTH	244
VK_HELP	156
VK_HIRAGANA	242
VK_HOME	36
VK_I	73
VK_INPUT_METHOD_ON_OFF	263
VK_INSERT	155

VK_INVERTED_EXCLAMATION_MARK	518
VK_J	74
VK_JAPANESE_HIRAGANA	260
VK_JAPANESE_KATAKANA	259
VK_JAPANESE_ROMAN	261
VK_K	75
VK_KANA	21
VK_KANA_LOCK	262
VK_KANJI	25
VK_KATAKANA	241
VK_KP_DOWN	225
VK_KP_LEFT	226
VK_KP_RIGHT	227
VK_KP_UP	224
VK_L	76
VK_LEFT	37
VK_LEFT_PARENTHESIS	519
VK_LESS	153
VK_M	77
VK_META	157
VK_MINUS	45
VK_MODECHANGE	31
VK_MULTIPLY	106
VK_N	78
VK_NONCONVERT	29
VK_NUM_LOCK	144
VK_NUMBER_SIGN	520
VK_NUMPAD0	96
VK_NUMPAD1	97
VK_NUMPAD2	98
VK_NUMPAD3	99
VK_NUMPAD4	100
VK_NUMPAD5	101
VK_NUMPAD6	102
VK_NUMPAD7	103
VK_NUMPAD8	104
VK_NUMPAD9	105
VK_O	79
VK_OPEN_BRACKET	91
VK_P	80

VK_PAGE_DOWN	34
VK_PAGE_UP	33
VK_PASTE	65487
VK_PAUSE	19
VK_PERIOD	46
VK_PLUS	521
VK_PREVIOUS_CANDIDATE	257
VK_PRINTSCREEN	154
VK_PROPS	65482
VK_Q	81
VK_QUOTE	222
VK_QUOTEDBL	152
VK_R	82
VK_RIGHT	39
VK_RIGHT_PARENTHESIS	522
VK_ROMAN_CHARACTERS	245
VK_S	83
VK_SCROLL_LOCK	145
VK_SEMICOLON	59
VK_SEPARATER	108
VK_SEPARATOR	108
VK_SHIFT	16
VK_SLASH	47
VK_SPACE	32
VK_STOP	65480
VK_SUBTRACT	109
VK_T	84
VK_TAB	9
VK_U	85
VK_UNDEFINED	0
VK_UNDERSCORE	523
VK_UNDO	65483
VK_UP	38
VK_V	86
VK_W	87
VK_WINDOWS	524
VK_X	88
VK_Y	89
VK_Z	90

13.4.17 Привязки

Если компоненты 955 виджета формируют "тело" виджета, то его "обработчик" состоит из **привязок**. Привязки определяют отношения между компонентами виджета и данными в контекстах сервера, таких как переменные 61 и функции 70. Каждая привязка обрабатывается во время выполнения виджета. Результаты обработки используются для изменения компонентов виджета или контекстных данных.

Привязки виджета расширяют привязки сервера 736. В то время как привязки сервера определяют отношения между объектами сервера, привязки виджета могут также определять отношения между объектами сервера и визуальными компонентами. При этом привязки виджета используют ту же самую синтаксическую структуру.

Привязки можно просматривать, создавать и редактировать:

- изменяя свойство "Привязки 1275" любого компонента виджета в AtomMind GUI Builder 423. Таблица привязок для определенного компонента включает только привязки, относящиеся к данному компоненту.
- изменяя свойство "Все привязки 1042" корневой панели (root panel) 1042. Таблица привязок включает все привязки, доступные для виджета.

AtomMind GUI Builder значительно упрощает создание привязок. Большинство из них создается при помощи простых операций перетаскивания. Более подробную информацию о данном процессе см. здесь 439.

13.4.17.1 Цель привязки

Цель привязки является объектом, на который воздействует привязка. Это может быть как компонент виджета, так и какие-либо данные в контексте AtomMind Server. Чтобы воздействовать на компонент, цель привязки указывает на одно из его свойств. Для изменения данных контекста цель привязки указывает на переменную контекста, поле переменной или поле возвращаемой функций контекста переменной.

Фактически цель привязки является ссылкой 117. Поддерживаются два типа целей: первый - без схемы 117 (следовательно, без префикса), указывающий на данные контекста, и второй, в котором используются схема `form` и префикс (`form/`), указывающий на свойства компонента.

Цели привязок виджета поддерживают все синтаксические варианты целей привязок сервера 737.

Поддерживаются следующие дополнительные синтаксические варианты:

1. Свойство компонента

`form/component:property`

Данная цель указывает на конкретное **свойство** компонента "виджет".

Цель привязки, которая ссылается на определенную ячейку таблицы, имеет следующий формат:

`form/component:property$field[row]`

Пример: `form/textField1`

Привязка с такой целью запишет результат в свойство по умолчанию `textField1`. Если `textField1` является текстовым полем 958, его свойство **text** будет изменено.

Пример: `form/image1:imageTable$imageData[0]`

Данная цель меняет первое изображение таблицы изображений 994 компонента "Изображение".

Пример: `form/gauge:datasets[1]`

Данная цель меняет значение второго массива данных (с индексом = 1) компонента Индикатор.

2. Скрипт виджета

`form/script()`

Данный тип цели привязки используется для запуска скрипта 1308 виджета с именем **script** во время обработки привязки, передавая результат выполнения выражения привязки в скрипт в качестве его параметра. Привязки с такими целями создаются автоматически при добавлении новых скриптов к шаблону виджета. Однако изначальные свойства 1295 такой привязки не определяют условия запуска виджета. Необходимо изменить данные свойства, чтобы виджет запускался при начале работы виджета, какого-либо события или просто периодически.

Пример: `form/refreshChart()`

Данная привязка запускает скрипт[1308] `refreshChart` виджета, который обновляет компонент График[1051], впоследствии отображающий самые последние данные.

13.4.17.2 Выражение привязки

Выражения привязки виджета очень похожи на выражения привязки сервера[738]. Однако выражения привязки в виджетах могут включать два типа ссылок:

- **Стандартные ссылки,** которые указывают на переменные[61] или функции[70] контекста[955], их поля или свойства, и
- **Ссылки на компоненты,** которые указывают на свойства компонентов[955] виджета.

Ссылки на компоненты

Ссылка на компонент указывает на свойство компонента виджета (например, текст метки[956]). У нее следующий формат:

`form/component:property`

`form` - это имя схемы[117], используемой для определения ссылки на компонент. Оно указывает процессору языка выражений использовать преобразователь[118], который распознает ссылки на компонент и знает, что с ними делать. Ссылки на компонент всегда нужно начинать с `form/`.

`component` - это имя компонента[955] виджета, с которым вы хотите работать. Имя компонента показано в Окне Ресурса[428] и в названии Окна Свойств Компонента[431], когда компонент выбирается в AtomMind GUI Builder.

`property` - это имя конкретного свойства компонента. Имена свойств можно найти в описании свойств каждого компонента виджета. Часть "свойство" ссылки на компонент опционально. Если она опускается, ссылка указывает на свойство по умолчанию[1274] компонента.

Тип свойства[1274], например, тип значения, возвращенного ссылкой на свойство компонента, можно найти в описании этого свойства в ссылке на компоненты[955].

ПРИМЕРЫ ССЫЛКИ НА СВОЙСТВО КОМПОНЕНТА

```
{form/userNameField:}
{form/userNameField:text}
```

Обе этих ссылки разрешаются в текст, содержащийся в данный момент в `userNameField` (при условии, что это поле текста[958]). Первый вариант указывает на свойство по умолчанию[1274] компонента, которое называется `text`, а второй явно называет его (`:text`).

Пример выражения привязки виджета

```
{form/slider1:value} * 100
```

Это выражение разрешается в число, равное свойству **Значение** (которое является свойством по умолчанию и, таким образом, не выражено в явной форме) компонента Регулятор[987] под названием **slider1**, умноженного на 100.

13.4.17.3 Активатор привязки

Активатор привязки является ссылкой[117], указывающей на:

- свойство[1274] или событие[1285] компонента виджета
- свойство (переменную) или событие контекста сервера (см. активатор привязки сервера[739])
- элемент контекстного меню компонента виджета

Изменение переменной виджета/сервера, экземпляр события виджета/сервера или выбор элемента контекстного меню вызывает обработку этой привязки.

Активаторы привязок виджета поддерживают все варианты синтаксиса активаторов привязки сервера[739].

Далее приведен список дополнительно поддерживаемых вариантов синтаксиса:

1. Свойство компонента виджета

`form/component:property`

Если `property` `component` изменяется, запускается активатор.

Пример: `form/setpoint:text`

Этот активатор запускается, когда редактируется `text` в тестовом поле 958, названный `setpoint`.

2. Событие компонента виджета

`form/component:event@`

Если `component` имеет событие с именом `event`, запуск активатора происходит при возникновении события.

Пример: `form/button1:click@`

Запуск данного активатора произойдет при нажатии кнопки 968 `button1`.

3. Элемент всплывающего меню

`menu/component:item`

Данный активатор запустит привязку при выборе элемента меню `item` из контекстного меню 1276 `component`.

13.4.17.4 Условие привязки

Условия привязки виджета очень похожи на условия привязки сервера 739. Однако условия привязки в виджетах могут включать два типа ссылок:

- **Стандартные ссылки,** которые указывают на переменные 61 или функции 70 контекста 955, их поля или свойства, и
- **Ссылки на компоненты,** которые указывают на свойства компонентов 955 виджета.

Смотрите выражения привязки виджета 1296 для получения более подробной информации о ссылках на компоненты виджета.

Пример условия привязки виджета

`{value/altDown} && ({value/keyCode} == 'A')` - это выражение вызывает привязку активатора при событии набора клавиши 1285, если была нажата комбинация клавиш Alt-A.

13.4.17.5 Производительность привязок

Привязки обычно обрабатываются одна за другой. Например, если у вас есть виджет, отображающий 20 разных метрик устройства, и чтение каждой метрики из удаленного сервера занимает 1 секунду, загрузка виджета занимает 20 секунд.

Чтобы увеличить скорость загрузки виджета, включите флажок **Одновременное выполнение начальных привязок** корневой панели 1042. В этом случае каждая привязка обрабатывается в отдельном потоке. В приведенном выше примере это заставляет все запросы метрик выполняться одновременно, поэтому виджет загружается за 1 секунду (в 20 раз быстрее).

Для более подробной информации см. основной раздел Производительность привязок 742.

13.4.17.6 Примеры привязок

Этот раздел описывает реальные примеры привязок виджета.

Пример 1. Открытие нового виджета нажатием кнопки

Этот пример показывает, как выполнить действие 87 сервера нажатием кнопки. Наиболее типичным при использовании привязки этого типа является открытие нового виджета, когда кнопка нажимается на любом другом виджете.

Во-первых, поместите компонент Кнопка 968 на вашем виджете. Во-вторых, добавьте привязку 1295 к кнопке и настройте эту привязку следующим образом:

- Настройте событие **Нажатие кнопки мышки** как Активатор 1296 привязки, то есть настройте Активатор на `form/button1:mousePressed@`
- Настройте его действие Запуск Виджета 1627 как цель 1295 привязки, то есть настройте Цель на `users.admin.widgets.bindingsDemo:launch!`, где `users.admin.widgets.bindingsDemo` - это контекстный путь запуска виджета.
- Оставьте Выражение привязки пустым.

Настройка привязки:

Цель	`users.admin.widgets.bindingsDemo:launch!`
Выражение	
Активатор	`form/button1:mousePressed@`
Условие	
Опции	`При событии`

Возможно также закрыть текущий виджет (тот, что с кнопкой) до выполнения действия. Для настройки этого:

- Откройте свойство **Операции** кнопки
- Добавьте запись с операцией **Закрыть**

Пример 2: Цвет панели привязки по статусу контроллера

Эта привязка окрашивает панель 1045 статуса в красный цвет, если от контроллера поступает тревога о перенапряжении. В других случаях панель останется зеленой.

Имейте в виду, что ссылки на выражение и цель используют пути относительного контекста ("."), чтобы виджет соотносился с множеством подобных контроллеров. Виджет должен быть относительным 946 и действительным для всех устройств типа "контроллер напряжения".

Цель	`form/voltageAlertPanel:background`
Выражение	`{.:voltageAlert$voltageAlert} ? color(255, 0, 0) : color(0, 255, 0)`
Активатор	
Условие	
Опции	`При запуске, при событии`

Возможно изменить вышеобозначенную привязку, чтобы окрашивать панель в красный цвет, если температура превышает 50 градусов:

```
({.:voltage$voltage} > 50) ? color(255, 0, 0) : color(0, 255, 0)
```

Пример 3: Компонент регулятора привязки для установки заданного значения контроллера

Эта пара привязок меняет заданное значение контроллера (например, контроллера температуры) как только смещается регулятор 987.

Первая привязка используется для чтения текущего заданного значения из контроллера и инициализации регулятора:

Цель	`form/slider1:value`
Выражение	`{.:currentTemperature$temperatureCelsius}`

Активатор

Условие

Опции При запуске

Вторая привязка записывает новые заданные значения в контроллер, как только смещается регулятор:

Цель .:currentTemperature$temperatureCelsius

Выражение {form/slider1:value}

Активатор

Условие

Опции При событии

Пример 4: Редактор таблицы привязок для результатов запроса

Эта привязка выполняет запрос 829 сервера, чей текст содержится в текстовом поле queryText и помещает результаты запроса в компонент Редактор таблицы данных 979 с названием results.

Запрос выполняется вызовом executeQuery корневого контекста сервера 1559. Операция выполняется при помощи клика кнопки 968 execute .

Обратите внимание, что первый параметр функции executeQuery помещается в одинарные кавычки, чтобы система обрабатывала его как выражение.

Цель form/result:dataTable

Выражение {:executeQuery('{form/queryText:text}')}

Активатор form/execute:click@

Условие

Опции При событии

Возможно использовать альтернативный синтаксис для вышеобозначенных выражений привязки:

```
callFunction("", "executeQuery", {form/query:text})
```

Это выражение функционирует так же, как вышеобозначенное. Однако оно задействует функцию языка выражений callFunction(), чтобы вызвать функцию контекста сервера, ясно определяя путь контекста, имя функции и ее параметры.

Пример 5: Вызов функции устройства

Этот пример объясняет, как произвести вызов функции 70 устройства или сервера из виджета при условии, что у нас есть устройство, поддерживающее операцию multiply с двумя числовыми аргументами. Можно вызвать эту операцию при помощи нажатия кнопки calculate, вводя числа в два текстовых поля (argument1 и argument2).

ПЕРВЫЙ СПОСОБ: ВЫЗОВ ФУНКЦИИ ИЗ ЦЕЛИ ПРИВЯЗКИ ВИДЖЕТА

В этом случае наша цель привязки указывает на функцию, а ее выражение возвращает предварительно подготовленную таблицу параметров:

Цель .:multiply()

Выражение table(functionInputFormat(), {form/argument1:text}, {form/argument1:text})

Активатор form/calculate:click@

Условие	
Опции	При событии

Имейте в виду, что мы используем путь относительного контекста (".") в цели, чтобы сделать расчет виджета совместимым со многими агентами вычисления. Чтобы использовать его всего лишь с одним агентом, определите абсолютный путь, например `users.my_user.devices.my_agent.`

В этом примере результат вычисления, возвращенный множеством операций, исключается. Но что если мы захотим поместить его в другой компонент виджета (такой как метка 956)? Вот решение:

ВТОРОЙ СПОСОБ: ВЫЗОВ ФУНКЦИИ ИЗ ВЫРАЖЕНИЯ ПРИВЯЗКИ ВИДЖЕТА

В этом случае наше выражение привязки вызывает функцию и возвращает ее выход (Таблицу данных 49). Действительный результат вычисления (числовое поле `result`) извлекается из таблицы при помощи функции `cell()`. Этот результат записывается в метку `result`:

Цель	`form/result:text`
Выражение	`cell(callFunction(dc(), "multiply", {form/argument1:text}, {form/argument1:text}), "result")`
Активатор	`form/calculate:click@`
Условие	
Опции	При событии

Имейте в виду, что используется относительный путь контекста устройства (возвращаемый функцией `dc()`). Чтобы использовать абсолютный путь контекста, поменяйте `dc()` на полный путь устройства, например `users.my_user.devices.my_agent`.

Существует еще один (более компактный) стиль записи вышеобозначенного выражения привязки для получения точно такого же результата.

Этот стиль использует ссылку функции 120 вместо функции языка выражений `callFunction()`:

```
{.:multiply('{form/argument1:text}', '{form/argument2:text}')$result}
```

13.4.18 Визуализация топологии

Компоненты Граф 1025 и Карта 1010 позволяют визуализировать топологию устройства, например, связи между устройствами.

Визуализация топологии настраивается определенными свойствами:

ПОСТАВЩИК

Выбирает тип поставщика топологических данных. Список доступных типов может разниться в различных инсталляциях AtomMind. Например, доступные типы включают:

- **Пользовательский** (визуализирует любые связи между устройствами, которые определены нестандартным образом)
- **Сеть** (топология сети всех типов, например, Уровень 2 и Уровень 3)
- **Уровень 2** (OSI модель, слой топологии 2)
- **Уровень 3** (OSI модель, слой топологии 3)

Для компонента Карта 1010 Поставщик может быть выставлен в пустую строку, чтобы отключить визуализацию топологии.

Имя свойства: **provider**

Тип свойства: **String**

ВХОДНОЙ И ВЫХОДНОЙ УЗЛЫ ЯВЛЯЮТСЯ КОНТЕКСТАМИ

Определяет, является ли контекстом 41 входной и выходной режимы. Если этот флажок активен, различные выражения, которые оценивают свойства узла, могут посылать этот контекст как контекст по умолчанию 119. Если данный флажок неактивен, **Входное выражение** и **Выходное выражение** возвращают имена узлов вместо их путей контекста.

Имя свойства: **contextMode**

Тип свойства: **Boolean**

ПОКАЗЫВАТЬ НЕСВЯЗАННЫЕ УЗЛЫ

Определяет показанные узлы, которые не имеют ссылок.

Имя свойства: **displayUnlinkedNodes**

Тип свойства: **Boolean**

ВЫРАЖЕНИЕ ТОПОЛОГИИ

Используется только **Пользовательским** поставщиком топологии. Это выражение должно иметь результатом таблицу данных 49. Эта таблица данных 49 становится таблицей данных по умолчанию для **Выражения узлов** и **Выражения связей**.

Среда вычисления 114 выражения топологии:	
Контекст по умолчанию 119	Отсутствует.
Таблица данных по умолчанию 120	Отсутствует.
Строка по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

Имя свойства: **topologyExpression**

Тип свойства: **String**

ВЫРАЖЕНИЕ СВЯЗЕЙ

Используется только **Пользовательским** поставщиком топологии. Это выражение должно иметь результатом таблицу данных 49. Каждая запись в этой таблице будет определять одну связь топологии, то есть она должна каким-либо образом соотносить источник связи с целью, которая будет получена через **Выражение идентификатора связей**, **Выражение источника** и **Выражение назначения**. Запись может также определять и другие свойства связи, например, цвет, ширину, описание и пр. Эти свойства будут получены с помощью **Выражения цвета**, **Выражения ширины** и пр.

Среда вычисления 114 выражения топологии:	
Контекст по умолчанию 119	Отсутствует.
Таблица данных по умолчанию 120	Результат **Выражения топологии**.
Строка по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

Имя свойства: **linkExpression**

Тип свойства: **String**

ВЫРАЖЕНИЕ ИДЕНТИФИКАТОРОВ СВЯЗЕЙ

Используется только **Пользовательским** поставщиком топологии. Это выражение вычисляется для каждой записи таблицы, возвращенной **Выражением связей**. Результатом выражения должна быть строка, интерпретируемая как идентификатор связи.

Среда вычисления 114 выражения топологии:	
Контекст по умолчанию 119	Отсутствует.
Таблица данных по умолчанию 120	Результат **Выражения связей.**
Строка по умолчанию 119	Обрабатываемая в данный момент строка таблицы по умолчанию.
Переменные среды 123	Только стандартные 123 переменные.

Имя свойства: **linkIdExpression**

Тип свойства: **String**

ВЫРАЖЕНИЕ ИСТОЧНИКА

Используется только **Пользовательским** поставщиком топологии. Это выражение оценивается для каждой записи таблицы, возвращенной **Выражением топологии**. Выражение должно иметь результатом Строку, интерпретируемую как путь контекста 41 источника топологической связи (например, контекст устройства 1494).

Среда вычисления 114 выражения источника:	
Контекст по умолчанию 119	Отсутствует.
Таблица данных по умолчанию 120	Таблица данных, возвращенная **Выражением Топологии**.
Строка по умолчанию 119	Обрабатываемая в данный момент строка таблицы по умолчанию.
Переменные среды 123	Только стандартные 123 переменные.

Имя свойства: **sourceExpression**

Тип свойства: **String**

ВЫРАЖЕНИЕ ЦЕЛИ

Используется только **Пользовательским** поставщиком топологии. Это выражение оценивается для каждой записи таблицы, возвращенной **Выражением топологии**. Выражение должно иметь результатом Строку, интерпретируемую как путь контекста 41 источника топологической связи (например, контекст устройства 1494).

Среда вычисления 114 выражения цели:	
Контекст по умолчанию 119	Отсутствует.
Таблица данных по умолчанию 120	Таблица данных, возвращенная **Выражением Топологии**.

Строка по умолчанию 119	Обрабатываемая в данный момент строка таблицы по умолчанию.
Переменные среды 123	Только стандартные 123 переменные.

Имя свойства: **targetExpression**

Тип свойства: **String**

ВЫРАЖЕНИЕ УЗЛОВ

Используется только **Пользовательским** поставщиком топологии. Результат выражения должен быть Таблицей данных 49. Каждая запись этой таблицы будет опрелеять один узел топологии, поэтому она должна содержать информацию для **Выражения идентификаторов связей**. Запись может также определять другие свойства узла, например, цвет, ширину, описание и пр.

Среда вычисления 114 выражения топологии	
Контекст по умолчанию 119	Отсутствует.
Таблица данных по умолчанию 120	Результат **Выражения топологии**.
Строка по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

Имя свойства: **nodeExpression**

Тип свойства: **String**

ВЫРАЖЕНИЕ ИДЕНТИФИКАТОРОВ УЗЛОВ

Используется только **Пользовательским** поставщиком топологии. Это выражение вычисляется для каждой записи таблицы, возвращенной **Выражением узлов**. Выражение должно иметь результатом Строку, интерпретируемую как идентификатор узла.

Среда вычисления 114 выражения топологии	
Контекст по умолчанию 119	Отсутствует.
Таблица данных по умолчанию 120	Результат **Выражения узлов**.
Строка по умолчанию 119	Обрабатываемая в данный момент строка таблицы по умолчанию.
Переменные среды 123	Только стандартные 123 переменные.

Property name: **nodeIdExpression**

Property type: **String**

ВЫРАЖЕНИЕ ОПИСАНИЯ УЗЛА

Это опциональное выражение оценивается для каждого Контекста, являющегося источником или целью топологической связи. Оно должно иметь результатом Строку, которая определяет пользовательское описание топологического узла.

Среда вычисления 114 выражения описания узла:	
Контекст по умолчанию 119	Контекст объекта (обычно устройство), соответствующий источнику или цели топологической связи.
Таблица данных по умолчанию 120	Таблица данных, представляющая текущую топологическую связь. Строковое поле `source` содержит путь контекста источника связи. Строковое поле `target` содержит путь контекста цели связи.
Строка по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

Имя свойства: **nodeDescriptionExpression**

Тип свойства: **String**

ВЫРАЖЕНИЕ ЦВЕТА

Это опциональное выражение оценивается для каждого Контекста, являющегося источником или целью топологической связи. Оно должно иметь результатом Цвет, используемый для отрисовки топологического узла.

Среда вычисления 114 выражения цвета:	
Контекст по умолчанию 119	Контекст объекта (обычно устройство), соответствующий источнику или цели топологической связи.
Таблица данных по умолчанию 120	Таблица данных, представляющая текущую топологическую связь. Строковое поле `source` содержит путь контекста источника связи. Строковое поле `target` содержит путь контекста цели связи.
Строка по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

Имя свойства: **colorExpression**

Тип свойства: **String**

ВЫРАЖЕНИЕ ТИПА

Это выражение оценивается для каждого Контекста, являющегося источником или целью топологической связи. Оно должно иметь результатом Строку, которая будет приравнена к имени файла пиктограммы топологического узла.

Пиктограммы топологического узла находятся в подпапке `/images/shapes` установочной папки AtomMind. Имя файла пиктограммы должно соответствовать Строке, возвращенной **Выражением типа**. Расширение файла пиктограммы должно быть `.svg`, т.е. все пиктограммы - это векторные SVG изображения.

Каждое SVG изображение, используемое как пиктограмма топологического узла, должно включать единственный элемент (тег) внутри его тегов <svg></svg>. Этот тег может быть, например, <path>, <circle> или любым другим разрешенным SVG элементом. Вложенные элементы не разрешаются.

Среда вычисления 114 выражения типа:	
Контекст по умолчанию 119	Контекст объекта (обычно устройство), соответствующий источнику или цели топологической связи.
Таблица данных по	Таблица данных, представляющая текущую топологическую связь. Строковое поле `source` содержит путь контекста источника связи. Строковое поле `target` содержит путь контекста

умолчанию 120	цели связи.
Строка по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

Имя свойства: **typeExpression**

Тип свойства: **String**

ВЫРАЖЕНИЕ ИНТЕРФЕЙСА

Это опциональное выражение оценивается для каждого Контекста, являющегося источником или целью топологической связи. Оно должно иметь результатом целое число, которое определяет интерфейс узла источника/цели, к которому принадлежит текущая связь.

Среда вычисления 114 выражения интерфейса::	
Контекст по умолчанию 119	Контекст объекта (обычно устройство), соответствующий источнику или цели топологической связи.
Таблица данных по умолчанию 120	Таблица данных, представляющая текущую топологическую связь. Строковое поле `source` содержит путь контекста источника связи. Строквое поле `target` содержит путь контекста цели связи.
Строка по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

Имя свойства: **interfaceExpression**

Тип свойства: **String**

ВЫРАЖЕНИЕ НАПРАВЛЕННОСТИ

Это опциональное выражение оценивается для каждой топологической связи. Оно должно иметь результатом логическое значение, определяющее, направлена или не направлена связь (например, направляется из своего Источника к Цели). По умолчанию все связи не направлены.

Это свойство недоступно для компоновки типа Дерево, Радиальная и Пузырьковая. Эти компоновки не рекомендуются для визуализации топологий, т.к. изначальная информация направления связи перекрывается алгоритмом компоновки.

Среда вычисления 114 выражения направленности:	
Контекст по умолчанию 119	Контекст цели топологической связи.
Таблица данных по умолчанию 120	Таблица данных, представляющая текущую топологическую связь. Строковое поле `source` содержит путь контекста источника связи. Строковое поле `target` содержит путь контекста цели связи.
Строка по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

Имя свойства: **directedExpression**

Тип свойства: **String**

ВЫРАЖЕНИЕ ШИРИНЫ

Это опциональное выражение оценивается для каждой топологической связи. Оно должно иметь результатом число с плавающей точкой, определяющее ширину топологической связи. Ширина связи по умолчанию - 1 пиксель.

Среда вычисления 114 выражения ширины:	
Контекст по умолчанию 119	Контекст цели топологической связи.
Таблица данных по умолчанию 120	Таблица данных, представляющая текущую топологическую связь. Строковое поле `source` содержит путь контекста источника связи. Строковое поле `target` содержит путь контекста цели связи.
Строка по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

Имя свойства: **widthExpression**

Тип свойства: **String**

ВЫРАЖЕНИЕ ОПИСАНИЯ СВЯЗИ

Это опциональное выражение оценивается для каждой топологической связи. Оно должно иметь результатом строку, определяющую пользовательское описание топологической связи.

Среда вычисления 114 выражения описания связи:	
Контекст по умолчанию 119	Контекст цели топологической связи.
Таблица данных по умолчанию 120	Таблица данных, представляющая текущую топологическую связь. Строковое поле `source` содержит путь контекста источника связи. Строковое поле `target` содержит путь контекста цели связи.
Строка по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

Имя свойства: **linkDescriptionExpression**

Тип свойства: **String**

ПОКАЗЫВАТЬ ОПИСАНИЯ СВЯЗИ

Если этот флажок активирован, будет отображаться описание связи.

Имя свойства: **linkDescriptionShowMode**

Тип свойства: **Boolean**

ВЫРАЖЕНИЕ ЦВЕТА СВЯЗИ

Это опциональное выражение оценивается для каждой топологической связи. Оно должно иметь результатом Цвет, используемый для отрисовки топологической связи.

Среда вычисления 114 выражения цвета связи:	
Контекст по умолчанию 119	Контекст цели топологической связи.

Таблица данных по умолчанию 120	Таблица данных, представляющая текущую топологическую связь. Строковое поле `source` содержит путь контекста источника связи. Строковое поле `target` содержит путь контекста цели связи.
Строка по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

Имя свойства: **linkColorExpression**

Тип свойства: **String**

ВЫРАЖЕНИЕ ИЗОБРАЖЕНИЯ

Выражение, которое вернет изображения объектов слоя.

Контекст по умолчанию 119	Контекст слоя (для типа Геозона), контекст устройства (для типа Контекст), отсутствует (для типа Выражение таблицы узлов).
Таблица данных по умолчанию 120	Итоговая таблица выражения таблицы узлов (для типа Выражение таблицы узлов), отсутствует для остальных.
Строка по умолчанию 119	Указывает на обрабатываемую в данный момент ряд таблицы данных по умолчанию, отсутствует (для типа Контекст).
Переменные среды 123	Только стандартные 123 переменные.

Имя свойства: **imageExpression**

Тип свойства: **String**

Выражение по умолчанию `(select({users.admin.models.deviceImages:deviceImages}, 'image', 'name', {.:genericProperties$type}))` берет изображения из **Изображения устройства** 1022

ТОЛЬКО ЧТЕНИЕ

Отключает возможность удаления или сохранения расположений.

МЕТКИ

Конфигурация динамических меток, обозначающих текущее состояние устройства.

Свойства метки:

Поле	Тип	Описание
Выражение	Строка	Выражение 112 текста метки. Это выражение будет рассчитываться каждый **Период**. Его результат будет преобразован в строку и отображен на метке.

Контекст по умолчанию 119 выражения метки:

Контекст по умолчанию 119	Контекст, определенный свойством **Источник** данного компонента устройства. Если **Источник** не определен, Контекст по умолчанию 119 работающего в данный момент виджета.
Таблица данных по умолчанию 120	Таблица параметров виджета 949.
Строка по	0

		умолчанию 119
		Переменные среды 123 — Только стандартные 123 переменные.
Вертикальное выравнивание	Целое	Вертикальное выравнивание метки.
Горизонтальное выравнивание	Целое	Горизонтальное выравнивание метки.
Шрифт	Таблица данных	Шрифт 1278 метки.
Передний план	Цвет	Цвет метки
Период	Длинный	Период обновлений текста метки, т.е. период переоценки **Выражения**.

Имя свойства: **labels**

Тип свойства: **Data Table**

Визуализация отношения "устройство-родитель дочернему устройству" на карте визуализации

Этот пример объясняет, как использовать поставщика топологии `Custom` для визуализации отношения "устройство-родитель дочернему устройству", определенное моделью 810 **Подключаемость устройств**, включенной в продукт AtomMind Network Manager.

Модель **Подключаемость устройств** прикрепляет табличное свойство **Родители** к каждому устройству Хост сети 593 (можно легко прикрепить ко всем устройствам). Таблица **Родители** имеет поле **Путь,** представляющее путь контекста устройства-родителя. Таблица содержит список устройств, влияющих на подключаемость к сети текущего устройства.

Чтобы визуализировать все связи "дочернее устройство устройству-родителю" на топологическом графике или карте, необходима следующая настройка **Пользовательского** поставщика топологии:

- Выражение топологии: `callFunction("", "executeQuery", "SELECT parents.parents$path as parent, children.info$remotePath as child FROM users.*.devices.*:parents as parents LEFT OUTER JOIN users.*.devices.*:info as children ON children$CONTEXT_ID = parents$CONTEXT_ID")`. Это выражение выполняет запрос 829, возвращающий список всех связей "устройство-родителю", определяемых в AtomMind Server и доступных текущему пользователю. Таблица результатов запроса имеет поля `parent` и `child`, содержащие пути контекстов конечной точки связи.
- Выражение источника: `{parent}`. Это выражение извлекает путь контекста источника связи из каждой строки вышеобозначенных результатов запроса.
- Выражение цели: `{child}`. Это выражение извлекает путь контекста цели связи из каждой строки вышеобозначенных результатов запроса.
- Все другие выражения могут сохранять настройки по умолчанию, поскольку извлекают данные из источника связи и контекстов цели.

13.4.19 Скрипты виджета

Скрипты виджета позволяют выполнять операции пользователя с компонентами виджета и данными сервера. Скрипты написаны на языке Java. Каждый скрипт выполняется в Java Virtual Machine (JVM), в которой выполняется виджет (это может быть JVM, запускающая AtomMind Client или AtomMind Server JVM, если виджет запускается в веб-интерфейсе). Таким образом, скрипт имеет доступ ко всем объектам внутренней памяти и структурам виджета. Скрипты дают вам возможность полностью контролировать виджет.

Права доступа скрипта никак не ограничены. Одна случайная ошибка в скрипте или враждебный программный код могут привести к неправильному функционированию %ag%> сервера или клиента, к их зависанию, 100% загрузке процессора, повреждению данных или даже повреждению данных устройства, которое запустило скрипт!

Запуск выполнения скрипта

Скрипт виджета выполняется в двух случаях:

- При обработке привязки, чья цель указывает на этот скрипт;
- Когда выражение привязки вызывает функцию `script()`.

Скрипты создаются в потоке обработки привязки, поэтому их выполнение может занимать много времени с учетом обработки других привязок виджета. Рекомендуется создавать новые потоки для выполнения задач скриптов виджета, требующих большого количества времени.

Управление скриптами

Скрипты создаются и управляются редактированием свойств Скриптов корневой панели виджета.

Интерфейс скрипта

Каждый скрипт является классом Java, который должен выполнять интерфейс `WidgetScript`:

```
public interface WidgetScript
{
  public void execute(WidgetScriptExecutionEnvironment environment, Object parameter);
}
```

Данный интерфейс определяет метод `execute()`, который вызывается при выполнении скрипта.

Результат выполнения выражения привязки передается скрипту виджета в качестве объекта `parameter.`

Среда выполнения скрипта

Каждый скрипт имеет доступ к объекту, выполняющему интерфейс `WidgetScriptExecutionEnvironment`, который передается как аргумент для метода **execute()**. `WidgetScriptExecutionEnvironment` выглядит следующим образом:

```
public interface WidgetScript
{
  public void execute(WidgetScriptExecutionEnvironment environment, Object parameter);
}
```

Экземпляр `WidgetScriptExecutionEnvironment` обеспечивает доступ к объекту, выполняющему интерфейс `GUIEngine` (получить интерфейс можно путем вызова метода `getEngine()`). `GUIEngine` обеспечивает доступ к объектам, отвечающим за выполнение виджета.

Метод `getCause()` возвращает ссылку (объект типа "ссылка"), которая инициирует обработку привязки, вызвавшую выполнение скрипта. Данная ссылка может указывать на данные контекста или свойство компонента виджета.

Шаблон скрипта

При создании нового скрипта его текст непустой. Он содержит автоматически генерируемый класс с пустым методом `execute()`. Далее приведен текст скрипта по умолчанию:

```
import com.tibbo.platform.guibuilder.*;
public class users_admin_widgets_scripts_refresh implements WidgetScript
{
  public void execute(WidgetScriptExecutionEnvironment environment, Object parameter)
  {
  }
}
```

Разработка скриптов

См. раздел Общие указания по программированию для получения информации о разработке скриптов виджетов AtomMind.

НАПИСАНИЕ СКРИПТОВ

По сути скрипты должны выполнять следующее:

- Читать/записывать свойства компонентов виджета
- Генерировать события компонентов виджета
- Читать/записывать переменные контекста сервера (т.е. настройки аппартаного устройства и свойства ресурсов сервера)
- Вызывать операции сервера и устройства (функции)

В большинстве случаев все операции должны производиться через интерфейс `Context`.

Для получения `Context`, соответствующего любому объекту сервера, используйте следующий код:

```
WidgetEngine engine = environment.getEngine();
ContextManager contextManager = engine.getServerContextManager();
Context serverContext = contextManager.get("server.context.path");
```

Для получения `Context`, соответствующего определенному компоненту виджета, используйте следующий код:

```
Context componentContext = environment.getComponentContext("widget_component_name");
```

Когда извлекается экземпляр `Context`, вы можете читать/записывать переменные, используя методы `getVariable()` и `setVariable()`. Для контекстов компонента виджета большинство переменных напрямую соответствуют свойствам компонента (т.е. Шрифт, Ширина, Цвет и т.д.)

Пример 1: Обработка свойства компонента

Например, для применения пользовательской обработки к Таблице Данных, хранящейся в компоненте Редактор Таблицы Данных 979, используйте следующий код:

```
Context dataTableEditorContext = environment.getComponentContext("dataTableEditor1");
DataTable dataTable = dataTableEditorContext.getVariable("dataTable");
// Process the data here
```

Пример 2: Закрытие другого виджета

Этот пример иллюстрирует, как один виджет закрывает другой виджет при помощи щелчка мыши.

Скрипт вызывает статический метод `ClientUtils.removeFrame()` и передает ключу фрейма действие к закрытию. Ключ фрейма виджета создается методом `ClientUtils.createWidgetFrameKey()`, который принимает путь контекста виджета и путь его контекста по умолчанию 946 (или корневой путь контекста, т.е. пустую строку в случае абсолютного 946 виджета).

```
import com.tibbo.aggregate.common.script.*;
import com.tibbo.aggregate.common.widget.*;
import com.tibbo.aggregate.client.util.*;

public class %ScriptClassNamePattern% implements WidgetScript
{
  public Object execute(WidgetScriptExecutionEnvironment environment, Object... parameters)
  {
    ClientUtils.removeFrame(ClientUtils.createWidgetFrameKey("users.admin.widgets.test", ""));
    return null;
  }
}
```

Пример 3: Открытие URL в браузере

Этот пример иллюстрирует, как открыть определенный веб-сайт в браузере. Нижеприведенный скрипт можно запустить при помощи щелчка мыши. Он принимает URL как единственный параметр.

```
import java.awt.*;
import java.net.*;

import com.tibbo.aggregate.common.widget.*;

public class %ScriptClassNamePattern% implements WidgetScript
```

```
{
  public Object execute(WidgetScriptExecutionEnvironment environment, Object... parameters)
  {
    try
    {
      Desktop.getDesktop().browse(new URI(parameters[0].toString()));
    }
    catch (Exception ex)
    {
      throw new IllegalStateException("Error opening url '" + parameters[0].toString() + "'", ex);
    }

    return null;
  }
}
```

13.4.20 Журнал событий виджета

Во время загрузки и работы виджета он производит несколько типов событий. Их можно просмотреть в Журнале событий 398:

- В редакторе виджетов 423 журнал событий открывается автоматически при запуске виджета
- Во время работы виджета журнал можно открыть, нажав на иконку на инструментальной панели виджета.

События виджета

В журнале событий отображаются следующие типы событий:

ОШИБКА ЗАГРУЗКИ

Данное событие возникает в случае, если свойство или сам компонент не могут быть правильно декодированы из XML шаблона виджета.

Имя события loadError

Записи: 1

Формат 49 записи:

Имя поля	Тип поля	Примечания
component	строка	Имя компонента, загрузка которого не выполняется.
property	строка	Имя свойства или значение NULL, если весь компонент не может быть декодирован из шаблона.
error	строка	Сообщение об ошибке.

ОШИБКА ПРИВЯЗКИ

Данное событие ошибки запускается во время обнаружения ошибки привязки 1295, т.е. выражение привязки не может быть выполнено и результат не может быть корректно записан в цель привязки.

Имя события bindingError

Записи: 1

Формат 49 записи:

Имя поля	Тип поля	Примечание
target	строка	Цель 1295 привязки.

expression	строка	Выражение привязки.
activator	строка	Активатор привязки.
execution	строка	Режим выполнения привязки: при запуске, по событию или периодически.
error	строка	Текст ошибки.
stack	таблица данных	Трассировка источника ошибки.

13.4.21 Безопасность виджетов

Когда запускается виджет, все операции, выполняемые им, наследуют права доступа пользователя, запустившего его. Например, если пользователь Джо запускает виджет, принадлежащий администратору по умолчанию, виджет получает ограниченные права доступа Джо. Однако если администратор запускает виджет Джо, этот виджет получает неограниченные права доступа, представляя потенциальный риск безопасности, если Джо не является пользователем с полномочиями высшего уровня.

В отличие от систем автоматизации и контроля старого поколения, AtomMind не располагает настройками уровня интерфейса пользователя, чтобы избежать небезопасного контроля доступа со стороны клиента. Однако возможно сделать поведение интерфейса пользователя виджета зависимым от прав доступа текущего пользователя. Используйте функции языка выражений, чтобы проверить, доступны ли текущему пользователю определенные контексты и их переменные/функции/события и соответственно изменить поведение интерфейса пользователя (например, спрятать/выключить кнопки и т.д.):

- Используйте функцию `available()`, чтобы проверить, существует ли определенный контекст сервера и доступен ли текущему пользователю
- Используйте `hasVariable()`, `hasFunction()` и `hasEvent()`, чтобы определить, существует и доступна ли соответствующая переменная/функция/событие в определенном контексте

13.4.22 Производительность виджетов

Виджеты - это активные ресурсы, которые могут значительно влиять на производительность. "Движок" виджета - это его привязки.

Виджет, запущенный в AtomMind Client, может вызывать значительную нагрузку процессора и использование памяти на AtomMind Client, AtomMind Server или обоих.

Влияние на нагрузку процессора виджета - это сложение следующих показателей:

- Количество экземпляров виджета, запущенных системными операторами.
- Количество привязок виджета.
- Частота обработки привязок виджета. Она может быть явно определена в опциях привязки (для периодических привязок) или косвенно определена частотой событий или изменений состояний, которые вызывают выполнение привязок (для привязок При Событии).
- Сложность и влияние выражений привязки. Для получения более подробной информации см. производительность выражений.
- Влияние записи цели привязки. Запись цели привязки - это в большинстве случаев запись переменной контекста или вызов фнкции контекста. См. производительность переменных и производительность функций для получения более подробной информации.
- Влияние пользовательского кода Java скриптов виджетов.

Шаблоны виджетов кэшируются в памяти сервера, чтобы дать возможность быстрого запуска виджета. Большое количество сложных виджетов может вызывать постоянную значительную загрузку памяти сервера.

Запуск виджетов, которые имеют сотни сложных динамических компонентов (таких как таблицы, журналы событий, карты, графики или динамические векторные изображения), может вызывать постоянную значительную загрузку памяти со стороны клиента.

Запущенные виджеты могут также вызывать дополнительную нехватку памяти, если большие массивы данных (например, множество исторических событий) загружаются привязками виджета или графиками.

13.4.23 Примеры виджетов

AtomMind и производные продукты для вертикальных рынков содержат тысячи готовых виджетов, которые можно легко отредактировать в AtomMind GUI Builder, содержащем огромную базу данных различных примеров виджетов.

13.4.24 Перемещение виджетов между серверами

Часто возникает необходимость перенести один или более виджетов из одной инсталляции AtomMind Server в другую. Это может быть как перемещение на рабочий сервер с сервера разработки, так и частичная репликация конфигурации во время настройки распределенной инсталляции 1332.

Существует три основных способа переноса виджетов между серверами:

1. Копировать/переместить всю базу данных 692 с одного сервера на другой. Если сервер базы данных работал на отдельной машине, и старый AtomMind Server больше не будет использоваться, возможно просто остановить старый сервер и реконфигурировать новый сервер для подключения к той же базе данных. С переносом всей базы данных будут перенесены и все виджеты вместе со всей конфигурацией сервера.
2. Для переноса группы виджетов используйте действия Экспортировать 108/Импортировать 108 контекста Виджеты 1627. Это позволит за один раз перенести все или выбранные виджеты определенного пользователя 478.
3. По сути, виджет является XML шаблоном в сочетании с несколькими дополнительными параметрами (имя, описание и тип 946). Таким образом, можно также открыть XML шаблон виджета и отправить его на другой сервер (например, по e-mail). На целевом сервере потребуется создать новый виджет, настроить его имя/описание и другие параметры, а затем вставить XML шаблон, скопированный с исходного сервера.

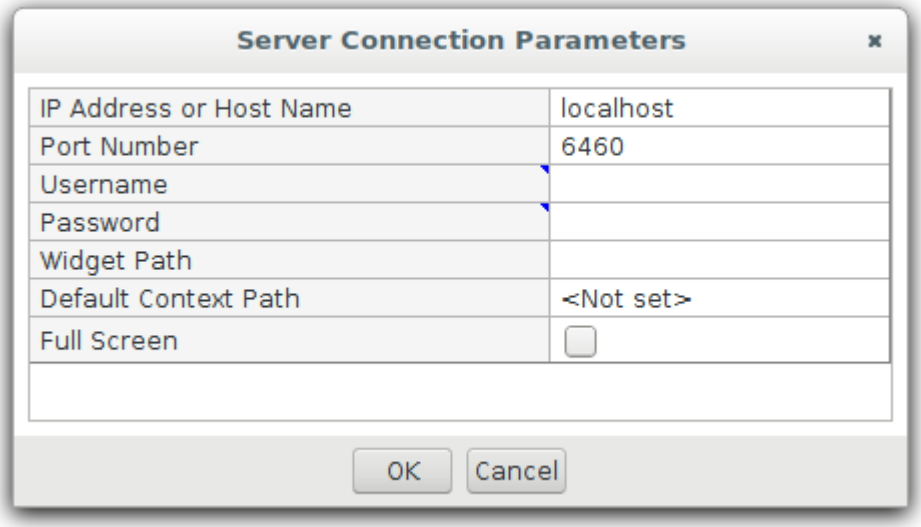
13.4.25 Проигрыватель виджетов

Проигрыватель виджетов - это средство, используемое для запуска виджета 943 как независимого приложения. Он необходим для:

- запуска виджета на сенсорной панели, подсоединенной к промышленному или домашнему контроллеру;
- запуска важных виджетов мониторинга без запуска самого AtomMind Client.

Диалоговое окно параметров

Если проигрыватель виджетов запущен без параметров командной строки, он открывает диалоговое окно, позволяющее определить настройки соединения/авторизации, путь виджета для запуска и путь его контекста по умолчанию 946:



Опции командной строки

Проигрыватель Виджетов запускается при помощи исполняемого файла `widget_player` (его расширение не зависит от операционной системы) со следующими параметрами:

```
widget_player [server_address server_port username password] widget_context_path [other_options]
```

- `server_address` - это IP или имя хоста AtomMind Server (например, localhost);
- `server_port` - это порт AtomMind Server, принимающий соединения с клиентом (`6460` по умолчанию);
- `username` - это имя AtomMind Server учетной записи пользователя 478, которое используется для входа на сервер;
- `password` - это пароль учетной записи пользователя AtomMind Server;
- `widget_context_path` - это полный путь контекста виджета для запуска, например `users.admin.widgets.myCustomWidget`;

Если Проигрыватель Виджета не может подключиться к AtomMind Server с определенными параметрами конмандной строки (или параметры командной строки не установлены), он отображает диалоговое окно авторизации, предлагающее войти в опции Проигрывателя Виджета.

ДРУГИЕ ОПЦИИ

`-c default_context_path`	Определяет полный путь контекста по умолчанию 946, который будет использоваться во время работы виджета, например `users.admin.devices.myDevice`. Этот параметр применим только к относительным 946 виджетам.
`-f`	Включает полноэкранный режим.
`-h`	Отображает подсказку к параметрам командной строки.

Полноэкранный режим

Любой виджет может быть увеличен на полный экран нажатием клавиши F11.

Полноэкранный режим отключается нажатием клавиши ESC или F11.

13.4.26 Веб Виджет Плеер

Веб Виджет Плеер - это веб-приложение, работающее в браузере и запускающее виджеты даже на системах, которые не поддерживают стандартную редакцию Java, включая:

- Android OS
- Apple iOS
- Windows Phone OS
- Любые другие операционные системы, которые имеют браузеры на основе Javascript

В настоящее время веб виджет плеер находится на стадии бета-тестирования. Пожалуйста, свяжитесь с ТВЭЛ, если вы испытываете проблемы с проигрывателем.

Веб виджет плеер не поддерживает никакие интерактивные действия.

Чтобы запустить виджет в производственной среде, пожалуйста, используйте настольное приложение Проигрыватель виджетов 1313.

Чтобы запустить веб-проигрыватель виджетов:

- Откройте центр веб-управления и выберите **веб-проигрыватель виджетов** из главного меню или
- Перейдите через браузер по `https://server.address:8443/widget` или `http://server.address:8080/widget`. Проверьте настройки веб-приложений 354, если у вас возникли проблемы с переходом на данные страницы
- Введите свои имя пользователя и пароль AtomMind Server

- Введите полный путь контекста 42 вашего контекста виджета 1450 (например, users.admin.widgets.my_widget) и контекст по умолчанию 946 виджета (только если ваш виджет относительный 946)
- Нажмите **Log In**. Виджет откроется в окне браузера.

- **ПРЯМЫЕ ССЫЛКИ НА ВИДЖЕТЫ**

Также есть возможность избежать ввода имени пользователя, пароля и других параметров. Может быть предоставлена прямая ссылка на виджет. Когда оператор щелкает по этой ссылке, виджет открывается напрямую. Ссылка должна выглядеть следующим образом:

```
http[s]://server.name[:port]/widget/login?
username=admin&password=admin&widget=users.admin.widgets.main[&context=users.admin.devices.hub]
```

Где **http[s]://server.name[:port]/widget/login** – это журнал проигрывателя виджетов в адресе страницы. Затем перечислены параметры:

- **username** - имя пользователя для входа
- **password** - парол пользователя
- **widget** - путь к контексту виджета
- **context** - путь к контексту по умолчанию (требуется для относительных виджетов 946)

Рекомендуется строго ограничить права доступа оператора, которому вы предоставляете прямые ссылки, потому что имя пользователя и пароль, находящиеся в URL - это обычный текст. Они могут быть использованы для получения доступа к серверу.

13.4.27 Карта устройств

Карта устройств является виджетом, используемым для представления динамически обновляемого статуса нескольких Device и любой другой дополнительной информации в графическом виде. Далее приведены примеры, объясняющие применение карт устройств:

- **Статические карты (карты здания/этажа, различные схемы, такие как план центра обработки данных)** для решений по управлению сетью, контролем доступа, безопасностью и видеонаблюдением. Подобная карта обеспечивает быстрый доступ к панелям управления,терминалам учета рабочего времени, камерам и другим устройствам по безопасности/автоматизации зданий.
- **Географические карты (карта региона/города/района)** для картографического отображения географически распределенных сетей устройств и управления движущимся транспортом. Такая карта показывает географическое расположение любого отслеживаемого устройства на картах Google/Bing/OpenStreet.
- **Карты топологии сети** для решений по управлению сетью. Такая карта показывает статус и доступность серверов, рабочих станций, роутеров, коммутаторов и другого сетевого оборудования.

Все виды карт отображают отношения устройство-устройству вдобавок к самим устройствам.

Статические карты

Статическая карта устройства - это стандартный виджет, состоящий из:

- Компонента корневая панель 1042 со свойством **фоновое изображение**, представленного растровым изображением в виде статической карты региона/района/здания/этажа.
- Нескольких компонентов устройств 1023, представляющих отслеживаемые аппартные средства. Каждый компонент устройства окрашивается в определенный цвет в зависимости от статуса устройства и дополняется метками, отображающими статус устройства.
- Кнопки 968 и ссылки, открывающие подробные карты.
- Любые другие компоненты, показывающие общую статистику и дополнительные данные: таблицы 979, графики 1051, метки 956 и т.д.

Чтобы использовать векторное изображение (например, SVG-изображение) в качестве фона статической карты, сделайте Многослойную панель 1049 базой своего виджета, добавьте к нижнему слою изображение Векторный рисунок 995 и поместите остальные компоненты (устройства, метки, ссылки и т.д.) в верхний слой.

- **МАСТЕР НАСТРОЙКИ ДЛЯ СОЗДАНИЯ СТАТИЧЕСКИХ КАРТ**

Карты устройств создаются при помощи действия Создать карту устройства 1624 контекста виджета. Как только создается виджет карты, его, как и любой другой виджет, можно отредактировать в Редакторе виджетов 423.

Новые устройства добавляются к карте перетаскиванием контекста Device из Селектора объектов 430 и помещением его в любой контейнер Рабочей формы 426 (см. подробности здесь 442).

Географические карты

Чтобы создать географическую карту:

- Задайте в свойствах 510 **Выражение Широты** и **Выражение Долготы** учетных записей ваших устройств статические значения широты/долготы (такие как 12.033234) или выражения, которые извлекают широту/долготу из данных устройства.
- Создайте новый виджет и добавьте к нему компонент Карта 1010. Настройте **Источник карты**, **Автоматическое определение уровня масштабирования**, **Тип** местоположения и **Геолокацию** центра карты или ее широту/долготу (если не использовать **Автоматический** центр карты как **Тип** положения).
- Установите **Отслеживаемые устройства** на `users.*.devices.*` или любую контекстную маску, соответствующую набору устройств, которые вы хотели бы видеть на карте.
- Настройте **Google Client ID/API Key** или **Bing API Key**, если необходимо.

Топологические карты

Чтобы создать топологическую карту:

- Обнаружьте топологию вашей сети и добавьте пользовательские ссылки в топологическую базу данных, если ваша карта будет базироваться на топологии сети L1/L2/L3. Пропустите этот шаг, если вы будете использовать **Пользовательский Поставщик** топологии.
- Создайте новый виджет и добавьте к нему компонент График 1025.
- Задайте свойство графика **Поставщик** к типу поставщика топологии, который вы хотите использовать.
- Задайте **Маску вершин**, соответствующую набору устройств, которые вы желаете видеть на карте.
- Настройте другие свойства Визуализации топологии 1300.

13.4.28 Поддержка сенсорного дисплея

Виджеты AtomMind часто запускаются на промышленном компьютере, оборудованном сенсорным дисплеем вместо обычной клавиатуры. Однако такие виджеты все же иногда требуют ввод с клавиатуры, например, пароля и различных цифр.

Такой ввод возможен при помощи поддерживаемой экранной клавиатуры. Некоторые компоненты виджета (например, Текстовое поле 958, Поле с форматированием 960 или Поле ввода пароля 959) могут отображать экранную клавиатуру при одинарном или двойном клике, позволяя операторам сенсорных дисплеев вводить любой текст, который будет вставляться в компонент.

Существует два варианта экранной клавиатуры:

1. Оригинальная клавиатура AtomMind, работающая в любой операционной системе

2. Приложение экранной клавиатуры в операционной системе или любое подобное сторонее приложение

Тип и стиль экранной клавиатуры выбираются в Настройках корневой панели виджета.

Каждый компонент, который должен отображать экранную клавиатуру, когда она выбирается на экранной панели, должен иметь настройку **Экранная клавиатура** с выбранным параметром **Одиночный клик** или **Двойной клик**.

14 Развертывание приложений и DevOps

Этот раздел описывает функции, относящиеся к разработке и развертыванию приложений и сервисов без программирования.

Программное расширение возможностей платформы описано в разделе Расширение и интеграция платформы 1339.

14.1 Приложения

Модуль Приложения позволяет упаковывать различные *ресурсы*, такие как тревоги 779, инструментальные панели 912 или модели 810, а также настройки сервера для их дальнейшего распространения через магазин 1321 AtomMind или Систему управления версиями (например, Subversion или Git).

Это полезно для:

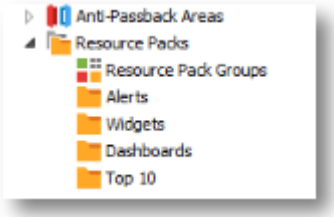
- доставки ваших приложений на серверы конечных заказчиков
- реализации коллективной разработки в различных IT ландшафтах - dev, test, RC, или production.

Приложение можно экспортировать как AtomMind Server плагин 207. После установки такого плагина, предоставляемые им ресурсы могут управляться 208 как любые другие ресурсы, включенные в AtomMind и производные коробочные решения.

Настройки и ресурсы приложений можно также экспортировать на/импортировать с дисковой папки, синхронизированной с Системой управления версиями. Эту папку можно зафиксировать в репозитории исходного кода для управления версиями и изменениями.

Управление приложениями

Для управления приложениями используются два контекста: первый - общий контекст Приложения 1462, который служит контейнером; второй- контекст Приложение 1463, который хранит информацию для отдельного приложения.



У каждого пользователя 478 имеется собственный набор приложений.

14.1.1 Упаковка ресурсов

Каждое приложение 1463 позволяет организовать и экспортировать предварительно сконфигурированные ресурсы и настройки сервера в следующих формах:

- **Как плагин 207 AtomMind Server**. Такой плагин может быть развернут на целевых серверах AtomMind, добавлен а магазин 1321 модулей и решений.
- **Как дисковая папка с настройками Приложения и шаблонами ресурсов**. Эта папка пригодна для дальнейших операций с Системой контроля версий, например, Subversion или Git.

Процесс упаковки

Экспортировать ресурсы при помощи приложений довольно просто:

- Разработайте и протестируйте ресурсы (т.е. виджеты или инструментальные панели) при помощи визуальных инструментов AtomMind.
- Создайте контекст приложения или используйте один из существующих.
- Используйте действие Сконфигурировать контекста приложения и выберите вкладку Ресурсы.
- Добавьте новые ресурсы и сконфигурируйте их, указав для каждого разработанного ресурса его подкатегорию, версию и зависимости в соответствующих полях.

- Сохраните изменения и используйте действие Упаковать 1463 для создания AtomMind Server плагина, либо используйте действие Экспорт 1464, чтобы сохранить настройки и ресурсы приложения в папке на стороне AtomMind Server.

14.1.2 Конфигурирование приложений

Описывает свойства конфигурации приложения.

Все описанные здесь свойства доступны при помощи действия Конфигурировать 105 в контексте Приложения 1599.

14.1.2.1 Свойства приложений

Следующие свойства определяют базовые параметры приложения.

Описание поля	Имя поля
Имя. Имя контекста приложения, необходимое для ссылки на данное приложение из других частей системы. Должно соответствовать соглашениям о наименованиях 42.	name
Описание. Текстовое описание приложения. Является также описанием 43 контекста приложения.	description
Путь к папке приложения. Папка AtomMind Server, используемая по умолчанию для импорта/экспорта файлов приложения.	applicationFolderPath
Комментарий. Любые подробности о приложении, такие как история версия и т.д.	comment

Получить доступ к данным свойствам можно через переменную childInfo 1466.

14.1.2.2 Ресурсы приложений

Ресурсы приложений определяются следующими свойствами:

Описание поля	Имя поля
Ресурс. Контекст ресурса, планируемого к упаковке.	resource
Описание ресурса. Описание ресурса, которое будет использоваться в таблице ресурсов 208 сервера. Это поле поддерживает локализацию 1320. Если указано **Использовать описание контекста**, Описание ресурса будет таким же, как и описание локального контекста, отмеченного параметром **Ресурс**.	description
Описание контекста. Описание, которое будет использоваться для наименования контекста, созданного во время развертывания приложения. Это поле поддерживает локализацию 1320. Если указано **Использовать описание контекста**, Описание контекста на целевом сервере (где развернуло приложение) будет таким же, как и описание локального контекста, отмеченного параметром **Ресурс**.	contextDescription
Категория. Категория, в которой будет размещен ресурс в таблице ресурсов 208. Это поле поддерживает локализацию 1320. Если указано **Использовать описание приложения**, Категория будет такой же, как описание самого Приложения.	category
Подкатегория. The subcategory in which the resource will be placed в таблице ресурсов 208. Это поле поддерживает локализацию 1320.	subcategory
Имя группы. Это поле определяет имя группы 751, в которой будет создан ресурс во время развертывания приложения. Если указано **Использовать описание**	groupName

приложения, Имя группы будет таким же, как имя контекста этого Приложения. Если указано **Не создавать группу**, ресурс не будет добавлен ни в какую группу во время развертывания приложения.	
Описание группы. Это поле определяет описание группы 751, в которой будет создан ресурс во время развертывания приложения. Это поле поддерживает локализацию 1320. Если указано **Использовать описание приложения**, Описание группы будет таким же, как описание самого Приложения.	groupDescription
Версия. Версия ресурса, целое число, начиная с 1, которое следует увеличивать при каждом изменении шаблона ресурса.	version
Зависимости. Таблица зависимостей, от которых будет зависеть текущий ресурс. Важно точно определить все зависимости. Каждый контекст из списка зависимостей должен существовать.	dependencies
Создать при первом запуске сервера. Ресурс будет создан при первом запуске сервера, если это не противоречит режиму создания ресурсов, выбранному при установке сервера.	createOnFirstServerLaunch
Защищенный. Защищенный ресурс не позволяет читать и изменять его настройки на целевом сервере, где развернуто приложение. Это помогает защитить приложения от декомпиляции. Обратите внимание, что защита ресурсов в настоящее время поддерживается только для моделей 810.	protected

Получить доступ к данным свойствам можно через переменную resources 1466.

Если приложение используется для взаимодействия с Системой контроля версий путем экспорта в папку на диске, будет применен только параметр **Ресурс**. Все остальные параметры в таблице ресурсов будут проигнорированы.

14.1.2.3 Локализация

Приложения можно интернационализировать и локализовать для разных стран/регионов, выполнив несколько шагов:

- настройка списка поддерживаемых языков
- описание идентификаторов уникальных текстовых ресурсов, используемых приложением
- предоставление значения идентификатора каждого текстового ресурса в каждом поддерживаемом языке
- ссылка на идентификаторы текстовых ресурсов из самого приложения

Если приложение правильно интернационализировано, оно будет локализоваться в соответствии с местным языком AtomMind Server после развертывания на определенном сервере.

конфигурирование локализации

Локализация настраивается ченез конфигурирование следующих свойств приложения:

Описание поля	Имя поля
Языки. Список поддерживаемых приложением языков.	languages
Словарь. Таблица со всеми константами ресурсов, используемыми приложениями, и их переводами на каждый поддерживаемый язык.	textResourceBundle

Получить доступ к данным свойствам можно через переменную localization 1467.

ССЫЛКА НА ЛОКАЛИЗОВАННЫЕ РЕСУРСЫ

Сослаться на идентификаторы текстовых ресурсов изнутри текстовых описаний ресурсов[1319] приложений возможно, используя формат `$R{resourceID}`, где `resourceID` - идентификатор любого текстового ресурса, добавленного в **Словарь** приложения. Соответствующие локализуемые свойства включают **Описание ресурса**, **Описание контекста**, **Категорию**, **Подкатегорию** и **Описание группы**.

Конфигурирование определенных типов ресурсов (например, виджетов[943]) также допускают ссылку на идентификаторы текстовых ресурсов при помощи формата `$R{resourceID}`.

14.1.2.4 Переменные контекстов

Переменные контекстов позволяют Приложению упаковывать значения переменных[61] из исходной системы (где создается Приложение) и переопределять эти переменные во время развертывания Приложения на целевом сервере.

Репликация переменных контекстов может быть полезной для переноса определенных частей общих настроек сервера, которые не могут быть упакованы как Ресурсы[1319], из исходной системы в целевую.

конфигурирование переменных контекстов

Переменные контекстов можно добавить к приложению, определив следующие свойства:

Описание поля	Имя поля
Контекст. Путь контекста переменных для пакетирования.	context
Переменная. Имя переменной для пакетирования.	variables

Получить доступ к данным свойствам можно через переменную contextVariables[1467].

14.1.2.5 Зависимости

Зависимости помогают указать, какие другие приложения требуются для данного приложения.

конфигурирование зависимостей

Зависимости можно добавить к приложению, определив следующие свойства:

Описание поля	Имя поля
ID приложения/плагина. ID другого приложения, необходимого для правильной работы данного приложения.	id

Получить доступ к данным свойствам можно через переменную applicationDependencies[1467].

14.2 Магазин приложений

Магазин приложений - это особый AtomMind Server, на котором работает модуль "marketplace" (Store Server), и который предоставляет решения и модули[1322] для других серверов AtomMind. Другие экземпляры AtomMind Server могут подключиться к Магазину и загрузить предлагаемые решения и модули при помощи модуля Store Client. Таким образом, Магазины - это способ централизованного распространения решений и модулей во всех больших экосистемах AtomMind.

Для разработчиков Магазины представляют способ доставки созданного контента конечным пользователям. Контент можно упаковать и разместить в Магазине. С этого момента он будет доступен для скачивания с любого AtomMind Server, подключенного к Магазину. Магазины также предоставляют поддержку версионности модулей. Каждый модуль иметь особую целевую платформу и иметь несколько версий.

Функционал Магазина обеспечивается плагином[207] **Store Client,** который скачивает решения и модули с серверов Магазина приложений с работающим плагином **Store Server**.

АО "ТВЭЛ" использует собственный публично доступный Store server, с которого можно скачать наши стандартные решения и модули на любой AtomMind Server партнера или заказчика. Этот публичный магазин зарегистрирован по умолчанию в любом Store Client и не требует никаких специальных действий для получения доступа к нему.

Наши крупные партнеры часто используют свои собственные Store Servers для распространения собственных решений и модулей среди заказчиков. Такие партнерские Магазины могут использоваться параллельно с публичным магазином АО "ТВЭЛ".

основные понятия

Магазин. Сервер, предоставляющий решения и модули. Магазины могут быть онлайновыми, или размещены локально. Онлайн Магазин - это отдельный AtomMind Server, к которому необходимо подключиться для получения решений и модулей. Локальный Магазин располагается на самом же AtomMind Server.

Решение. Пакет, содержащий несколько модулей. Обычно решение предлагает пользователю особый функционал или инструменты для решения бизнес задачи.

Модуль. Пакет, содержащий *ресурсы*: объекты, скрипты и другие файлы, преднастроенные для установки в определенную директорию на сервере. Модуль - это структурный элемент, дающий серверу особый функционал.

Версия. Каждый модуль имеет версию и целевую платформу. Версия определяет, на какой версии AtomMind Server можно использовать модуль. Платформа выбирается автоматически при установке. Один модуль может иметь несколько файлов для различных платформ.

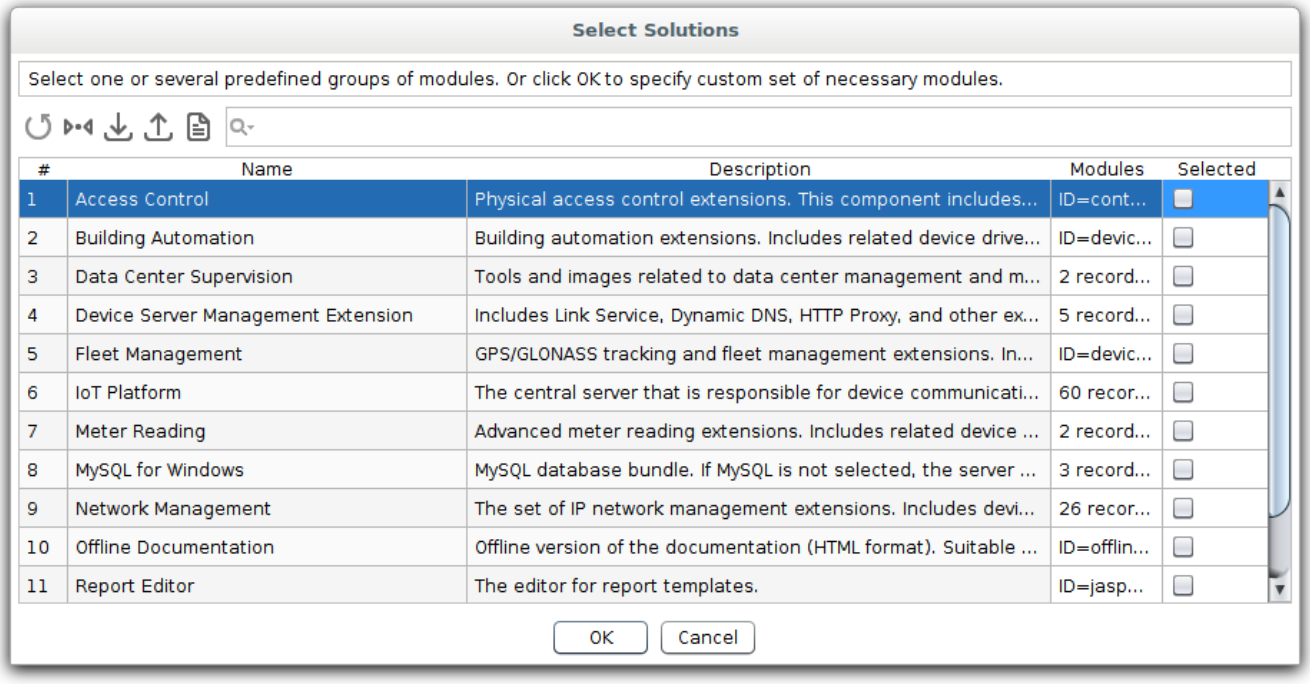
работа с магазинами

Этот раздел объясняет, как пользоваться магазинами.

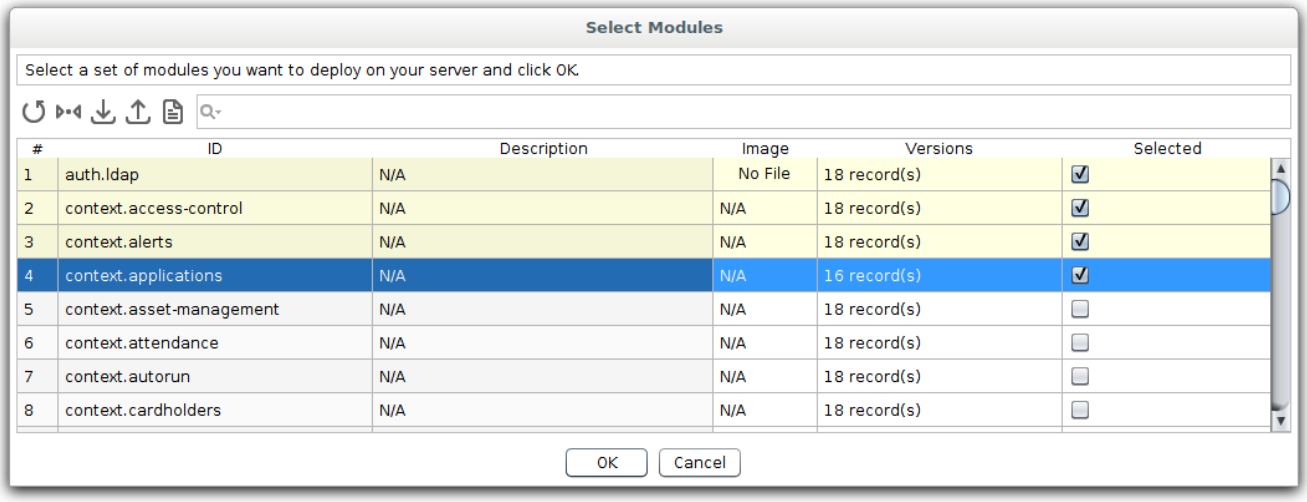
УСТАНОВКА МОДУЛЕЙ ИЗ МАГАЗИНА

Чтобы установить решение или модуль из магазина:

1. В Системном дереве 370 кликните правой кнопкой мыши на корневой контекст (Server).
2. Выберите действие **Установка модулей и решений**. Откроется диалоговое окно выбора магазина.
3. Выберите магазин - публичный или локальный, в зависимости от параметров настройки Магазина 205. Нажмите **ОК**.
4. Откроется диалоговое окно выбора решений. В нем показан перечень всех решений, доступных в магазине.



5. Выберите решения, которые вы хотите установить на ваш сервер. Если вы хотите установить отдельные модули, пропустите этот шаг. Нажмите **OK**.

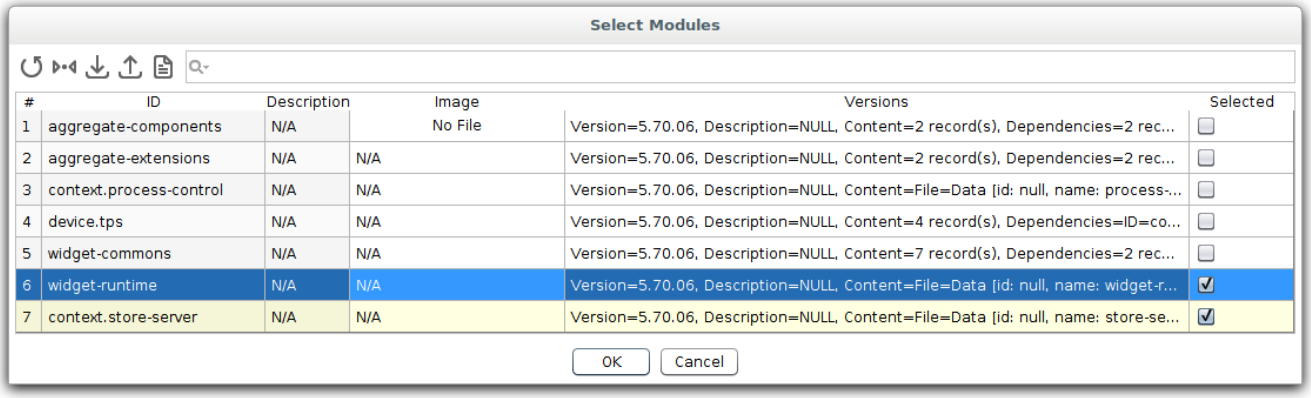


6. Откроется диалоговое окно со списком всех модулей, доступных в магазине.
7. Выберите все модули, которые вы хотите установить на ваш сервер. Если ранее вы выбрали решение, его модули будут выбраны автоматически.
8. Нажмите **OK**. Начнется процесс скачивания.
9. Скачанные модули установятся автоматически. После этого потребуется перезапустить сервер для активации модулей

УДАЛЕНИЕ МОДУЛЕЙ

Чтобы удалить модули:

1. В Системном дереве 370 кликните правой кнопкой мыши на корневой контекст (Server).
2. Выберите действие **Удаление модулей**.
3. Появится список установленных модулей.



4. Отметьте **Выбрано** для модулей, который вы хотите удалить. Нажмите **OK**.
5. Появится диалоговое окно со списком удаляемых модулей. Нажмите **OK**.
6. Выбранные модули будут удалены автоматически. Затем необходимо перезапустить сервер.

ДОБАВЛЕНИЕ МАГАЗИНА В СПИСОК МАГАЗИНОВ

Чтобы добавить новый магазин в список:

1. В Системном дереве 370 кликните правой кнопкой мыши на корневой контекст (Server).
2. Выберите действие **Настроить сервер**.
3. Перейдите во вкладку **Магазин**.
4. Во вкладке **Магазин**, перейдите во вкладку **Список магазинов**.

5. Укажите параметры для нового магазина. Более подробно см. опцию настройки Магазина 205.

СОЗДАНИЕ СОБСТВЕННОГО МАГАЗИНА

Чтобы добавить функционал магазина вашему серверу, установите плагин **Магазин,** который можно найти в магазине по умолчанию на AtomMind Server.

1. Следуйте инструкции из раздела Установка модулей из магазина 1322, чтобы установить модуль **context.store-server**.
2. Магазин автоматически добавится в список магазинов на вашем сервере как **Локальный магазин**. Если вы хотите дать доступ другим серверам к вашему магазину, добавьте ваш магазин в список, как описано в разделе Добавление магазина в список магазинов 1323.
3. Если вы хотите добавить в магазин собственные модули и решения, см. Публикация модулей и решений 1324.

ПУБЛИКАЦИЯ МОДУЛЕЙ И РЕШЕНИЙ

См. главу Публикация модулей и решений 1324.

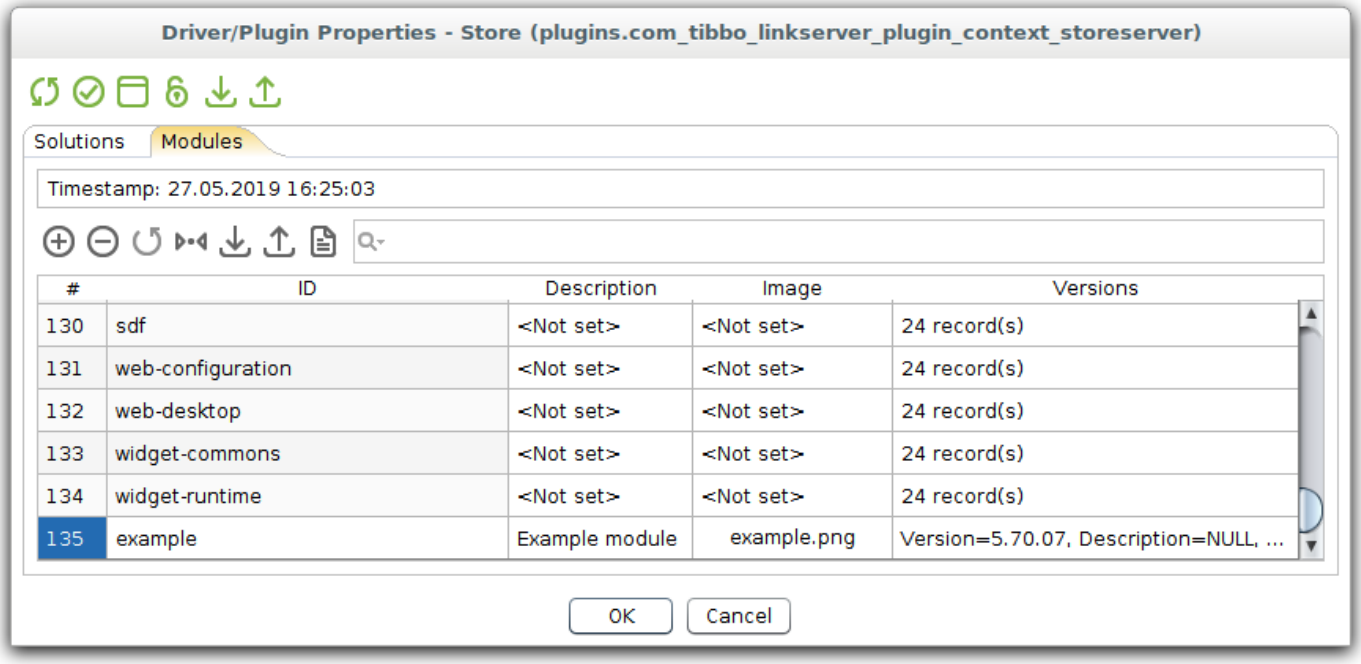
14.2.1 Публикация модулей и решений

Вы можете распространять свои модули и решения среди заказчиков при помощи магазинов. Например, если вы создали инструментальную панель, ее можно упаковать в модуль и разместить в магазине. Каждый, у кого есть доступ к этому магазину, сможет установить ваш модуль. Если позднее вы решите обновить инструментальную панель, вы сможете загрузить в магазин ее новую версию. Каждый, кто уже пользуется этой панелью, также сможет обновить ее.

Если вы хотите распространить созданные ресурсы, пройдите следующие шаги, чтобы опубликовать их в качестве модуля.

добавление модуля

Все доступные на сервере магазина модули хранятся в свойстве **modules** плагина **Магазин**.



Чтобы добавить модуль:

1. В Системном дереве 370 перейдите к **Драйверы и расширения**.
2. Кликните правой кнопкой мыши на плагин **Магазин**. Если этот плагин не установлен, см. Создание собственного магазина 1324.
3. В контекстном меню выберите действие **Редактировать свойства драйвера/расширения**.
4. Откроется диалоговое окно **Свойства драйвера/расширения**. Перейдите во вкладку **Модули**.
5. В тулбаре нажмите кнопку **Добавить строку** и укажите свойства нового модуля.

Определение свойств модуля

Свойства модуля хранятся в отдельных записях свойства **modules** плагина **Магазин**.

Имя свойства: **modules**

Тип свойства: **Data Table**

Каждая запись в таблице определяет один модуль:

- **ID (id)**. Идентификатор модуля.
- **Описание (description)**. Описание модуля.
- **Изображение (image)**. Иконка модуля.
- **Версии (versions)**. Версионное содержимое модуля.

ВЕРСИИ

Содержимое модуля входит в свойство **versions** свойста **modules**.

Имя свойства: **versions**

Тип свойства: **Data Table**

Каждая запись в таблице определяет содержимое модуля для определенной версии модуля:

- **Версия (version)**. Версия модуля. Единственная запись для текущей версии AtomMind Server будет создана автоматически. Можно добавить большее количество версий путем добавления записей к этому свойству.
- **Описание (description)**. Описание определенной версии данного модуля.
- **Содержимое (content)**. Содержимое модуля. В это поле входит список файлов с путями для установки. См. раздел Содержимое 1325 ниже.
- **Зависимости (dependencies)**. В это поле входит список других зависимостей для текущего модуля. См. раздел Зависимости 1325 ниже.
- **Скрипт, выполняющийся перед установкой (preInstallScript)**. Скрипт, который выполняется перед установкой модуля.
- **Скрипт, выполняющийся после установки (postInstallScript)**. Скрипт, который выполняется после установки модуля.
- **Скрипт, выполняющийся перед удалением (preUninstallScript)**. Скрипт, который выполняется перед удалением можуля.
- **Скрипт, выполняющийся после удаления (postUninstallScript)**. Скрипт, который выполняется после удаления модуля.

СОДЕРЖИМОЕ

Поле **content** свойства **versions** включает содержимое модуля. Оно содержит таблицу данных со следующими полями:

- **Файл (file)**. В поле находится содержимое файла.
- **Полное имя файла (path)**. Поле содержит путь, куда нужно установить соответствующий файл на сервере. Если это поле пустое, файл устанавливается в корневой каталог сервера.
- **Платформа (platform)**. Поле содержит значение целевой платформы. Это значение может использоваться, например, для создания специфичных для платформы версия каждого файла. После установки будет установлена версия для текущей платформы.

ЗАВИСИМОСТИ

Поле **dependencies** свойства **versions** содержит информацию о других модулях, от которых зависит данный модуль. Оно содержит таблицу данных со следующими полями:

ID (id). Идентификатор модуля.

Версия (version). Версия модуля.

СКРИПТЫ

У каждого модуля есть скрипты, которые могут выполняться до или после установки или удаления модуля. Эти скрипты - дополнительные свойства каждой версии модуля.

добавление решений

Все доступные на сервере магазина решения хранятся в свойстве **solutions** плагина **Магазин**.

Имя свойства: **solutions**

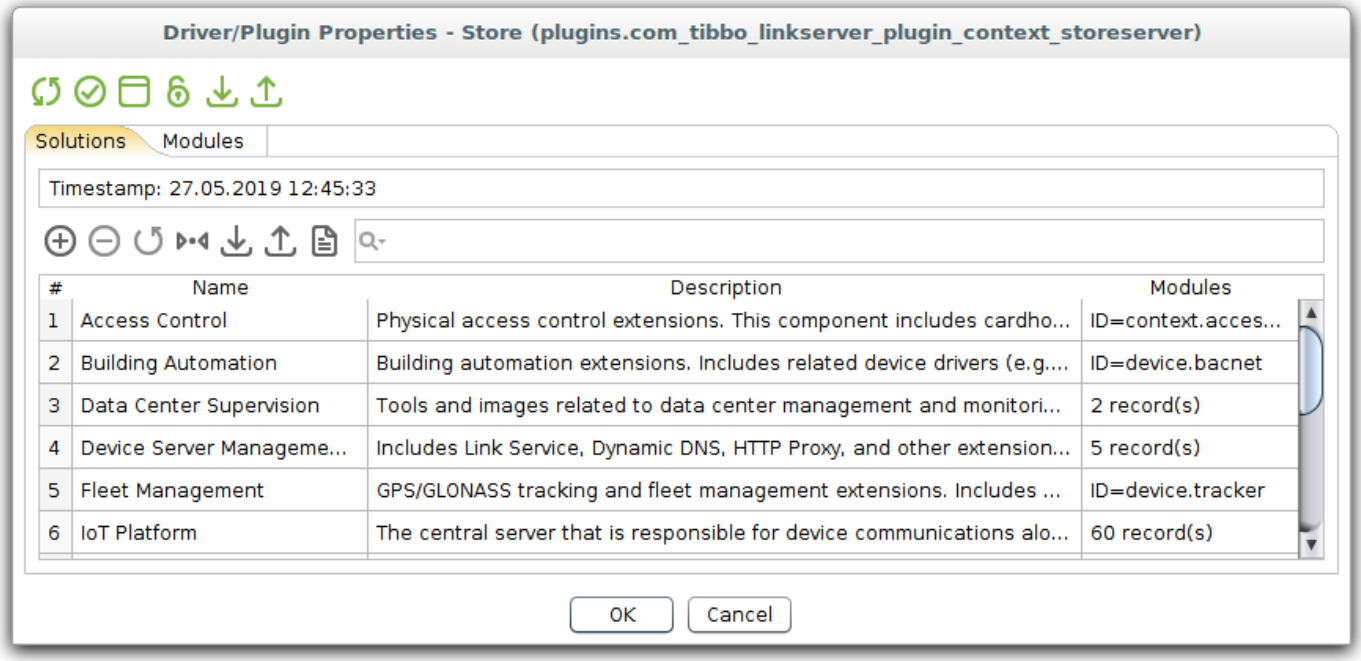
Тип свойства: **Data Table**

Каждая запись в таблице определяет одно решение:

- **Имя (name)**. Имя решения.
- **Описание (description)**. Описание решения.
- **Модули (moduleIds)**. Список модулей, входящих в решение.

Поле **moduleIds** содержит таблицу данных с идентификаторами модулей решения:

- **ID (id)**. Идентификатор модуля.



Чтобы добавить решение:

1. В Системном дереве 370 перейдите к **Драйверы и расширения**.
2. Кликните правой кнопкой мыши на плагин **Магазин**. Если этот плагин не установлен, см. Создание собственного магазина 1324.
3. В контекстном меню выберите действие **Редактировать свойства драйвера/расширения**.
4. Откроется диалоговое окно **Свойства драйвера/расширения**. Перейдите во вкладку **Решения**.
5. В тулбаре нажмите кнопку **Добавить строку** и укажите свойства нового решения.

14.3 Отказоустойчивый кластер

AtomMind имеет встроенную поддержку для выстраивания отказоустойчивого кластера, что позволяет обеспечить высокую доступность предоставляемых услуг. Отказоустойчивый кластер состоит из двух и более инсталляций AtomMind Server и одной или более инсталляций движка базы данных 692.

Высокая доступность - это системный подход и реализация предоставляемых услуг, которые обеспечивают должный уровень работы системы на протяжении договорного периода.

Отказоустойчивый кластер включает в себя два отдельных уровня, обеспечивающих высокий сервис доступности:

- Две или более инсталляции AtomMind Server (отказоустойчивый кластер AtomMind Server)
- Одна или более инсталляции базового движка базы данных 692 (отказоустойчивый кластер системы хранения)

Кластеры уровня приложения и БД полностью разделены.

Главный сервер AtomMind может быть запущен на одной и той же физической машине с "первой" инсталляцией кластеризованной/реплицированной базы данных, в то время как дублирующий сервер может делить аппаратное обеспечение со второй инсталляцией движка БД на второй машине. Это позволяет построить полноценное отказоустойчивое решение, используя лишь два физических сервера.

Может быть также кластеризованное приложение, работающее с некластеризованной БД или же наоборот, некластеризованное приложение, запущенное на кластеризованной БД.

Архитектура кластера AtomMind Server

Отказоустойчивый кластер позволяет достичь 100% доступности сервера. Кластер состоит из **Главного Узла** и одного или более **Дублирующих Узлов**. Во время обычной работы главный узел обслуживает все операции. Все дублирующие узлы запущены в режиме *standby* и отслеживают состояние главного узла.

Дублирующие узлы автоматически переключаются в режим **Дублирующего Главного** узла (т.е. активируются) в следующих случаях:

- Сбой питания или сети у главного узла
- Неполадки с техническим или программным обеспечением у главного узла

И главный, и дублирующий узлы представляют собой полноценную установку AtomMind Server, которые могут работать сами по себе. При этом главный и все дублирующие узлы связаны с одной и той же базой данных.

СВОЙСТВА КЛАСТЕРА ATOMMIND

У интегрированного в AtomMind движка отказоустойчивого кластера есть ряд уникальных свойств:

- Независимость от стороннего программного обеспечения или сервисов операционной системы, таких как Linux Heartbeat или Microsoft Cluster Service
- Зеркалирование базы данных опционально и может быть внедрено как посредством использования "родной" репликации, так и репликации средствами AtomMind.
- Узлы кластера могут быть запущены на различных ОС при разном техническом оснащении.

14.3.1 Отказоустойчивость AtomMind Server

Отказоустойчивый кластер AtomMind Server включает:

- Один **Главный сервер**
- Один или более **Дублирующих серверов**
- Кластеризованную или реплицированную БД, используемую всеми серверами

Отказоустойчивые серверы становятся активными при неполадках у главного сервера, например, вследствие:

- Сбоя сети
- Неполадок с оборудованием
- Сбоя ОС
- Неполадок AtomMind Server
- Нехватки дискового пространства
- Любой другой причины

Режимы работы дублирующего сервера

Дублирующие серверы могут работать в **Обычном режиме** или режиме **Только для чтения**. Разница между этими режимами разъясняется ниже. Режим контролируется общими настройками кластера в Режиме дублирования 194.

ОБЫЧНЫЙ РЕЖИМ ДУБЛИРОВАНИЯ

Если возник сбой в работе Главного сервера, дублирующий сервер или кластер AtomMind берут на себя полный контроль. Он контролирует и отслеживает устройства, соединения операторов сервиса и пр. Все изменения в конфигурации и события сохраняются в БД и доступны для Главного сервера, когда он вновь начнет работать.

РЕЖИМ ТОЛЬКО ДЛЯ ЧТЕНИЯ

В режиме только для чтения дублирующий сервер не выполняет изменения в основной БД. Его работа, на первый взгляд, аналогична работе Обычного дублирующего сервера: устройства контролируются, а их настройки конфигурации могут менять операторы, и все функции системы доступны. Однако изменения конфигурации и события не хранятся в БД. Это приводит к следующим ограничениям:

- События, полученные дублирующим сервером в режиме Только для чтения, будут недоступны, например, при просмотре событий или выстраивании графиков.
- Все изменения конфигурации будут утрачены, если Главный сервер возобновит работу или Дублирующий сервер будет перезапущен.

Дублирующие узлы в режиме только для чтения очень полезны для быстрого восстановления надежности кластера в случае **фатальной поломки Главного узла**:

- Сделать 1332 обычный дублирующий узел новым Главным узлом
- Сделать дублирующий узел только для чтения новым обычным дублирующим узлом, внеся изменения в Режиме дублирования 194 в общих настройках кластера.

На выполнение этих операций уйдет лишь несколько минут, и при этом функциональность и надежность кластера будет сохранена на случай другой неполадки нового Главного узла. После этого можно без спешки установить новый дублирующий узел только для чтения.

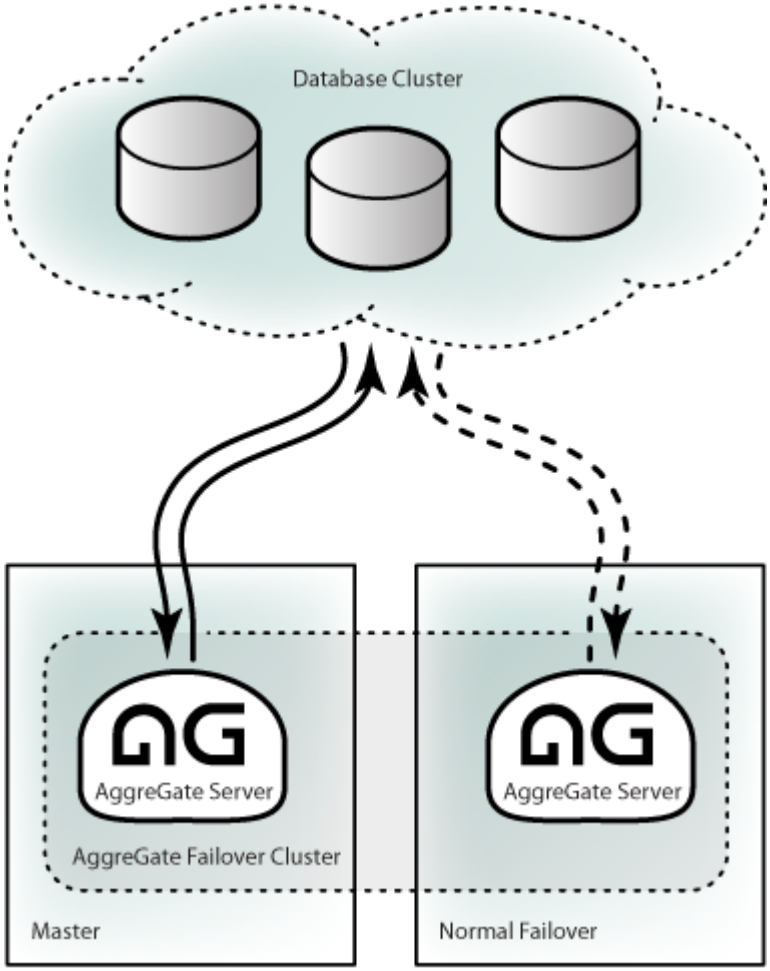
Лишь одному узлу кластера позволено работать в режиме нормального дублирования. Все остальные дублирующие узлы должны работать в режиме только для чтения.

Сценарии дублирования

В этом разделе описаны несколько конфигураций для типичных дублирующих кластеров. Обратите внимание, что кластер БД 710 изображен на рисунках в виде "облака". Фактически, базы данных, включенные в кластер БД, будут работать на одних и тех же физических серверах, что и инсталляции AtomMind Server.

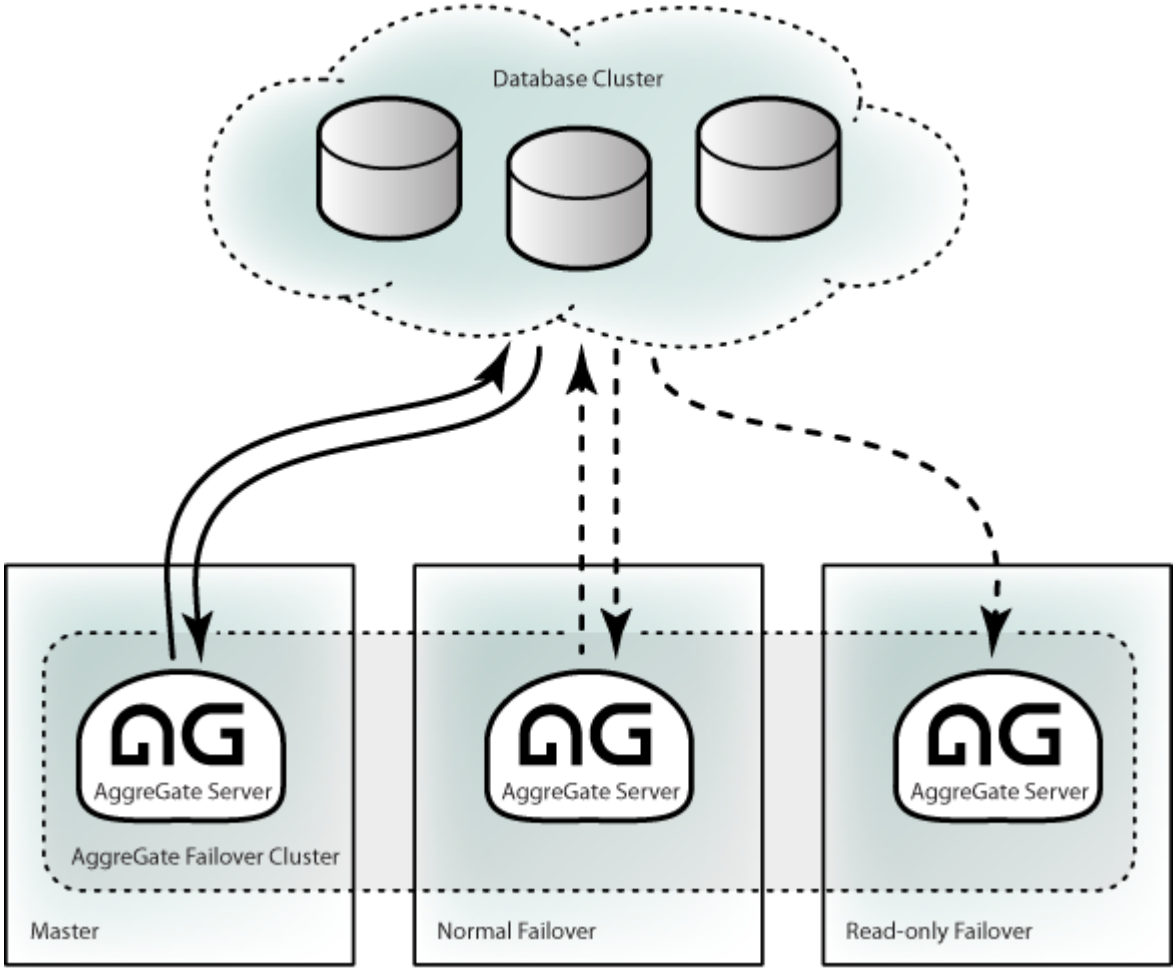
ДВА УЗЛА

Конфигурация обычного дублирующего узла включает в себя два сервера: Главный сервер и Обычный сервер для дублирования. При неисправности Главного, дублирующий сервер переключается в режим Главного, выполняя за него все операции.

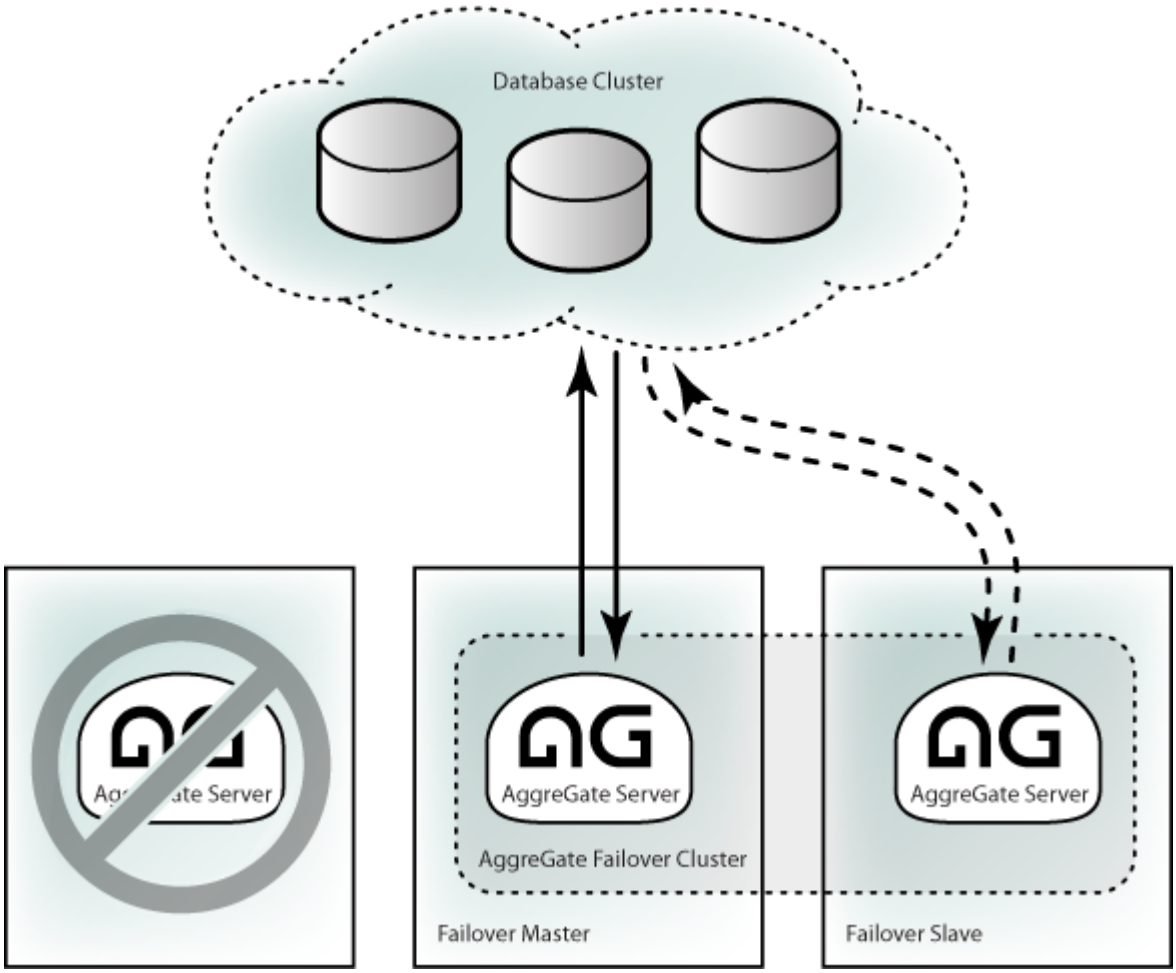


ТРИ УЗЛА

Кластер из трех узлов помогает сохранять надежность системы даже при сбое Главного сервера.



При сбое работы Главного сервера кластер из трех узлов будет работать аналогично тому, как работает кластер из двух узлов. Это позволяет обезопасить систему при сбое главного дублирующего сервера, во то время как у системного администратора появляется запас времени на то, чтобы восстановить работу трех узлов.



Установка отказоустойчивого кластера

Выполните следующие шаги, чтобы установить отказоустойчивый кластер AtomMind Server:

- Установите две или более копий AtomMind Server на разные физические сервера.
- Настройте хранилище данных 692, которое используется для каждого сервера, то есть настройте все серверы и единственную общую (возможно сгруппированную/реплицированную) базу данных.

> Использование единственной (сгруппированной и несгруппированной) базы даннях для всех узлов кластера AtomMind Server является абсолютным требованием.

- Измените настройку глобальной конфигурации роль кластера 194 одного сервера на **Master (главный)**. Измените роли других серверов на **Failover (дублирующий)**. Для получения деталей см. конфигурацию сервера 177.
- Измените отказоустойчивый режим 194 одного из отказойстойчивых серверов на **Normal**. Измените режим других отказоустойчивых серверов **Read-only**.
- Запустите все сервера в сервисном режиме 159.

Главный узел должен запустить нормальные операции, в то время как дублирующие узлы должны переключиться на режим ожидания, показывая сообщение "Проверка статуса главного сервера" на всплывающих таблчках.

Работа в режиме сбоя

Если главный узел выходит из строя, перестают выполняться регулярные обновления базы данных. Об отсутствии этих обновлений сообщают дублирующие узлы. Если обновление с главного узла не происходит за Время обнаружения сбоя узла 195, дублирующие узлы активизируются и начинают обслуживать обычные системные операции, такие как контроль устройств и действий оператора.

Интервал прерывания работы сервиса равен сумме времени обнаружения сбоя узла и периода активации дублирующего узла. Это обычно составляет меньше минуты.

Отключение дублирующих узлов

Если у дублирующего узла прервалась связь с кластером, например, в процессе обновления, работа кластера продолжается без изменений. Однако Главный сервер постоянно отслеживает доступность дублирующих узлов. Если дублирующий узел не реагирует в течении времени обнаружения сбоя узла, главный сервер отправит предупреждение в Контекст управления 1450.

Сигнал тревоги дублирующего узла

При неполадке главного сервера AtomMind в кластере дублирующий узел отправляет Сигнал тревоги 781. Это позволяет системному администратору быстро оценить ситуацию. По умолчанию администратору отправляется сообщение по e-mail. Однако мы рекомендуем настроить дополнительно и отправку SMS-сообщения, содержащего сигнал тревоги. См. Сигнал тревоги по SMS 793.

Назначение дублирующего сервера главным

В редких случаях главный сервер может полностью выйти из строя, например, из-за поломки аппаратного обеспечения. В этом случае необходимо сделать один из дублирующих узлов главным.

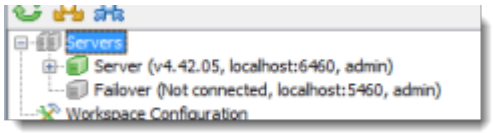
Чтобы сделать дублирующий узел главным, необходимо:

- Настроить новый дублирующий узел для сохранения общего числа узлов в кластере
- Указать в общих настройках для переключаемого узла Роль в кластере 194 как "главный"
- Перезагрузить новый главный узел

Настройка клиента с учетом дублирования

Чтобы подготовить AtomMind Client 359 к работе в кластере, необходимо:

- Создать два или более соединения к серверам 363 на Вашем рабочем пространстве: первое для главного сервера и остальные для дублирующих серверов. Указать адреса для серверов в настройках соединений.
- Отключить соединения с дублирующими серверами, чтобы не допустить ошибочного соединения при загрузке. Это необходимо, поскольку дублирующие узлы не примут соединения AtomMind Client , находясь в режиме ожидания.



- При неполадке главного узла (вы получите сообщения об ошибках соединения) просто активировать одно из соединений к дублирующему серверу.

Конфигурация Web UI для дублирующего узла

Когда главный узел кластера высокой доступности выходит из строя, системные операторы не могут зайти в Web UI 220, поскольку IP-адрес и имя хоста дублирующего узла отличается от адреса нерабочего главного узла. У этой проблемы есть два решения:

- Все операторы могут вручную набрать URL дублирующего узла. На этот случай URL можно сохранить в закладках в браузере.
- Можно настроить автоматическое переключение в DNS. Наберите в поисковой системе "*DNS failover*" и Вы найдете возможные решения. Вот одна из таких полезных ссылок: http://www.simplefailover.com/scenario3.aspx

Безопасность кластерных серверов

У всех серверов в кластере есть доступ ко всей информации, проходящей в AtomMind. Таким образом, для главного и дублирующего серверов предпринимаются одинаковые меры безопастности.

14.4 Распределенные операции

AtomMind входит в число немногих известных во всем мире систем мониторинга и управления устройствами, поддерживающих *распределенную архитектуру*. Подобная архитектура строится с целью безграничного масштабирования методами балансировки нагрузки между серверами AtomMind, разбитыми на множество уровней. Ожидается, что в-основном подобную архитектуру будет иметь ПО будущего.

В отличие от узлов отказоустойчивого кластера 1326 AtomMind, сервера, принимающие участие в распределенной системе, являются независимыми. Каждый сервер имеет свою собственную базу данных 692, профили пользователей 478, права доступа и пр.

Предназначение распределенных операций

Основными целями создания распределенной системы являются:

- **Масштабируемость**. Нижестоящие узлы могут подвергаться высокой нагрузке высокоскоростными протоколами обмена данными и частыми опросами устройств. На практике число устройств, которыми может управлять один сервер, ограничено несколькими тысячами. Чтобы увеличить производительность системы и обслуживать большее число устройств, необходимо установить несколько серверов и ввести их в единую группу распределенной системы.
- **Балансировка нагрузки**. Каждый сервер в распределенной инсталляции решает свои собственные задачи. Сервера управления сетью проверяют доступность и работоспособность сетевой инфраструктуры, в то время как сервера контроля доступа обрабатывают запросы с дверных и турникетных контроллеров. А контроль за деятельностью этих серверов и генерирование отчетов может выполняться центральным сервером.
- **Обход фаерволов**. Вторичные сервера могут устанавливаться на удаленных участках сети с подключением к центральному серверу. Системные администраторы подключают такие вторичные сервера из своих подсетей к центральному серверу через VPN или пробросом портов.
- **Централизация**. Вторичные сервера могут рабоать в полностью автоматизированном режиме, в то время как их общая операция будет контролироваться человеком посредством основного сервера, установленного в центральный командный пункт.

Распределенная архитектура идеально подходит для производственного контроля и операций настроек в большой мультисерверной инсталляции AtomMind. Что касается передачи данных между системными уровнями или узлами горизонтального кластера, то для этого должен использоваться драйвер Локальный агент 574.

ПРИМЕР 1: СИСТЕМА УПРАВЛЕНИЯ УМНЫМ ГОРОДОМ

Приведем в качестве примера многоуровневую конфигурацию AtomMind для управления и мониторинга **умным городом**:

- **Уровень 1**: оборудование (маршрутизаторы, контроллеры, промышленное оборудование и пр.)
- **Уровень 2**: серверы прямого управления (сервер управления сетью, сервер доступа, сервер автоматизации здания и пр.)
- **Уровень 3**: центральный сервер здания (один на дом, объединяет информацию со всех "специализированных" серверов Уровня 2 в этом здании)
- **Уровень 4**: сервера городских районов (конечные точки оповещений о событиях нижестоящих уровней, управляются людьми-операторами в режиме реального времени, интегрированы с системами CRM)
- **Уровень 5**: головной сервер (управление серверами районов, сбор отчетов и оповещений о событиях)

Каждый из серверов AtomMind в этой схеме, конечно, может являться многоузловым отказоустойчивым кластером 1326.

ПРИМЕР 2: МУЛЬТИСЕГМЕНТНОЕ УПРАВЛЕНИЕ СЕТЬЮ

Система AtomMind Network Manager, построенная на AtomMind - это один из наиболее типичных примеров использования распределенной архитектуры. Невозможно контролировать большие сегментированные сети корпораций и операторов дальней связи из одного места в связи с ограничениями маршрутизации, проблемами безопасности и ограниченной пропускной способностью между отдельными географическими сегментами.

Поэтому единая система мониторинга обычно состоит из нескольких компонентов:

- Главный или центральный сервер, объедняющий информацию из всех сегментов сети
- Вторичные сервера, производящие опрос устройств в изолированных сегментах
- Специализированные сервера, такие как сервера для анализа трафика, обрабатывающие миллиарды событий NetFlow в день

Вторичные и специализированные сервера действуют как поставщики данных для главного сервера, предоставляя часть своей модели данных центру управления. Это может быть:

- Все содержимое контекстного дерева вторичного сервера, предоставляющее полный мониторинг и конфигурирование через центральный сервер. В этом случаев вторичный сервер используется только как прокси для решения вопросов сегментации сети.

- Оповещения, сгенерированные вторичными серверами. В этом случае 99% работы выполняется удаленно, но операторы центрального сервера сразу же информируются о событиях, происходящих во вторичном сегменте.
- Пользовательский набор данных вторичного сервера, такой как устройства по решению критических задач и важные обзорные отчеты. Фактические опросы и генерирование отчетов осуществляются вторичным сервером, повзоляя правильно сбалансировать нагрузку системы

Плагин для распределенной архитектуры

Связь поставщик/потребитель в AtomMind Server реализуется плагином 207 **Распределенной Архитектуры**. Глобальный конфигурационный контекст 1506 этого плагина предоставляет доступ к:

- Настройкам поставщиков 1335
- Конфигурации потребителей 1337
- Статусу распределенных серверов 1337

14.4.1 Понятия распределенной архитектуры

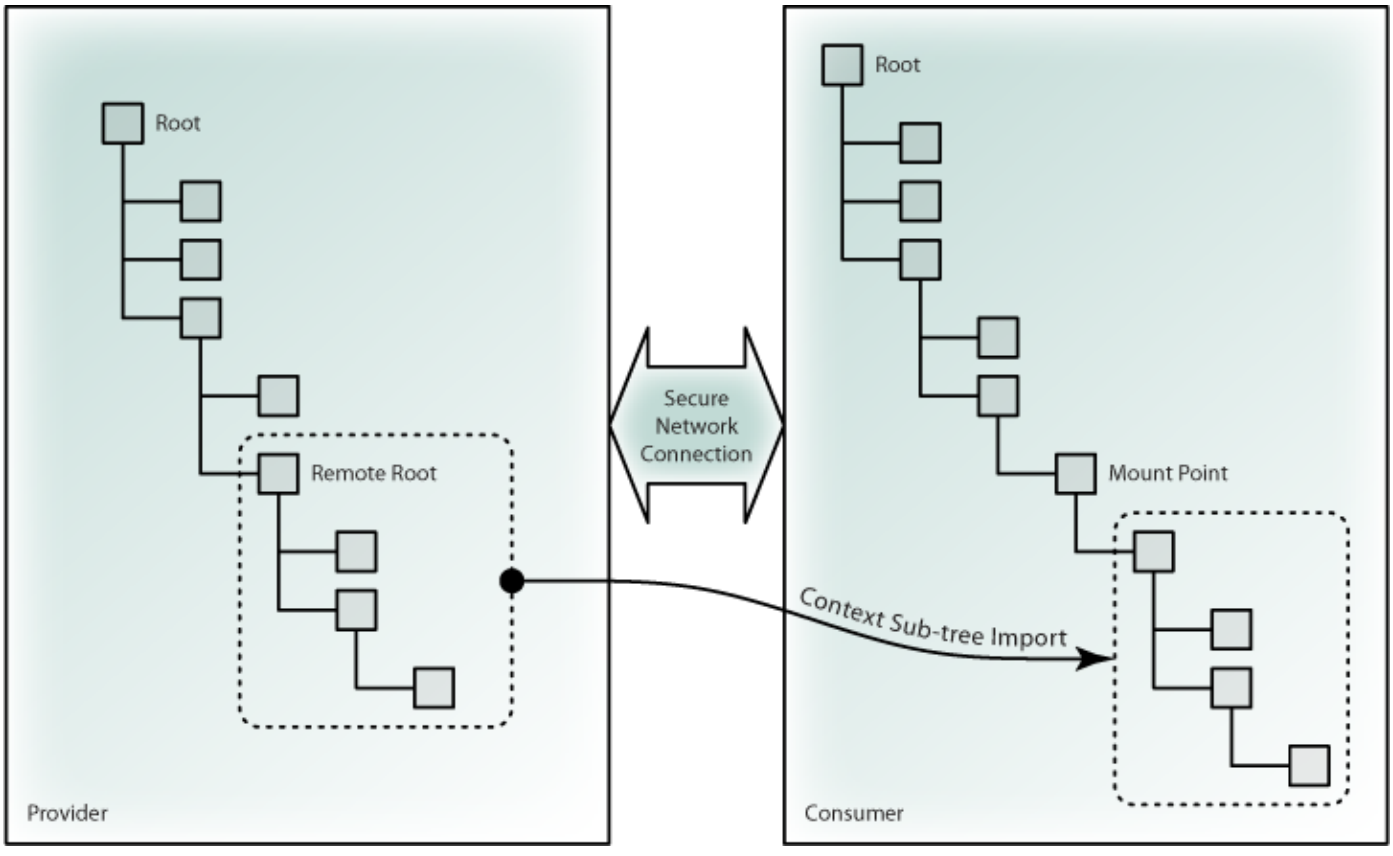
Все AtomMind Serverы имеют единую модель внутреннего представления данных: дерево контекстов 41. Каждый контекст в этом дереве отвечает за управление определённым устройством 497 или системным ресурсом, будь то оповещение 779 или отчёт 928.

В распределённой архитектуре каждый из серверов может работать как **поставщик** или **потребитель** данных для/с других серверов:

• **Потребитель** получает определенные поддеревья из деревьев контекстов других серверов и подключает их в собственное дерево контекстов (импортирует контексты с других серверов)

• **Поставщик** позволяет другим серверам использовать свои контексты (экспортирует контексты другим)

Контекст потребителя, который выступает как основа для привлечения контекстного поддерева поставщика, называется *Точкой Монтирования*. У каждого поставщика может быть несколько точек монтирования, т.е. несколько поддеревьев, подсоединенных к контекстному дереву потребителя.



Как только потребитель импортировал поддерево с другого сервера, собственные контексты этого потребителя могут взаимодействовать с импортированными контекстами. Вот несколько примеров:

- Если **сервер B** импортировал контекст устройства 1494 с **сервера A**, он может определить оповещение, которое проверяет некоторые метрики устройства и показывает оповещение о событии, если метрики выходят за определенные границы. При этом устройство будет опрошено сервером A, а оповещение будет обрабатываться на сервере B.

- Если **сервер B** импортирует контекст пользовательских аккаунтов 1608 с **сервера A**, системные операторы могут полностью контролировать все устройства и ресурсы этого пользователя через сервер B. При этом все модификации будут переданы серверу A и сохранены в его базе данных 692. Это может быть полезно, если операторы не имеют прямого доступа к серверу A, например, когда он находится в отдельной подсети.

- В конце концов, **сервер B** может импортировать все дерево контекстов с **сервера A**. В этом случае любой ресурс с сервера B может иметь доступ ко всем ресурсам сервера A, позволяя делать что угодно.

Взаимодействие поставщиков и потребителей

Все взаимодействие между потребителями и поставщиками происходит по IP-сетям с использованием шифрованных SSL-соединений. Соединения двунаправленные:

- Потребители могут подключаться к поставщикам и принимать входящие соединения от поставщиков.

- Точно так же поставщики могут подключаться к потребителям и принимать входящие соединения от них.

Двунаправленные соединения позволяют обходить различные сложности в конфигурации сетей и фаерволов.

Любой AtomMind Server может быть и потребителем и поставщиком одновременно. Более того, он может иметь множество связей, т.е. импортировать контексты с других серверов и экспортировать свои контексты другим или тем же серверам.

14.4.2 Настройка распределенной архитектуры

Настройка общей конфигурации распределённых операций AtomMind доступна через действие **Редактирование Опций Драйвера/Плагина** в общем конфигурационном контексте 1506 плагина **Распределённая Архитектура**.

Следующие опции доступны для настройки:

- **Номер порта для приема подключений поставщиков**. Это порт, на который будут подключаться поставщики. Должен совпадать с номером порта, указанным в таблице потребителей 1337 на удаленном сервере.

- **Номер порта для приема подключений потребителей**. Порт для входящих соединений от потребителей. Должен совпадать с номером порта, указанным в таблице поставщиков 1335 на удаленном сервере.

- **Время ожидания команды**. Время ожидания команды в соединении между поставщиком и потребителем.

- **Максимальная длина очереди исходящих событий.** Количество событий, которые могут быть помещены в очередь до ее переполнения. Все события сверх этого количества будут отбрасываться, пока длина очереди не уменьшится.

- **Максимальная длина очереди входящих событий.** Количество событий, которые могут быть получены и помещены в очередь до ее переполнения. Все события сверх этого количества будут отбрасываться, пока длина очереди не уменьшится.

14.4.2.1 Настройка поставщиков

Таблица настройки поставщиков доступна через действие **Редактирование опций Драйвера/Плагина** в общем конфигурационном контексте 1506 плагина **Распределённая Архитектура**. Каждый поставщик обладает следующими свойствами:

- **Доступен**. Включает или отключает провайдера.

- **Имя**. Уникальное имя поставщика. Имя должно удовлетворять соглашениям о наименовании контекстов 42.

- **Тип соединения**. При **Входящем** типе соединения потребитель ждет и принимает TCP-соединения от провайдеров. При **Исходящем** типе соединения потребитель соединяется по TCP с поставщиком.

- **Адрес**. IP-адрес сервера-поставщика. Для **Исходящих** соединений.

- **Порт**. Номер порта, к которому необходимо подключиться, чтобы получить доступ к поставщику. Для **Исходящих** соединений.

- **Использовать сжатие**. Активирует сжатие данных между серверами. Для **исходящих** соединений.

- **Имя пользователя**. Имя учетной записи пользователя 478 для авторизации на сервере.

- **Пароль**. Пароль к учетной записи пользователя для авторизации на сервере.

- **Точка монтирования**. Список удаленных поддеревьев контекста или отдельные контексты, прикрепляемые к локальному дереву контекстов. Свойства точки монтирования описаны ниже.

Настройка точки монтирования

- **Имя**. Когда корневой контекст удаленного поддерева контекста прикрепляется к локальному дереву, его локальное имя будет отличаться от удаленного имени. Это удаленное имя определяется настройками **Имени** точки монтирования.
- **Локальный путь (Точка монтирования)**. Путь контекста, действующий как "точка соединения" для контекстов, импортируемых из севера-поставщика. Это путь на локальном (потребительском) сервере.
- **Удаленный путь (Корневой узел импортируемого поддерева контекста)**. Путь корневого контекста контекстного поддерева, импортируемый из сервера-поставщика. Это путь на удаленном сервере (поставщика).

ПРИМЕР 1

Допустим, сервер-поставщик имеет следующие контексты:

A

A.B

A.B.C1

A.B.C2

Потребитель желает импортировать контексты поддерева A.B. Импортированное поддерево должно подключаться к пути X.Y контекста поставщика.

Для этого мы будем использовать следующие опции конфигурации точки монтирования:

- **Имя**: P
- **Локальный путь**: X.Y
- **Удаленный путь**: A.B

Как только соединение между провайдером и потребителем установится и будет произведена операция импорта, мы получим такие связи между путями в контекстах:

Путь к контексту на сервере-поставщике	Получившийся путь к контексту на сервере-потребителе
A.B	X.Y.P
A.B.C1	X.Y.P.C1
A.B.C2	X.Y.P.C2

Может показаться более логичным, чтобы пути на сервере-потребителе начинались с X.Y.B. Однако имя корневого импортированного контекста подменяется именем провайдера (B подменяется на P). Это необходимо для соблюдения уникальности имен.

Например, потребитель имеет контекст "admin" в "users" и желает импортировать контекст "admin" с поставщика, используя "users" как точку монтирования. Мы можем назвать поставщика "provider_admin", и в результате импорта получим путь "users.provider_admin".

Подстановка имен контекста также позволяет импортировать удаленный корневой контекст (путь к которому представляет собой пустую строку).

ПРИМЕР 2

Этот пример объясняет, как подсоединить пользователя 478 удаленного сервера к Контейнеру пользователя 1604 локального сервера, чтобы удаленный пользователь действовал на уровне локального. В этом случае сервер-потребитель (тот, который размещает удаленного пользователя) сам подсоединяется к серверу-поставщику классическим способом (как клиент посоединяется к серверу).

Чтобы настроить изложенный вариант использования, добавьте следующую запись в таблицу **Поставщиков** на сервере-потребителе:

Имя	Любое имя соединения, например remoteUserProvider.

Тип соединения	`Outgoing`
Адрес	Адрес сервера-поставщика
Порт	`6420`
Имя пользователя	Имя пользователя для авторизации на сервере-поставщике.
Пароль	Пароль для авторизации на сервере-поставщике.
Точка монтирования	Одна точка монтирования со следующими параметрами: • **Имя**: Имя, дающееся удаленному контексту пользователя, например `importedUser`. • **Локальный путь контекста (Точка монтирования)**: Контекст локальных пользователей, например `users`. • **Удаленный путь контекста (Корневой узел импортируемого поддерева контекста)**: Полный путь удаленного контекста пользователя, например `users.exportedUser`.

14.4.2.2 Настройка потребителей

Таблица потребителей определяет исходящие соединения к серверам-потребителям. Как только сервер-поставщик подключается к удаленному серверу-потребителю, он указывает последнему, какая запись в таблице провайдеров 1335 должна использоваться для импортирования контекстов с этого сервера.

Таблица потребителей доступна через действие **Редактирование опций Драйвера/Плагина** в общем конфигурационном контексте 1506 плагина **Распределённая Архитектура**.

Каждый потребитель имеет следующие свойства:

- **Имя**. Имя поставщика в таблице поставщиков 1335, используемое на удаленном сервере-потребителе.
- **Адрес**. Адрес сервера потребителя.
- **Порт**. Номер порта сервера-потребителя, куда необходимо производить подключение.
- **Использовать сжатие**. Активирует сжатие данных между серверами.

14.4.3 Статус распределенных серверов

AtomMind предоставляет доступ к информации о статусе подключенных поставщиков и потребителей. Чтобы увидеть статус, используйте действие **Просмотреть Статус** в общем конфигурационном контексте 1506 плагина **Распределённая Архитектура**. Там две таблицы состояний:

- **Состояние Поставщиков**. Показывает состояние серверов, чья информация импортируется в данный сервер.
- **Состояние Потребителей**. Показывает состояние серверов, которые импортируют данные с данного сервера.

Состояние поставщиков

Таблица состояния поставщиков содержит следующие поля:

- **Имя**. Имя поставщика.
- **Тип подключения**. Входящее или исходящее.
- **Активный**. Показывает, активен ли данный поставщик.
- **Подключен**. Показывает, подключен ли в настоящий момент этот поставщик.
- **Адрес**. IP-адрес сервера-поставщика.

Состояние потребителей

Таблица состояний потребителей содержит следующие поля:

- **Имя**. Имя локального поставщика, к которому данный потребитель подключен.
- **Тип подключения**. Входящее или исходящее.

- **Активный**. Показывает, активен ли данный поставщик.
- **Адрес**. IP-адрес сервера-потребителя.
- **Длина очереди исходящих событий.** Текущее количество неотправленных событий.
- **Количество отброшенных событий.** Количество неотправленных событий, отброшенныхиз-за переполнения очереди на отправку.

14.4.4 Контроль доступа в распределенной среде

Как только устанавливается соединение между сервером-потребителем и сервером-поставщиком, сервер-потребитель авторизуется на сервере-поставщике, используя определенные параметры доступа пользователя 478. Таким образом, сервер-потребитель получает права доступа 477 этого пользователя на сервере-поставщике. Все операции на сервере-поставщике, начатые сервером-потребителем (или людьми-операторами сервера-потребителя), производятся в соответствии с этими правами доступа.

Проще говоря, для поставщика потребитель является обычным Клиентом 359, который подключился и вошел в систему.

Однако если операторы сервера-потребителя хотят получить доступ к контекстам, импортируемым из сервера-поставщика, они должны подключиться к серверу-потребителю и сначала авторизоваться как некий пользователь сервера-потребителя.

Теперь обобщим вышесказанное. Представьте, что сервер B (потребитель) импортирует контекст "context123" из сервера A (поставщика). *Локальный путь* этого контекста на сервере B - это "mount.context123" (в соответствии с настройками поставщика). Сервер B авторизуется на сервере A как пользователь "userA" (определенный на сервере A). Человек-оператор подключается к серверу B и хочет выполнить операцию с импортируемым контекстом. Он авторизуется на сервере B как пользователь "userB" (конечно, определенный на сервере B). Вот необходимые права доступа:

- Пользователь "userB" на сервере B должен иметь права доступа к контексту "mount.context123" на сервере B.
- Пользователь "userA" на сервере A должен иметь права доступа к контексту "context123" на сервере A.

15 Расширение и интеграция платформы

В этой главе говорится об интеграции AtomMind в производственную инфраструктуру и ее взаимодействии со сторонними приложениями.

Информация, предоставленая в этой главе подразумевает хорошее понимание Единой модели данных 41 AtomMind.

15.1 Комплект разработчика (SDK)

SDK (Software Development Kit) для *AtomMind* - комплект разработчика с открытым кодом доступа, который позволяет расширять Платформу AtomMind, а также интегрировать ее с другими производственными системами. SDK AtomMind - это набор широкодоступных модулей, которые обеспечивают совместимость с разными операционными системами. Модули реализованы на языке Java. Существует две различных версии Java SDK для AtomMind:

- Java SDK AtomMind для стандартной версии Java
- Java SDK AtomMind для Android (Dalvik JVM)

SDK AtomMind включает в себя следующие компоненты:

- Прикладной программный интерфейс (API) AtomMind Server 1340 для управления AtomMind Server и аппаратным оборудованием из других приложений через безопасное сетевое соединение.
- Программный комплект разработчика драйверов (DDK) AtomMind Server 1345 для подсоединения новых аппаратных устройств к системе при помощи пользовательских Драйверов Устройства 518
- Плагин SDK AtomMind Server 1342 для расширения AtomMind Server через обработку новых данных и презентацию модулей.
- Agent SDK 1360 для внедрения Агентов, запускаемых на контроллерах, которые базируются на ПК.

Драйвер и Агент

Нас часто спрашивают, в чем разница между драйвером устойства 518 и Agent на базе Java. Приведем краткое объяснение:

- Код драйвера устройства Java - это часть AtomMind Server, он выполняется на серверах Виртуальной Машины Java. Агент - это автономное приложение Java, которое запущено на отдельном ПК.
- Драйвер устройства использует родной протокол (обычно на основе IP или последовательной связи) для взаимодействия с устройствами, в то время как Агенты общаются с AtomMind Server, используя Проткол AtomMind 2119.

Дистрибутивный пакет SDK для AtomMind

SDK для AtomMind доступен в виде архивированного файла в формате ZIP. который содержит:

- Исходный код (папка `/src`)
- Примеры (папка `/src/examples`)
- Тесты элементов (папка `/test`)
- Функциональные тесты (папка `/test`)
- Javadocs (`/docs` folder)
- Заготовка Java Archive (JAR) с классами SDK (`aggregate-api.jar`)
- Необходимые сторонние библиотеки (`aggregate-api-libs.jar`)

Использование SDK AtomMind

Чтобы использовать SDK AtomMind, Вам необходимо добавить следующие java-архивы (JARs) в путь к классам для Вашего Java-приложения:

- **aggregate-api.jar**
- **aggregate-api-libs.jar**

Первый архив содержит специфичный для AtomMind код, в то время как второй архив содержит сторонние библиотеки, используемые SDK.

JAVADOCS И ИСХОДНИКИ

Эта часть руководства содержит функциональную документацию лишь для верхнего уровня и также пошаговую инструкцию по использованию различных аспектов SDK AtomMind. Дополнительную информацию о конктетных классах и интерфейсах, включенных в SDK, можно найти в:

- *Javadocs,* расположенном в папке `/docs` дистрибутивного пакета
- *Исходные тексты* - в папке `/src` дистрибутива

Версия Java

Большинство дистрибьютивов AtomMind Server и AtomMind Client включают объединенную виртуальную машину Java. Любая виртуальная машина Java, установленная на ОС, не используется по умолчанию. Поэтому любые пользовательские модули должны быть скомпилированы для версии Java, которая используется сервером/клиентом. Эту версию можно найти в системных требованиях (требования сервера 146, требования клиента 359).

15.1.1 API сервера

API с открытым исходным кодом AtomMind Servera для Java (Java API (AtomMind Servera) позволяет контролировать, конфигурировать и отслеживать AtomMind Server, а также все устройства аппаратного оборудования, которые работают с одной инсталляцией AtomMind удаленно с любого приложения, написанного на языке Java.

Используя этот API, можно:

- Иметь доступ к ресурсам сервера и связанным устройствам;
- Изменять настройки сервера и устройства;
- Выполнять операции на сервере и устройствах;
- Получать события с сервера и устройств;

Технически Java API предоставляет следующую функциональность:

- Полный доступ к контекстам 41 сервера через так называемое *proxy дерево контекста;*
- Получение и настройка значений переменных 61 контекста 41;
- Вызов функций 70 контекста;
- Прослушивание событий 73 контекста;
- Создание и обработка Таблиц данных 49;
- Выполение действий 87 контекста;

Всё взаимодействие с AtomMind Server выболняются по IP, через одно защищенное SSL соединение TCP, используя протокол взаимодействия AtomMind 2119.

Использование API AtomMind Servera

Этот параграф предоставляет пошаговую инструкцию для установления связи с удаленным AtomMind Serverом из Java-приложения.

СОЗДАНИЕ СОЕДИНЕНИЯ С СЕРВЕРОМ

Начните с создания объекта `RemoteLinkServer`:

```
RemoteLinkServer rls = new RemoteLinkServer("localhost",
RemoteLinkServer.DEFAULT_PORT, "admin", "admin");
```

Вам необходимо определить следующие параметры в конструкторе `RemoteLinkServer`:

- IP-адрес имени хоста удаленного AtomMind Servera
- Номер порта для установления соединения (6460 по умолчанию)
- Имя учетной записи пользователя 478 для авторизации (имя пользователя)
- Пароль для учетной записи пользователя

Затем создайте `RemoteLinkServerController`. Он будет использоваться для управления соединением с сервером:

```
RemoteLinkServerController rlc = new RemoteLinkServerController(rls, true);
```

Установите соединение с сервером посредством вызова метода connect():

```
rlc.connect();
```

Соединение не установится, если:

- Сервер не запущен;
- Не активирован API удаленного сервера;
- Адрес/порт указаны неверно;
- Версии AtomMind Server и API библиотеки не совместимы.

На этом этапе у Вашего приложения есть установленное по TCP соединение, подготовлено шифрование соединения по SSL и произошел обмен основной информацией 2121 с AtomMind Serverом. Чтобы выполнять операции, необходимо авторизоваться на сервере и получить уровень доступа. Это выполняется путем вызова метода `login()` контроллера сервера:

```
rlc.login();
```

Вызов не удастся, если логин/пароль, указанные в конструкторе `RemoteLinkServer` неверны.

ПОЛУЧЕНИЕ ДОСТУПА К МЕНЕДЖЕРУ КОНТЕКСТОВ

Теперь можно получить экземпляр `ContextManager` из контроллера. Это дерево proxy-контекста, структура и операции с которым очень похожи на удалённое дерево контекста AtomMind Servera:

```
ContextManager cm = rlc.getContextManager();
```

РАБОТА С КОНТЕКСТАМИ

Все остальные важнейшие операции по управлению данными AtomMind Server, выполняются посредством вызова операций переменной/функции/события в различных проксируемых контекстах , предоставляемых менеджером прокси-контекста.

См. работа с контекстами 1378 для получения более подробной информации.

РАЗЪЕДИНЕНИЕ С СЕРВЕРОМ

Вызовите метод `disconnect()` контоллера AtomMind Server, чтобы закрыть соединение.

Соответствующие классы

Класс	Описание
`AbstractAggreGateDeviceController`	Общий клиентский контроллер сеанса по протоколу AtomMind.
`AggreGateDeviceController`	Интерфейс клиентского котроллера сеанса по протоколу AtomMind.
`RemoteConnector`	Базовый интерфейс котроллера сеанса по протоколу AtomMind.
`RemoteServer`	Параметры подключения удаленного сервера.
`RemoteServerController`	Контроллер сеанса подключения удаленного сервера.

Соответствующие примеры

Пример	Описание
`/src/examples/api/GetServerVersion.java`	Этот простой пример показывает, как удаленно подключаться к AtomMind Server при помощи API AtomMind Servera и получать номер версии сервера.
`/src/examples/api/ManagerUsers.java`	Этот примеры показывают, как: • Вывести учетные записи пользователей 478 и просмотреть информацию о них • Создать, обновить и удалить учетные записи

/src/examples/api/ManagerDevices.java	Этот пример показывает, как: • Перечислить доступные учетные записи устройства 497 и просмотреть/изменить их свойства • Создать и удалить учетные записи устройства • Просмотреть статус устройства и выждать, пока Device не перейдет в определенное состояние • Вывести, прочитать и записать настройки устройства • Выполнить операции устройства • Получать и обрабатывать события устройства.
/src/examples/api/ExecuteAction.java	Это усложненный пример, который показывает, как выполнять действие 87 на сервере, симулируя ввод данных человеком-оператором.

15.1.2 Отчеты об ошибках

Самый простой способ отправки отчета об ошибке или сообщении из драйвера или плагина AtomMind - это использование стандатной функции журналирования 166. AtomMind использует библиотеку Log4j для журналирования. Для журналирования сообщения требуется всего одна строка кода Java:

```
Logger.getLogger("ag.core").info("A message");
```

или

```
Logger.getLogger("ag.device.acme").error("Unexpected error", exception);
```

Документация библиотеки Log4j предлагает комплексную документацию всех аспектов конфигурирования и реализации журналирования.

Класс `Log` обеспечивает заранее определенные устройства регистрации общими категориями журналирования AtomMind. Вот пример использования:

```
Log.CORE.info("A message");
```

Отчет об ошибках через события AtomMind

Сообщения и ошибки, регистрируемые при помощи журнала, не видны системным операторам. Только системные администраторы, имеющие доступ к машине с запущенными AtomMind Server или AtomMind Client, могут проверять журнал. Перенаправление вывода журнала другим адресатам все же не сделает информацию доступной операторам.

Чтобы сделать сообщение или проблему доступной одному и более системным операторам, сообщите о ней путем поиска события 73 AtomMind типа инфо 77:

- Извлеките экземпляр соответствующего `Context` через `ContextManager.get()`. Это может быть контекст устройства 1494 (если сообщать о проблеме из драйвера), контекст администратора 1450 (если сообщать о проблеме масштаба системы, такой как проблема запуска плагина) или любой другой контекст AtomMind Server.
- Вызовите `Context.fireEvent()` для генерирования нового события контекста. Событие будет сохраняться в базе данных 692 сервера и направляться подписанным слушателям (например, Журнал событий 398), чтобы системные операторы и администраторы могли получить доступ.

Ниже приведенный пример иллюстрирует, как составляется отчет о внутренней исправимой ошибке из драйвера устройства:

```
Context deviceContext = getDeviceContext();

deviceContext.fireEvent(AbstractContext.E_INFO, "Something happened inside the driver");
```

15.1.3 SDK плагинов

AtomMind Server и AtomMind Client поддерживают несколько видов плагинов для связи с разными Devices, добавляя новые контексты 41 в серверное дерево контекстов, создавая пользовательские запросы, локализуя и индивидуализируя текстуальные и графические ресурсы и прочее. Основная информация доступна в разделе Плагины 207, при этом данная часть документации рассказывает о разработке новых плагинов.

Архитектура плагина

Архитектура плагина основывается на библиотеке *Модульной архитектуры Java* (JPF). JPF выбран из-за архитектуры наподобие OSGi, так как занимает мало места. Это позволяет серверу оперировать в среде с небольшой памятью, наподобие одноплатных компьютеров, шлюзов IoT, сенсорных панелей и программируемых логических контроллеров Linux (PLC).

Большинство серверных модулей реализуются как плагины, поэтому являются опциональными.

Точки расширения

Все плагины, за исключением так называемого *корневого плагина,* расширяют многие другие плагины путем соединения с *точкой расширения*. Точка расширения должна быть обозначена в дескрипторе плагина 1341.

Типы плагинов

Доступно несколько типов плагинов.

Типы плагинов AtomMind Server отмечены в таблице ниже:

Тип плагина	ID точки расширения	Путь к архиву плагина
Драйвер устройства 1345	device	/plugins/device
Плагин контекста 1353	context	/plugins/context
Постоянный плагин 1358	persistence	/plugins/persistence
Плагин UI компонента 1362	component	/plugins/component
Внешний плагин аутентификации 1359	auth	/plugins/auth
Плагин сервера устройства 2092 (устаревший)	ds	/plugins/ds

Структура плагина

У плагина AtomMind Server есть два основных компонента:

- *Plugin Java class*, который реализует интерфейс `ContextPlugin`. Большинство реализаций расширяют дочерний класс класса `BasePlugin`, чтобы избежать реализации ненужных методов и сохранить функциональность по умолчанию.
- *Дескриптор плагина* 1344, который определяет свойства плагина и его место в иерархии плагинов AtomMind Server.

Эти компоненты могут быть упакованы в Java Archive (JAR). Дескриптор плагина (`plugin.xml`) должен располагаться в корневой папке архива.

ID плагина

У каждого плагина есть уникальный ID, к примеру, `com.tibbo.linkserver.plugin.context.uniqueid`. В некоторых местах такой ID может быть заменен *коротким ID*, который является последним сегментом полного ID плагина, например, `uniqueid`.

Создание нового плагина AtomMind Server

Чтобы создать с нуля новый плагин AtomMind Server, необходимо создать новый класс Java, наследуемый из дочерних классов `BasePlugin` (таких как `AbstractDeviceDriver`, `AbstractContextPlugin` или `AbstractAuthPlugin`). Вам придется переписать по крайней мере некоторые из методов, чтобы обеспечить фукциональность плагина.

Процесс создания плагина включает в себя следующие основные стадии:

- Реализация конфигурации плагина 1344
- Подготовка дескриптора плагина 1344
- Создание и внедрение архива плагина 1345

Базовые методы плагина

У каждого плагина AtomMind есть следующие методы (заявленные в интерфейсе `AggreGatePlugin`):

- `getId()`, `getShortId()`, `getDescription()` и `getSortIndex()` вызываются ядром, чтобы собрать основную информацию о плагине. Обычно не запрашивается переписывание этих методов с момента, когда классы абстрактного плагина предоставляют верные реализации.
- `globalInit()`, `globalDeinit()`, `userInit()` и `userDeinit()` должны быть переписаны, чтобы выполнить базовую инициализацию/деинициализаю плагина. Эти методы вызываются в период загрузки/выгрузки процессов включения[1393] и выключения[1394] сервера. Методы реализации должны вызывать `createGlobalConfigContext()` и `createUserConfigContext()`, чтобы установить конфигурацию[1344] плагина.
- `globalStart()` и `globalStop()` должны быть переписаны, чтобы реализовать логику включения и выключения. Эти методы вызываются в период запуска/остановки дерева контекстов, в процессе включения[1393] и выключения[1394] сервера.

Загрузчики классов плагинов

У каждого плагина есть собственный загрузчик класса (реализация *java* `ClassLoader`), отделенный от первичного загрузчика класса сервера.

Доступ к загрузчику класса отдельного плагина возможен путем передачи ID плагина в метод `getPluginClassLoader(String) PluginDirector`, который, в свою очередь, может быть получен посредством метода `getPluginDirector() ContextManager`.

15.1.3.1 Конфигурация плагина

У определенных плагинов могут быть настройки конфигурации. Настройки каждого плагина доступны в отдельном контексте[1506] в контейнере драйверов/плагинов[1505].

Есть два типа настроек конфигурации:

- **Глобальные настройки**. Каждый плагин имеет единственный набор настроек на AtomMind Server.
- **Настройки для каждого пользователя**. Для каждого пользователя системы[478] есть отдельный набор настроек.

Реализация глобальных конфигураций

Чтобы внедрить глобальные настройки в ваш плагин, перезапишите метод `globalInit(Context)`. Его реализация должна вызывать `createGlobalConfigContext(Context, boolean, VariableDefinition...)` и предоставлять метод `VariableDefinition` глобальных настроек. Эти настройки будут переданы системным администраторам, их значения будут постоянно храниться в БД.

Чтобы принять настройки значений из любого метода плагина, получите контекст глобальных конфигураций, используя `getGlobalConfigContext()`, затем вызовите `Context.getVariable()`, чтобы достать экземпляры `DataTable`, представляющие значения настроек.

Используйте метод `globalDeinit(Context)`, чтобы деактивировать плагин.

Реализация настроек для каждого пользователя

Реализация таких настроек похожа на реализацию глобальных конфигураций. Методы `userInit(Context)` и `userDeinit(Context)` должны быть реализованы для управления конфигурацией плагина. Вызовите `createUserConfigContext(Context, boolean, VariableDefinition...)`, чтобы добавить переменные настроек для каждого пользователя. Используйте `getUserConfigContext(String)`, чтобы предоставить доступ к контексту настроек для каждого пользователя с указанным именем пользователя.

15.1.3.2 Дескриптор плагина

Дескриптор плагина AtomMind Server -- это XML-файл, описывающий отдельный плагин[207] AtomMind Server и его место в иерархии плагинов сервера.

Создание дескриптора плагина

Чтобы создать дескриптор[1344] для плагина, сделайте копию файла плагина *plugin.xml* и редактируйте следующее:

- Измените последнее слово в атрибуте `id` тэга `<plugin>` для добавления ID нового плагина. ID должен содержать только строчные буквы, цифры и символы нижнего подчеркивания. Например, если вам потребовался плагин с ID `xyz`, установите атрибут ID на `com.tibbo.linkserver.plugin.context.xyz`.

- Измените атрибут `class` тэга `<plugin>` на полное имя класса плагина java.
- Введите дескриптор плагина в тело тэга `<doc-text>`.
- Измените атрибут `id` тэга `<extension>` на новый ID плагина.

Зависимости плагина

Если плагин запрашивает другие плагины для работы, укажите их как зависимости в тэге `<requires>`. См. пример ниже.

Пример

Пример дескриптора плагина:

```
<?xml version="1.0" ?>
<!DOCTYPE plugin PUBLIC "-//JPF//Java Plug-in Manifest 0.4" "http://jpf.sourceforge.net/plugin_0_4
<plugin id="com.tibbo.linkserver.plugin.context.access-control" version="0.0.1" class="com.tibbo.l
  <doc>
    <doc-text>Access Control</doc-text>
  </doc>
  <requires>
    <import plugin-id="com.tibbo.aggregate.common.plugin.extensions"/>
    <import plugin-id="com.tibbo.linkserver.plugin.context.cardholders"/>
  </requires>
  <runtime>
    <library id="tibbo" path="/" type="code">
      <export prefix="*"/>
    </library>
  </runtime>
  <extension plugin-id="com.tibbo.aggregate.common.plugin.extensions" point-id="context" id="acces
</plugin>
```

15.1.3.3 Архив плагина

Каждый плагин поставляется в форме единственного файла Java Archive (JAR).

Создание и внедрение архива плагина

- Сделайте копию демо-файла плагина **Ant** (`build.xml`)
- Измените имя проекта и название места назначения файла (архив плагина)
- Запустите `build.xml`, используя **Ant** (http://ant.apache.org/), чтобы создать архив плагина Java
- Остановите AtomMind Server
- Скопируйте файл JAR в надлежащую дочернюю папку `%AtomMind Server Installation Folder/plugins/`
- Запустите AtomMind Server

15.1.4 Набор для разработки драйверов (DDK)

AtomMind Server Driver Development Kit (DDK, набор инструментальных средств для разработки драйверов**) --** это часть часть комплекта разработчика AtomMind, который позволяет создавать драйвера устройств[518] AtomMind Servera на языке программирования Java.

Плагины драйверов устройства

Драйвер Устройства AtomMind Server -- это особый тип плагина[207] AtomMind Server. Технически драйвер включает в себя два обязательных компонента:

- *Driver Java class,* который реализует интерфейс `DeviceDriver`. Большинство реализаций расширяет класс `AbstractDeviceDriver`, чтобы избежать ипользования неподходящих методов и сохранения их функциональности по умолчанию.

- *Дескриптор плагина* 1344 *драйвера*, который определяет свойства плагина и его место в иерархии плагинов AtomMind Server.

Комплект разработчика устройства Device Server включает пример с открытыми исходными текстами для реализации Драйвера Устройства AtomMind Server, именуемого *Demo Device Driver*. Он находится в пакете `examples.driver` и содержит три файла:

- `DemoDeviceDriver.java` - исходный код для драйвера
- `plugin.xml` - дескриптор плагина драйвера
- `build.xml` - файл сборки проекта для Ant с единственным заданием собрать драйвер в виде JAR-архива.

Чтобы испытать драйвер:

- Запустите build.xml, используя Ant для сборки `demo.jar`
- Скопируйте `demo.jar` в `%AtomMind Server Installation Folder/plugins/device` , когда не запущен AtomMind Server .
- Запустите AtomMind Server
- Создайте новую учетную запись устройства Device и задайте Demo Device как тип драйвера

Создание экземпляра драйвера устройства

Разработчики драйверов устройств должны знать политику, которую AtomMind Server использует для создания драйверов, т.е. экземпляры класса Java, наследуемые из `AbstractDeviceDriver`:

- При запуске сервера создается один экземпляр класса драйвера. Этот экземпляр отвечает за инициализацию/деинициализацию плагина глобального уровня или уровня каждого пользователя. Таким образом, сервер вызывает следующие методы этого экземпляра: `globalInit()`, `globalDeinit()`, `globalStart()`, `globalStop()`. Он также вызывает несколько методов, один раз на учетную запись каждого пользователя 478 в системе: `userInit()`, `userDeinit()`.
- На каждую учетную запись устройства 497, которая использует этот тип драйвера, создается один экземпляр класса драйвера. Этот экземпляр проводит непосредственную коммуникацию с аппаратными устройствами и источниками данных и взаимодействует с определенным `DeviceContext`.
- Также экземпляры класса драйвера могут создаваться во время подключения нового устройства (на этапе спецификации свойств подключения устройства). Единственный метод, который будет вызываться из этих экземпляров - это `createConnectionPropertiesFormat()`.

15.1.4.1 Реализация драйвера

Чтобы создать новый драйвер устройства с нуля, создайте новый Java-класс, наследуемый из `AbstractDeviceDriver`. Вам потребуется заменить некоторые методы для того, чтобы наделить драйвер функциональностью.

Глобальная настройка и настройка отдельной учетной записи

У некоторых плагинов могут быть как основные настройки 521, так и настройки индивидуально для каждой учетной записи пользователя.

Чтобы добавить общие и пользовательские настройки, используйте метод `globalInit()` и `userInit()`. Их реализация должна вызывать `createGlobalConfigContext()` и `createUserConfigContext()`, а также предоставлять глобальные и пользовательские настройки `VariableDefinition`. Эти настройки будут видны системным администраторам, а их значения сохранятся в БД. Переменные должны принадлежать к группе "по умолчанию", доступной как константа `ContextUtils.GROUP_DEFAULT`.

Чтобы получить доступ к значениям настроек из любого метода плагина, выберите контексты общей и пользовательской настройки, используя `getGlobalConfigContext()` или `getUserConfigContext()`, а затем вызовите `Context.getVariable()` , чтобы забрать экземпляры `DataTable`, представляющие значения установок.

Установка контекста устройства

AtomMind Server вызывает метод драйвера `setupDeviceContext()`, когда создана новая учетная запись устройства, или же когда запускается сервер. Метод получает экземпляр интерфейса `DeviceContext` , которым является Контекст Устройства 1494, отражающий учетную запись этого устройства. Вызовите

`super.setupDeviceContext()` , чтобы сохранить его ссылку внутри `AbstractDeviceDriver`. Она будет доступна через метод `getDeviceContext()`.

Основная цель `setupDeviceContext()` заключается в добавлении специфичных для устройства параметров связи. Они добавляются в виде переменных контекста, принадлежащих группе переменных `ContextUtils.GROUP_ACCESS`. Система подсказывает оператору отредактировать эти настройки в момент создания учетной записи устройства, и в последующем доступ к ним осуществляется через действие Редактировать Свойства Устройства 1494.

Параметрыв связи следует добавить к контексту устройства, используя метод `Context.addVariableDefinition()`.

Актуальные значения, заданные пользователем, можно затем забрать из `DeviceContext` , используя `Context.getVariable()`, например из `DeviceDriver.startSynchronization()` или `DeviceDriver.connect()`.

Запуск и завершение синхронизации

У драйвера устройства есть два метода, которые вызываются в начале и конце синхронизации: `startSynchronization()` и `endSynchronization()`. Эти методы можно использовать, когда необходимо инициализировать и очистить внутренний кэш драйвера или иные структуры данных.

Обработка подключения и отключения

Метод `DeviceDriver.connect()` должен установить соединения с оборудованием. Он обычно использует параметры связи, описанные в параграфе Установка Контекста Устройства (см выше). В большинстве случаев, установка соединения состоит из нескольких этапов:

- Выборка параметров связи устройства посредством вызова `getDeviceContext.getVariable()`. Используйте `getDeviceContext().getCallerController()` в качестве контроллера вызова для обеспечения должного уровня прав доступа.
- Создание подключения (открыть сокет TCP, серийный порт и пр.)
- Отправка регистрационных данных авторизации/аутентификации и обработка ответа.
- Переключение устройства на "режим конфигурации", если это применимо.

Метод `connect()` должен сгенерировать исключение, если соединение прерывается на любом из этапов. В ином случае необходимо вызвать метод `super.connect()` в конце тела метода `AbstractDeviceDriver`, чтобы обозначить, что устройство подключено. `AbstractDeviceDriver` будет предоставлять результат для метода `isConnected()`.

`DeviceDriver.disconnect()` должен правильно закрывать соединение с оборудованием. Вызывается при остановке сервера, удалении учетной записи, или когда система запрашивает повторное соединение с устройством.

Чтобы зставить драйвер переподключиться во время тестирования/отладки драйвета, следует использовать действие Перезапустить драйвер устройства 1495.

Чтение метаданных устройства

Существует три метода, вызываемые ядром AtomMind Servera во время полной синхронизации 514:

- `readVariableDefinitions()`, который должен проверять доступные настройки устройства (или свойства связи, если протокол устройства не поддерживает самодокументирование) и создавать объект `VariableDefinition` для каждой доступной настройки устройства. В зависимости от протокола связи, настройки устройства могут также называться *каналы ввода/вывода, тэги, записи настроек* и пр. См. определение переменной 61.
- `readFunctionDefinitions()`, который должен предоставлять по одному объекту `FunctionDefinition` для каждой операции, доступной для выполнения на устройстве. Для различных видов соединений согласно различным стандартам именования такие операции могут таже называться *методами, функциями, действиями, процедурами*, и т.д. См. определение функций 70.
- `readEventDefinitions()`, который предоставляет по одному объекту `EventDefinition` для каждого **типа** события, которе может быть сгенерировано оборудованием. В некоторых протоколах устройств события могут также называться *сообщениями, уведомлениями* и пр. См. определение события 73.

Определения переменных, функций и событий, предоставленные драйвером, добавляются в контекст устройства Device и кэшируются 502. Запросы драйверу об их повторном чтении могут не поступать до момента следующей полной синхронизации, даже если сервер был перезапущен.

Чтобы принудить к повторному чтениею метаданных устройства во время тестирования/отладки драйвера, используйте действие Перезапустить драйвер устройства 1495.

Чтение/запись настроек устройства

Если драйвер предоставил непустой список настроек устройства из метода `readVariableDefinitions()`, ядро сервера вызовет метод драйвера `readVariableValue()` и `writeVariableValue()`. `readVariableValue()` должен прочитать последнее значение одной из настроек оборудования и преобразовать его в форму `DataTable`, соответствующую формату, определенному в его `VariableDefinition`. С другой стороны, `writeVariableDefinition()` принимает значение вида DataTable и конвертирует его в исходный формат, записывая переменную в устройство.

Настройки можно прочитать с оборудования:

- во время первой синхронизации и
- во время любой другой синхронизации, если на сервере их значения не изменились

Настройки записываются в оборудование во время синхронизации, если они могут быть записаны по определению и на сервере были произведены некоторые изменения их значения.

Значения настроек будут кэшироваться внутри контекста устройства. Драйвер может не запросить о повторном их прочтении до следующей полной синхронизации, даже если сервер был перезапущен.

Чтобы возбудить повторное чтение настроек устройства во время тестирования/отладки драйвера, следует использовать действие Синхронизировать 1495.

Выполнение операций устройства

Если драйвер предоставил непустой список настроек устройства из метода `readFuncitonDefinitions()`, AtomMind Server добавит действие Вызвать функцию 103 для каждой операции устройства. Выполняемое системным оператором, это действие запрашивает входные данные (согласно формату ввода функции, и только если он не "пустой"), а затем вызывает метод драйвера устройства `executeFunction()` и показывает его вывод оператору (также только если он не "пустой").

Таким образом, реализация ориентированного на драйвер метода `executeFunction()` должна:

- конвертировать входные Таблицы Данных в исходный формат устройства
- отправлять входные данные на оборудование
- переключать операции устройства
- читать вывод из устройства
- конвертировать данные вывода в Таблицу Данных и возвращать их

Работа с событиями устройства

При оперировании событиями, драйвер устройства:

- возвращает список *определений событий* из метода `readEventDefinitions()`. Список должен включать каждый тип события, который может производить аппаратное оборудование.
- асинхронно вызывает метод `DeviceContext.fireEvent()`, когда новой *экземпляр события* получен из оборудования.

Драйвер устройства может начать запускать события контекста только после окончания первой синхронизации, т.е. после того, как его метод `finishSynchronization()` вызван хотя бы один раз.

Перед тем, как запустить событие, драйвер должен удостовериться, что формат Таблицы Данных События, содержащий отосящиеся к событию данные, соответствует формату, хранящемуся в соответствующем Определении События.

Любое событие, предоставляемое драйвером устройства, постоянно. Оно сохраняется в истории событий 73, если времени хранения истории 503 не присвоено нулевое значение.

Расширенные свойства

См javadocs по `DeviceDriver`, `DeviceContext` и описание всех методов родительских интерфейсов.

РЕИНИЦИАЛИЗАТОРЫ ПАРАМЕТРОВ ДОСТУПА

Если вы уже добавили несколько настроек связей, зависящих от девайсов (см. **Создание Контекста Девайсов**), вы можете выделить действия, которые будут вызываться, если эти настройки поменяются.

Чтобы сделать это, используйте `DeviceContext.setAccessSettingReinitializer (String variable, AccessSettingReinitializer reinitializer)`. Метод реинициализатора `reinitialize()` будет вызываться каждый раз, когда предоставленная переменная изменяется. Реиниализатор может представлять действия, зависящие от девайсов. Его выход имеет булевское значение, которое показывает, требуется ли девайсу повторная синхронизация после изменения.

15.1.4.2 Работа с активами устройства

Некоторые драйверы устройства поддерживают активы[501] для обеспечения иерархического логичного группирования настроек, операций и событий.

Некоторые примеры использования активов:

- драйвер JMX-устройства возвращает под одному средству для каждого класса MBean. У каждого актива есть один подактив для каждого экземпляра MBean.
- драйвер OPC-устройства предоставляет дерево активов, которое соответствует иерархии сервера OPC.
- драйвер BACnet-устройства предоставляет актив для каждого объекта типа BACnet. У каждого актива есть один подактив для каждого экземпляра BACnet-объекта.
- Аналогичным образом, драйвер WMI-устройства создает актив для каждого типа объекта типа WMI.

Внедрение поддержки активов в драйвере

Драйверы, которые поддерживают активы, должны возвращать TRUE из метода `isUsesAssets()`.

Если определенный драйвер заявляет о поддержке активов, ядро AtomMind Server вызывает его метод `readAssetDefinitions()` сразу после установления или подтверждения соединения с устройством. Этот метод должен вернуть список объектов `DeviceAssetDefinition`. У каждого определения актива есть уникальный ID, человекочитаемое описание и список вложенных определений (подактивов).

AtomMind Server читает определения активов из драйвера лишь раз, во время этапа создания устройства, и кэширует их в БД. Чтобы заставить сервер перечитать определения активов из драйвера, следует использовать операцию Вернуть в исходное состояние драйвер устройства[1495].

Каждый актив можно активизировать/отключить системным операторам, используя действие Редактировать свойства устройства[1494].

Если драйвер поддерживает активы, AtomMind Server вызывает следующие методы, чтобы получить метаданные с устройства:

- `readVariableDefinitions(List<DeviceAssetDefinition>)`
- `readFunctionDefinitions(List<DeviceAssetDefinition>)`
- `readEventDefinitions(List<DeviceAssetDefinition>)`

Драйвер **должен возвращать только те определения, которые принадлежат активированным активам** (т.е. `DeviceAssetDefinition.isEnabled()` возвращает true). Определения настроек, операций или событий, который принадлежат отключенным активам, не будут доступны в AtomMind.

15.1.4.3 Синхронизация

Этот раздел описывает последовательность вызовов различных методов `DeviceDriver` ядром сервера во время синхронизации[514] устройства с сервером.

1. Если устройство приостановлено[511], синхронизация пропускается.
2. Если выражение зависимости[512] устройства оценивается как ложное, синхронизация пропускается, а соединение с устройством переключается в режим **Offline**.
3. Должен быть вызван метод драйвера `shouldSynchronize()`. Если он возвращает false (например, если настройки для связи с устройством еще не определены), синхронизация пропускается.
4. Если активирован расширенный статус[511], статус синхронизации[516] переключается на **Соединение**.
5. Если метод драйвера `isUsesConnections()` возвращает true:

5.1. Если `isConnected()` возвращает false, или ядро системы запрашивает повторное соединение с оборудованием (например, потому, что предыдущая синхронизация не удалась, и включена настройка Прервать синхронизацию при ошибке 510), система вызывает `disconnect()` (только если `isConnected()` верно) и затем `connect()`.

5.2. Если `connect()` не генерирует исключительную ситуацию, устройство переключается в статус соединения в режиме **Online**.

6. Определенные задачи синхронизации, именуемые "connect-only" могут завершится на этом этапе.

7. Если активизирован расширенный статус 511, статус синхронизации 516 устройства переводится в **Чтение Метаданных**.

8. Вызвается `startSynchronization()`.

9. Если параметны синхронизации требуют чтения метаданных (т.е. выполняется "полная" синхронизация):

9.1. Если драйвер поддерживает группы (т.е. `isUsesGroups()` возвращает true), а группы еще не прочитаны из устройства (или же драйвер был перезапущен), вызывается метод `readGroupDefinitions()`.

9.2. Вызывается `readVariableDefinitions()`, новые и измеренные определения добавляются в контекст устройства и кэшируются в БД. Если драйвер поддерживает группы, вместо него вызывается метод `readVariableDefinitions(List<GroupDefinition>)`.

9.3. Если у настроек синхронизации 502 некоторых только что добавленных определений переменных есть период общей синхронизации, назначаются задачи синхронизации для каждой из настроек.

9.4. Вызвается `readFunctionDefinitions()`, новые измененные определения добавляются в контекст устройства и кэшируются. Если драйвер поддерживает группы, вместо него вызывается метод `readFunctionDefinitions(List<GroupDefinition>)`.

9.5. Соответствующее действие Вызвать функцию 103 добавляется в контекст устройства для каждого нового определения функции.

9.6. Вызывается `readEventDefinitions()`, новые и измененные определения добавляются в контекст устройства и кэшируются. Если драйвер поддерживает группы, вместо него вызывается метод `readEventDefinitions(List<GroupDefinition>)`.

9.7. Если любой из выше упомянутых методов генерирует `DisconnectionException`, в журнал регистрации событий записывается ошибка, устройство переключается в статус соединения **Offline**, вызвается метод драйвера `disconnect()`, а синхронизация завершается.

10. Если активизирован расширенный статус 511, статус синхронизации 516 устройства перводится в **Синхронизация Настроек**.

11. Все настройки устройства (или настройки, заданные в параметрах синхронизации) синхронизируются с сервером. Алгоритм синхронизации завиит от режима синхронизации 503 настроек. На этом этапе Сервер может вызвать следующие методы драйвера:

11.1. Метод `getVariableModificationTime()`, чтобы получить отметку времени устройства для настроки. Если отметки времени модификации не поддерживаются оборудованием, этот метод возвратит нулевое значение. В этом случае сервер запишет значение настройки в устройство, если оно не было изменено в кэше сервера, или же прочитает его из устройства в противном случае.

11.2. Метод `readVariableValue()`, чтобы прочесть значение настройки с устройства.

11.3. Метод `writeVariableValue()`, чтобы записать значение с сервера в устройство.

11.4. Метод `updateVariableModificationTime()`, чтобы сохранить новые метки времени модификации в устройстве.

11.5. Если любой из выше упомянутых методов генерирует `DisconnectionException`, это регистрируется в журнале событий, устройство переходит в режим `Offline`, вызвыается метод драйвера `disconnect()`, синхронизация завершается.

11.6. Если любой из выше перечисленных методов генерирует исключительную ситуацию, это регистрируется в журнале событий, статус синхронизации настроек 506 устанавливается в **Error**, а синхронизация завершается, если активирована настройка Прервать синхронизацию при ошибке 510.

12. Вызвается метод `finishSynchronization()`.

13. Вычисляются статусы синхронизации и режима устройства в сети.

15.1.4.4 Управление входящими соединениями устройств

AtomMind Server обычно устанавливает исходящие соединения TCP/UDP с подключенными к сети устройствами, аутентифицируя себя с учетными данными, находящимися в учетной записи устройства. Однако некоторые устройства сами подключаются к серверу. Это в основном M2M устройства, подключаемые к IP сети через модемы GPRS/3G. Такие устройства не имеют статичных IP адресов и имен хоста, поэтому доступ сервера к ним невозможен.

Драйвера устройств, обслуживающие входящие соединения с устройствами, требуют специального подхода к их разработке:

1. Метод драйвера устройства `globalInit()` должен вызывать метод `createGlobalConfigContext(Context rootContext, VariableDefinition... properties)`. Его параметры должны включать определение переменной, которая содержит настройки глобального подключения драйвера, т.е. хотя бы номер порта для прослушивания входящих соединений с устройствами.

2. Метод драйвера `globalStart()` должен открывать прослушивающий сокет и запускать отдельный поток, который принимает и обслуживает входящие соединения с устройствами. Этот метод имеет доступ к свойствам глобальной конфигурации драйвера путем получения контекста глобальных свойств через `getGlobalConfigContext()` и извлечения значения переменной глобальной конфигурации из этого контекста. Возможно (но не необходимо) создать класс, унаследованный из `SocketServerThread` (для нормальных сокетов) или `SslSocketServerThread` (для сокетов, защищенных SSL), и использовать этот класс в качестве потока прослушивания сокетов.

3. Вышеупомянутый поток сокетов сервера должен прерваться из метода драйвера `globalStop()`.

4. Поток сокетов сервера должен принимать входящие соединения и создавать новый поток (поток контроллера соединений) для обслуживания каждого соединения. Например, неабстрактная реализация `SocketServerThread` или `SslSocketServerThread` должна включать метод `createConnectionThread()`, который создает поток контроллера входящих соединений каждый раз при подключении нового устройства к серверу.

5. Каждая учетная запись устройства должна в большинстве случаев определять уникальный идентификатор устройства (такой как код мобильного устройства IMEI) и пароль устройства, используемый для аутентификации и авторизации входящих соединений с устройствами. Переменная, содержащая эту информацию, должна добавляться к учетной записи устройства путем реализации метода `setupDeviceContext()`.

6. Поток контроллера соединений должен сначала аутентифицировать и авторизовать устройство, используя родной протокол устройства для извлечения его уникального идентификатора, и найдя учетную запись устройства, соответствующую этому идентификатору, а также проверив, что пароль устройства соответствует паролю, определенному в настройках учетной записи устройства.

7. Как только устройство аутентифицируется и определяется соответствующая учетная запись (контекст) устройства сервера, поток контроллера устройства должен:

   7.1. Получить экземпляр класса драйвера/плагина на устройство путем вызова `deviceContext.getDriver()`.

   7.2. Зарегистрировать данный контроллер устройства в экземпляре устройства для установления временных отношений между учетной записью/драйвером устройства и текущим входящим соединением, управляемым контроллером.

   7.3. Вызвать `deviceContext.requestSynchronization()` для получения нормальной последовательности синхронизации 1349.

8. Все "классические" методы драйвера должны выполнять ввод/вывод устройства через экземпляр контроллера устройства, зарегистрированного потоком контроллера входящих соединений. Однако входящие соединения вносят в этот процесс некоторые особенности:

   8.1. Метод драйвера `shouldSynchronize()` должен вернуть true, если контроллер входящих соединений в данный момент зарегистрирован в экземпляре драйвера, в ином случае false.

   8.2. Метод драйвера `connect()` должен отправлять устройству некоторые тестовые команды, чтобы убедиться, что входящее соединение все еще имеется. Если соединение TCP/UDP прервано или устройство не отвечает, метод соединения должен вызвать исключение для приостановки последующей синхронизации.

   8.3. Метод драйвера `disconnect()` должен отключать контроллер входящих соединений и закрывать его входящие соединения.

15.1.4.5 Обработка асинхронных обновлений переменных

Некоторые коммуникационные протоколы позволяют устройствам асинхронно информировать сервер об обновлениях значений переменных. Это часто происходит в двух случаях:

- Если коммуникационный протокол оптимизирован с целью избежания опроса через низкоскоростыне, ненадежные соединения с плохой пропускной способностью
- Если устройство работает с операционной системой реального времени и коммуникационный протокол позволяет ему отправлять обновления значений с определенной скоростью

Если драйвер был разработан с целью получения асинхронных обновлений значений, он должен выполнить следующие действия при получении такого обновления:

- Создать новую `DataTable`, представляющую значение переменной, и заполнить ее данными
- Вызвать `DeviceContext.asyncVariableUpdate(String variable, DataTable value)`

AtomMind Server обработает новое значение, сохранит его в истории и доставит с целью обновления слушателей, как только будет вызвана `asyncVariableUpdate()`.

15.1.4.6 Стандартный набор инструментов драйвера

Набор для разработки драйверов AtomMind Server также включает в себя так называемый *стандартный набор инструментов драйвера,* который является набором классов, упрощающим создания типичных драйверов для сетевых устройств, программируемых логических контроллеров (PLCs), шлюзов и сенсоров интернета вещей (IoT), спутниковых трекеров и т.д.

Стандартный набор инструментов - это технически полная, простая разработка драйверос со следующими характеристиками:

- Связь по TCP, UDP или по последовательному соединению
- Разбор входящих данных потока в раздельные команды
- Поддержка оповещений (асинхронные входящие команды) и ответов (синхронные ответы на ранее отправленные команды)
- Дополнительный разбор и обработка входящих команд по коду пользователя
- Кодировка и отправка исходящих команд
- Соотнесение входящих команд с ранее отправленными исходящими посредством пользовательских идетификаторов
- Поддержка отправления множественных исходящих команд одновременно и получения ответов вне очереди

Разработка драйверов с помощью стандартного набора инструментов

Чтобы разработать драйвер устройств AtomMind Server, который основывается на стандартном наборе инструментов и использует полностью дополнительную настройку соединения и код соединений устройства, следует:

- Создать новый общий драйвер устройства (плагин класс + драйвер класс), как описано в других статьях этого раздела
- Создать новый класс команды устройства, наследуемый от `Command`, а также, по желанию, больше дочерних классов для различных типов входящих и исходящих команд. Классы команды главным образом содержат обе базы устройств: первичную и обработанную
- Создать новый класс контроллера устройства, наследуемый от `AbstractDeviceController` и установить его параметры по базе входящих/исодящих классов команды. Контроллер устройства отвечает за управление связями устройства и за данные
- Создать новый командовый анализаторный класс путем реализации интерфейса `CommandParser` или (лучше) путем наследования у `AbstractCommandParser`
- Для второго подхода (при использовании `AbstractCommandParser)` система будет добавлять входящие данные для вашего буфера анализатора байт за байтом, затем вызовет метод `readCommand()`. Если вы увидите, что буфер содержит полностью сформированную команду, просто создайте объект `Command` и верните его, в другом случае верните null. Собранные на данном этапе данные хранятся в `ByteArrayOutputStream` и доступны посредством `byte[] getData()`
- Переопределить методы `connectImpl()`, `loginImpl()`, и `disconnectImpl()` вашего контроллера устройства и установить связь устройств, имя пользователя и логику разрыва соединения в них

- Убедиться, что вы вызываете метод `setCommandParser()` из вашей настройки `connectImpl()`, чтобы предоставить правильный анализатор команды для контроллера устройства
- Переопределить метод `send(Command cmd)` вашего контроллера устройства и реализовать команду, отправляя в нее логику, например отправьте данные команды в сокет, созданный в течение соединения
- Переопределить методы драйверов устройств `setupDeviceContext(DeviceContext deviceContext)` и добавить свойства контроллера устройства в группу переменных контекста устройства `ontextUtils.GROUP_ACCESS`
- Переопределить метод `connect()` вашего класса драйвера устройства, чтобы создать настроенный вами контроллер устройства и вызвать его метод `connect()`. Реализовать метод драйвера `disconnect()` путем переноса логики в вызов `disconnect()` контроллера устройства
- Внести необходимую логику связи устройства в методы драйвера `readVariableDefinitions()`, `readFunctionDefinitions()`, `readEventDefinitions()`, `readVariableValue()`, `writeVariableValue()` и `executeFunction()`. Эти методы обычно взаимодействуют с контроллером устройства путем отправления различных команд, получения ответов и нормализации значений протокола устройста, например, конвертируя их в/из `DataTable` экземпляров
- Если ваше устройство отправляет асинхронные команды (события), убедитесь в реализации метода `processAsyncCommand()` у `AbstractDeviceController`. Его реализация обычно должна создавать любое событие в контексте устройства. Убедитесь, что тип события был ранее представлен в ядре, полученном от драйвера `readEventDefinitions()`

Разработка драйверов с помощью набора инструмента и связи устройств

Этот раздел описывает, каким образом разработать драйверы устройств, которые:

- Используют TCP (протокол управления передачей), UDP (протокол пользовательских датаграмм) или серийные связи и единый подход конфигурации устройства
- Используют бинарные или текстовые команды устройства, разделенные одним или двумя пользовательскими символами characters ("разделители")

В таком случае вам необходимо следовать инструкции, написанной в разделе **Разработка драйверов с помощью стандартного набора инструментов**. Однако разработка такого драйвера даже проще. Список различий представлен ниже:

- Создать объект связянных свойств путем разработки интерфейс `ConnectionProperties` или увеличения класса `DefaultConnectionProperties`
- Используя наследование от `DefaultConnectionProperties`, клонировать его свойства `FORMAT` в статическом блоке, сделать необходимые модификации, добавить поля и передать новый формат конструктору `DefaultConnectionProperties`
- Использовать ваш контроллер устройства с помощью `StandardDeviceController` вместо `AbstractDeviceController`. Вам не понадобится разрабатывать логику установки/разрыва соединения, а также логику, которую отправляет команда. Просто убедитесь, что вы перезаписываете `connectImpl()`, чтобы установить анализатор команды с помощью `setCommandParser()` после вызова `super.connectImpl()`.

Резюмируя вышесказанное, важно отметить, что в этом случае вы не разбираете проблемы, касающиеся связи устройства, а имеете возможность полностью концентрироваться на обмене данными устройства.

Количество кодов, которые могут быть разработаны для создания нового драйвера устройств, относительно этого случая действительно минимально.

15.1.5 Плагины контекста

У AtomMind есть большой набор встроенных инструментов обработки данных, например: тревоги 779, фильтры событий 762, виджеты 943 или трекеры 2181. Плагин SDK AtomMind разработан для создания пользовательских экземпляров существующих инструментов (например, оперативное создание виджетов на базе текущего состояния системы) и создания полностью новых инструментов (например, политики доступа для сферы контроля физического доступа).

Эти инструменты добавлены в ядро AtomMind Server в форме плагинов 1342 контекста.

Используйте скрипты 879 вместо новых плагинов AtomMind Server, чтобы решить простые задачи обработки данных.

Итак, плагин AtomMind Server может:

- Добавлять новые контексты 41 в дерево контекстов AtomMind Server.

- Добавлять новые переменные 61, функции 70, события 73 и действия 87 к существующим контекстам. Функциональность таких объектов может быть реализована посредством пользовательского кода.
- Запустить любой асинхронный код (слушая сокеты сервера, потоки и т.д), который взаимодействует с существующими или пользовательскими контекстами путем изменения значения их переменных, выполнения их функций или создания событий в них.

Примеры

Далее следуют несколько примеров плагинов AtomMind Server:

- Плагин **Управление активами**, который создает множество общих таблиц 2176, содержащих информацию, касающуюся конкретных активов, а также соединяет их с устройствами
- Плагин **Сервер Syslog**, который получает сообщения от серверов через сеть и конвертирует их в события AtomMind для сохранения и дальнейшей обработки
- Плагин **промышленного ввода/вывода**, который создает виджеты для контроля статуса программных контроллеров

SDK комплект Device Server включает в себя пример реализации плагина AtomMind Server с открытым исходным кодом, который называется *Demo Plugin*. Он находится в пакете `examples.plugin` и содержит 3 файла:

- `DemoServerPlugin.java` - исходный код плагина
- `plugin.xml` - дескриптор плагина
- `build.xml` - файл **Ant** с единственной задачей построения плагина JAR

Чтобы испытать плагин, необходимо:

- Запустить *build.xml*, используя Ant, чтобы создать `demo.jar`
- Скопировать `demo.jar` в `%AtomMind Server Installation Folder/plugins/context`, когда AtomMind Server не запущен
- Запустить AtomMind Server

15.1.5.1 Подключение к серверу дерева контекста

Интерфейс `ContextPlugin` предоставляет несколько групп методов, чтобы подключить новые элементы к дереву контекстов 41 AtomMind Server. У каждой группы есть два метода: метод "загрузка", вызываемый во время загрузки сервера или создания контекста, и другой метод "выключение", вызываемый во время выключения сервера или удаления контекста.

Инициализация/деинициализация плагина

Когда создается новый плагин контекста, сервер вызывает его метод `initialize()`, который может содержать общий код инициализации.

Когда плагин удаляется, сервер вызывает его метод `deinitialize()`.

Настройка на базе контекста

Каждый раз, когда новый контекст создается впервые или повторно при загрузке сервера, сервер вызывает метод плагина `install(Context context)`. Его реализация может:

- Добавлять определения переменных/функций/событий/действий в контекст и при необходимости предоставлять их "пользовательский код" (переменные getters/setters, реализации функций)
- Добавлять слушателей события в контекст

Пожалуйста, избегайте предоставления разрешения доступа другим контекстам сервера из метода `install(Context context)`. С момента создания сервера дерева контекстов определенные контексты сервера могут быть доступны во время отладки и становиться недоступными в рабочей среде, что может стать причиной непредсказуемого поведения.

Чтобы получить доступ ко всему дереву контекстов, добавьте свой код в методы `install(ContextManager contextManager)` или `launch()`.

Настройка на базе контекстного дерева

Метод `install(ContextManager contextManager)` вызывается сразу же после завершения загрузки серверного контекстного дерева и после того, как все контексты загрузятся. Его реализация может получать определенные серверные контексты, используя метод `ContextManager.get()`, а также добавлять определения переменных/функций/событий/действий и/или слушателей событий.

Запуск плагина

Метод плагина `launch()` вызывается в самом конце загрузки сервера, когда все серверные контексты и компоненты включены и запущены. Он должен быть реализован, только если другие коды из любых описанных выше методов не сработают.

15.1.5.2 Реализация новых контекстов

Многие плагины контекстов добавляют новые типы контекстов в дерево контекстов AtomMind Server. В большинстве случаев эти дополнительные контексты придерживаются типичных двухуровневых парадигм:

- Контекст контейнера, который находится в контексте пользователя 1608, которому он принадлежит
- Системные администраторы, инженеры и пользователи создают экземпляры дополнительных контекстов в этом контейнере

Например, контекст контейнера Тревоги 1452 и множественные контексты Тревога 1454, определенные пользователем, следуют этой парадигме.

Многие классы, относящиеся к контекстам сервера, скорее являются частью AtomMind Server, нежели частью Java SDK с открытым исходным кодом. Эти классы недоступны в исходном коде. Чтобы получить доступ к этим классам, нужно добавить следующие файлы JAR к пути класса:

- *aggregate-commons.jar*
- *server-core.jar*

Эти файлы JAR можно найти в подпапке /jar установочной папки AtomMind Server.

Добавление нового контейнера для контекстов, определенных пользователями, включает в себя набор шагов:

1. Реализация класса контекста контейнера (контекст, который содержит в себе объекты бизнес-логики)
2. Реализация класса контекста для пользовательского ресурса (например, контекст объекта бизнес-логики)
3. Регистрация Создателя контейнера ресурса
4. Создание контекстов агрегации

Реализация класса контекста контейнера

Контексты контейнера (такие как контекст Тревога 1452 или Виджет 1624) наследуются от `EditableChildrenContext`. Чтобы реализовать контейнер для ваших пользовательских ресурсов, необходимо унаследовать класс из `EditableChildrenContext` путем вызова конструктора: `public EditableChildrenContext(String name, String objectName, Class<? extends EditableChildContext> childClass, TableFormat childInfoFormat)`

Ниже представлено значение параметров конструктора:

Параметр	Описание
name	Имя контекста контейнера.
objectName	Доступное для чтения описание дополнительного ресурса, например, "Тревога".
childClass	Класс контекста пользовательского ресурса. Может быть унаследован от `EditableChildContext`.
childInfoFormat	Формат 50 переменной, которая представляет основную настройку пользовательского ресурса. Пользователи должны будут вводить эти данные на протяжении процесса создания ресурса. Эта переменная должна содержать поля строк `name` и `description`. Использование статических экземпляров `TableFormat` всегда является предпочтительным поведением.

	Такой же формат должен быть использован для переменной `childInfo` контекстов пользовательских ресурсов.

Есть два метода, которые следует всегда переписывать в пользовательской реализации `EditableChildrenContext`:

- Метод `setupMyself()`. Реализация этого метода должна настроить контекст самостоятельно, также добавить дополнительные переменные, функции, события и действия.
- Метод `buildChild(String cname, boolean readOnly, String type)`. Этот метод должен возвращать экземпляр контекста пользовательского контекста, который должен быть унаследован от `EditableChildContext`. Имя возвращенного контекста должно соответствовать входному параметру `cname`.

Количество методов `EditableChildrenContext` обычно вытекает из реализаций `setupMyself()`:

- `addCreateAction()` для активации создания пользователем дочернего ресурса
- `allowGrouping()` для поддержки группирования 751 пользовательских ресурсов

Реализация класса контекста пользовательского ресурса

Контексты, соотносящиеся с вашими пользовательскими ресурсами (так, например, контекст соответствует Тревоге 1454 или Виджету 1627), должны быть унаследованы от `EditableChildContext`. Их создание происходит с помощью метода `buildChild()` от `EditableChildrenContext`, описанного выше.

Общедоступный конструктор `EditableChildContext` - это `public EditableChildContext(String name, boolean allowMakeCopy)`. Он принимает имя контекста (которое по факту относится к `buildChild()` как аргумент) и флажок, определяющий, будет ли действие Создать копию 109 включено для контекста пользовательского ресурса.

Реализации `EditableChildContext` должны перезаписать метод `setupMyself()` для персонализации самого контекста, а также для добавления дополнительных переменных, функций, событий и действий.

Код загрузки контекста обычно ссылается на следующие методы `EditableChildContext`:

- Вызов `super.setupMyself()` добавляет все необходимые основные переменные и действия.
- Выполнение `addChildInfoVariable(TableFormat rf, boolean readOnly)` добавляет переменную `childInfo` к контексту пользовательского ресурса. Его формат должен соответствовать `childInfoFormat`, передающемуся конструктору `EditableChildrenContext`, и включать в себя поля `name` и `description`.
- `addConfigureAction()`, `addDeleteAction()` и `addReplicateAction()` добавляют действия Конфигурировать 105, Удалить 106 и Реплицировать 110 соответственно.

Регистрация редактора контейнера ресурсов

Идея `ResourceContainerBuilder` заключается в необходимости добавления контейнеров контексту Пользователь 1608 и удалении их в подходящие моменты. Сам редактор обычно должен наследоваться от `DefaultResourceContainerBuilder` и быть зарегистрированным методом `ContextPlugin` от `install(ServerContext context)`.

Далее следует код в качестве примера добавления редактора контейнера ресурсов для тревог.

```
if (context instanceof UserContext)
{
  UserContext uc = (UserContext) context;

  uc.registerContainerBuilder(new DefaultResourceContainerBuilder(ContextUtils.scriptsContextPath(
  {
    @Override
    public void buildImpl(UserContext context) throws ContextException
    {
      context.addChild(new AlertsContext(context));
      context.addVisibleChild(ContextUtils.alertsContextPath(context.getName()));
    }

    @Override
    public void dismantleImpl(UserContext context) throws ContextException
    {
      context.removeVisibleChild(Contexts.CTX_ALERTS, false);
      context.removeChild(Contexts.CTX_ALERTS);
    }
```

```
    });
}
```

Метод `buildImpl()` добавляет контекст контейнера *UserContext* и регистрирует его как видимый дочерний. Метод `dismantleImpl()` возвращает эти операции.

Создание контекстов агрегации

Добавления самого контейнера в контекст пользователя недостаточно, потому что отображаемый для любого администратора AtomMind Server корень видимого дерева демонстрирует видимые дочерние части контекста корня сервера.

Поэтому контекст корня сервера также должен демонстрировать контексты, содержащие контекст пользовательских ресурсов в качестве дочерних. Эти контексты называются *контекстами агрегации*.

Больше информации о преобразованиях этого контекста доступно в статье Видимое и действительное дерево контекстов 45.

Итак, необходимо включить группирование 751 для пользовательских ресурсов.

Создание контекстов агрегации и контекстов групповых контейнеров представлено посредством одного статического вызова `ServerContextUtils.createAggregationContexts(Context context, String containerName, String containerDescription, String iconId, String helpId, String groupContainerDescription, boolean mapGroups, String... additionalChildren)`.

Ниже представлен пример того, как создаются контексты агрегации для тревог:

```
ServerContextUtils.createAggregationContexts(contextManager.getRoot(), Contexts.CTX_ALERTS, "Alert
```

15.1.5.3 Добавление ресурсов к плагинам

В большинстве случаев плагины AtomMind Server предоставляют различные *ресурсы*, такие как преднастроенные тревоги, отчеты, виджеты, инструментальные панели, модели, задачи планировщика, и т.д. Когда установлен этот плагин, предоставленные им ресурсы могут управляться, как любые другие ресурсы 208, включенные в AtomMind или коробочные решения на ее основе.

Используйте модуль Пакеты ресурсов 1318, чтобы подготовить новый плагин ресурсов без какой-либо разработки на языке Java.

Создание плагина с ресурсами

Включать ресурсы в плагин несложно:

- Разработайте и протестируйте ресурс (такой как инструментальная панель 912), используя любые визуальные инструменты, предоставляемые AtomMind
- Откройте конфигурацию ресурса в Редакторе свойств 377, кликните по кнопке *Экспортировать* на панели инструментов () и сохраните конфигурацию ресурса в файл `*.prs`.
- В исходном коде вашего плагина создайте новый пустой пакет программ ниже уровня главного пакета класса плагина. Назовите его, например, `resources`.
- Создайте пустой класс в этом пакете, напр. `ResourceAccessor`. Этот класс будет использоваться для доступа к файлу ресурса из исходного кода.
- Поместите ваш(и) файл(ы) ресурсов `*.prs` в папку уровнем выше.
- Перезапишите метод `install(ContextManager cm)` главного класса вашего плагина. В методе перезаписи создайте экземпляр класса `PropertiesResourceBuilder` для каждого ресурса и зарегистрируйте их в хранилище ресурсов, вызвав `RepositoryManager.add(ResourceBuilder builder)`.

Классы `PropertiesResourceBuilder` и `RepositoryManager` определены в архиве `server-core.jar`, доступном в любой инсталляции AtomMind Server.

Как только создан и зарегистрирован редакторы ресурсы, отредактируйте ваш плагин и перезапустите сервер. После этого вы должны увидеть новые ресурсы либо в таблице **Create/Update Resources**, либо в таблице **Delete Resources** (если ресурс с таким именем уже существует).

Различные конструкторы класса `PropertiesResourceBuilder` принимают некоторые или все следующие аргументы:

- **version** - версия ресурса, целое число, начиная с `1`, которое должно быть увеличено каждый раз, когда изменяется шаблон ресурса.
- **contextPattern** - путь контекста контейнера, который будет содержать результирующий контекст ресурса. Так как в большинстве случаев ресурс может быть создан для различных пользователей[478], шаблон отличается этим от стандартного пути контекста[42], он может содержать символ `%`, который будет заменен именем пользователя, для которого создается/обновляется/удаляется данный ресурс. Можно легко создать путь с помощью вызова статичных методов класса `ContextUtils`, например, `ContextUtils.dashboardsContextPath(ContextUtils.USERNAME_PATTERN)`.
- **name** - имя контекста источника, который будет создан, например, `networkDeviceOverview`. Не советуем включать тип ресурса в его описание, например `networkDeviceOverviewDashboard` не рекомендуется в качестве имени.

Имя ресурса должно соответствовать имени файла-шаблона `*.prs` (без расширения).

- **typeDescription** - удобное для чтения описание типа ресурса, например `Dashboard`.
- **category** - удобное для чтения описание категории, к которой будет принадлежать ресурс, например `Network Management`.
- **description** - удобное для чтения описание самого ресурса, например `Network Device Overview`. Не советуем включать тип ресурса в его описание, например `Network Device Overview Dashboard` не рекомендуется в качестве описания.
- **autoCreate** - флажок, который определяет, должен ли быть создан ресурс при первом запуске сервера в стандартной инсталляции.
- **resourceClass** - класс на один уровень выше, расположенный в том же пакете с файлом ресурса `*.prs`.
- **group** - группа[751] ресурсов, куда нужно поместить ресурс.

15.1.6 Плагины постоянного хранения данных

Плагины постоянного хранения данных реализуют хранение[692] серверных конфигураций, событий и бинарных данных. В большинстве случаев плагины постоянного хранения данных решают даже самые сложные задачи хранения данных. Однако возможно реализовать дополнительный плагин хранения данных, который будет использовать любой особый тип технологии хранения данных в БД.

Плагины постоянного хранения данных следуют шаблонам разработки Технологии доступа к данным (DAO).

Плагин постоянного хранения данных реализует интерфейс `PersistencePlugin`, который имеет единственный метод `createDaoFactory(DaoFactory parent, ServerRuntimeConfig runtimeConfig, ServerConfig config)`. Этот метод должен возвращать экземпляр реализуемого классом интерфейса `DaoFactory`.

У интерфейса `DaoFactory` есть набор методов, которые возвращают особые создания технологии доступа к данным DAO (средства хранения), поддерживаемые плагином. Самые важные реализации DAO:

- **DAO свойств**, реализованая интерфейсом `PropertyDao`. DAO отвечает за постоянное хранение значений переменных[61] сервера.
- **DAO событий**, реализованная интерфейсом `EventDao`. DAO поддерживает хранение, загрузку и управление постоянными событиями[73] сервера.
- **DAO данных**, реализованная интерфейсом `DataDao`. DAO поддерживает хранение и загрузку больших бинарных блоков данных.

Методы `DaoFactory`'s `start()` и `stop()` вызываются во время стадий загрузки[1393] и отключения[1394] сервера соотвественно.

DAO свойств

У реализации DAO свойств есть методы для хранения и загрузки значений переменных сервера, также есть перераспределение этих значений с разными путями контекста и именами переменных. Это перераспределение будет происходить, если контекст переменной был изменен или перемещен.

См. *Javadocs* интерфейса `PropertyDao` для получения дополнительной информации о каждом методе.

Как правило, таблица данных *DataTable* должна в первую очередь связываться со строкой (посредством метода `encode()`), а затем с массивом байт, который будет храниться постоянно.

Некоторые значения переменных содержат большие вложенные бинарные блоки данных (экземпляры класса `Data`). Большинство реализаций должны быть извлечены как блоки `Data` из `DataTable`, представляющей значения переменных, а также сохранять их отдельно с помощью реализации DAO данных, доступной через родительский *DaoFactory*.

Реализации DAO свойств должны также удалять блоки данных из переменной постоянного хранения, когда ее постоянное значение удаляется.

DAO событий

DAO событий отвечает за хранение, загрузку и управление событиями сервера, представленных экземплярами класса `Event`. Как правило, каждый `Event` должен быть конвертирован в `PersistentEvent`, чтобы он, в свою очередь, был конвертирован в массив байт, подходящий для постоянного хранения.

См. *Javadocs* интерфейса `EventDao` для получения дополнительной информации о каждом методе DAO событий.

ХРАНЕНИЯ СОБЫТИЙ

DAO событий во многом базируется на концепции *Хранений событий*. Хранение событий - это отдельный раздел средств хранения, предназначенный для хранения определенного набора событий, таких как таблица в реляционной БД или семейство колонок в БД NoSQL.

Хранения событий представлены классами, реализующими интерфейс `EventStorage`. Для получения дополнительной информации о хранениях событий и их параметрах см. раздел Хранения событий 200.

DAO событий должен возвращать реализацию интерфейса `EventStorageManager`, ответственного за управлениями хранениями событий. Несмотря на это, у интерфейса есть множество методов, его реализации должны быть унаследованы от `AbstractEventStorageManager`. Ниже представлены только два абстрактных метода для реализации:

`void createStorage(EventStorage storage, boolean create)`	Этот метод должен создавать новое пространство для хранения в хранилище, например, таблица базы данных или семейство колонок в NoSQL.
`void dropStorage(EventStorage storage)`	Этот метод должен удалять новое место хранения из хранилища. Например, он может отключить таблицу базы данных.

Обратите внимание, что абстрактная реализация `EventStorageManager` использует DAO свойств, чтобы сохранять информацию о доступных хранениях событий.

DAO данных

DAO данных отвечает за сохранение, загрузку и удаление больших бинарных блоков данных, представленных экземплярами класса `Data`. Каждый `Data` может быть закодирован как массив байтов (`byte[]`), которые должны храниться постоянно.

Каждый постоянно хранящийся блок данных подписан уникальным ID `long`, содержащимся в экземпляре `Data`.

См. *Javadocs* интерфейса `DataDao` для получения дополнительной информации о каждом методе.

15.1.7 Плагины внешней аутентификации

Внешний плагин аутентификации - это тип плагина 1342 AtomMind Server, который позволяет пользователям AtomMind Server аутентифицировать самих себя через внешние системы, такие как Active Directory или сервер LDAP.

Главный класс внешнего плагина аутентификации должен реализовать интерфейс `AuthenticationPlugin`. Однако большинство реализаций плагина унаследованы от `AbstractAuthPlugin`.

Процесс внешней аутентификации

Интерфейс `AuthenticationPlugin` включает в себя единственный метод:

`AuthenticationResult authenticate(CallerController caller, String username, String password, String code, String state, String provider)`.

Этот метод принимает имя пользователя и пароль, предоставляемые человеком или компонентом, которые пытаются аутентифицироваться. В случае удачной аутентификации метод должен вернуть имя пользователя 478 AtomMind Server. Эти права доступа будут использоваться, чтобы сделать доступной аутентификацию человека или компонента.

Полученное в результате имя пользователя должно быть упаковано в объект `AuthenticationResult` через вызов его `AuthenticationResult(String username)` конструктора.

15.1.8 SDK агента

Agent SDK - это Java-библиотека, которая позволяет реализовывать Agent в форме Java-приложения, запущенного на ПК или мобильном устройстве. Это приложение воспроизводит Agent , который базируется на контроллере аппаратного оборудования путем соединения с AtomMind Server через безопасное соединение, и предоставляет ряд настроек (переменных), операций (функций) и событий.

В отличие от Java-приложения, которое использует AtomMind Server API 1340, описанное приложение, используя SDK Агента, подключается к AtomMind Server в "режиме устройства". Оно эмулирует или реализует устройство, подключенное к AtomMind.

Agent может быть также реализован при помощи API для .NET 1397.

Основанные на Java Агенты идеально подходят для подключения размещенного на ПК оборудования к AtomMind с целью мониторинга и управления. Некоторые примеры устройств, на которых может быть запущен Агент для ПК:

- торговые автоматы
- киоски самообслуживания
- банкоматы
- станции дистанционного сбора данных
- и пр.

Пакет SDK сервера устройств включает пример с открытыми исходными текстами Агента на Java. Это запускаемый Java-класс `DemoAgent`, который находится в пакете `examples.agent`.

Чтобы протестировать демо-агент, измените адрес/порт/логин/пароль сервера в конструкторе класса `RemoteLinkServer` и запустите класс `DemoAgent`. Он должен подключиться к серверу и автоматически зарегистрировать новую учетную запись устройства 497.

Стуктура Агента, основанного на Java

Технически Agent на Java -- это Java-приложение или часть такого приложения, которая:

- создает экземпляр класса `Agent`
- возвращает реализацию `Context` Агента, используя метод `Agent.getContext()`
- настраивает контекст, добавляя определения переменных, функций и событий. Это настройки, операции и события, "предоставленные устройством аппаратного оборудования" в рамках AtomMind.
- предоставляет собственные `VariableGetter` и `VariableSetter` для каждой настройки Агента. Эти классы реализуют общую логику для чтения и записи значений настроек.
- предоставляет собственный `FunctionImplementation` для каждой операции Агента. Эти классы реализуют логику операций, т.е. обрабатывают ввод и генерируют вывод.
- подключается к AtomMind Server во время загрузки или в любой иной необходимый момент и начинает обрабатывать команды сервера. Эти команды переключат переменные чтения/записи и вызывают выполнение функций.

- асинхронно запускает события в контексте Агента, используя метод `Context.fireEvent()` после установления соединения с сервером.

Соответствующие классы

Класс	Описание
`AbstractClientController`	Абстрактный базовый класс контроллера соединения по протоколу AtomMind на стороне сервера.
`Agent`	Первичный класс структуры соединений на основе агента.
`AgentImplementationController`	Агентская версия контроллера соединения по протоколу AtomMind на стороне сервера.
`DefaultClientController`	Реализация по умолчанию контроллера соединения по протоколу AtomMind на стороне сервера.

15.1.8.1 Реализация Java Agent

Чтобы реализовать свой собственный Agent, следует добавить следующий код в Ваше java-приложение:

1. Создайте экземпляр `RemoteLinkServer` и задайте адрес/порт AtomMind Server, а также имя учетной записи пользователя, используемую для регистрации и пароль Агента.

   В AtomMind Server API 1340, поле **пароля** объекта `RemoteLinkServer` используется как пароль для пользовательского аккаунта 478.

   Однако, класс `Agent` использует поле пароля как пароль для учетной записи устройства 497 на сервере. Он не должен соответствовать паролю пользовательской учетной записи владельца Агента.

   Если учетная запись Агента с именем, заданным в конструкторе класса `Agent`, уже существует, и пароли аккаунта Агента и приложения Агента не совпадают, Агент не сможет "войти" на сервер.

2. Создайте экземпляр `Agent` и задайте имя учетной записи устройства 497 Агента в конструкторе.
3. Получите экземпляр `Context`, используя `Agent.getContext()` и добавьте определения/реализации переменных, функций и событий, используя метод `addVariableDefinition()`, `addFunctionDefinition()` и `addEventDefinition()`.
4. Вызовите `Agent.connect()`, чтобы заставить Ваш Агент подключиться к AtomMind Server и авторизоваться/аутентифицироваться. Этот вызов может завершиться неуспешно по ряду причин, включая недоступность сервера, неправильно указанный адрес/порт сервера, отсутствующие или отключенные драйверы Агента сервера, несовместимость с версиями Java-библиотек, отсутствие пользовательского аккаунта и неправильно указанный пароль аккаунта устройства Агента.
5. Вызывайте периодически метод `Agent.run()`. Он шлёт единственную команду на сервер и завершается, поэтому его следует вызывать в-основном из отдельного потока. На этом этапе будут вызываться Ваши собственные `VariableGetter`, `VariableSetter` и `FunctionImplementation`. Если метод `run()` генерирует `DisconnectionException`, значит связь с сервером прервана из-за ряда причин, и соединение следует восстановить.
6. Когда метод `Agent.getContext().isSynchronized()` вернет значение true, Ваше приложение может начать генерировать события Агента, используя метод `Agent.getContext().fireEvent()`.
7. Чтобы отсоединиться от сервера и остановить Агент, следует вызвать `Agent.disconnect()`.

Если некоторые события генерируются, когда Агент не подключен к серверу или еще не синхронизирован, эти события должны сохраняться локально (в памяти или области долгосрочного хранения) и отправляться в AtomMind Server при первой возможности.

Предоставление исторических значений

Многие Agents имеют периодическое или нестабильное соединение с AtomMind Server. Так как Agents продолжают получать и/или производить данные, когда соединение с сервером отсутствует, им необходимо отправлять на сервер журнал исторических событий после восстановления соединения.

Если Agent поддерживает отчеты об исторических событиях, он должен определить getHistory() function 689. Эта функция будет вызываться сервером вскоре после установления соединения. Функция должны возвращать список исторических значений с их временными метками.

Agent не должен асинхронно сообщать о каких-либо изменениях значений (см. раздел Информирование сервера об обновлении переменных ниже), пока не убедится, что функция getHistory() была вызвана, и сам Agent отвечает пустой таблицей (т.е. "данных больше нет"). Если этого не сделать, возникнет ошибка в каналах статистики 718 AtomMind Server.

Информирование сервера об обновлении переменных

Агент может уведомить сервер, что значение определенной переменной изменилось. Такое уведомление вызовет внеплановое обновление кэша сервера, которое происходит между периодами синхронизации 514.

Для отправки уведомления об обновлении переменной используйте событие обновленное 84 (пример для Агента на базе Java):

```
DataTable newTemperatureValue = new SimpleDataTable(agent.getContext().getVariableDefinition("temp

agent.getContext().fireEvent(AbstractContext.E_UPDATED, "temperature", newTemperatureValue);
```

Обновленное событие автоматически генерируется, если вызвать `agent.getContext().setVariable("temperature", newTemperatureValue);`

Разъединение с сервером

В определенных случаях (например, если были изменены настройки учетной записи устройства агента сервера) сервер может принудительно закрыть соединение TCP с Agent. Последний должен суметь обнаружить подобную ситуацию снова восстановить связь с сервером.

Подтверждение события

Класс `AgentContext` определяет событие `eventConfirmed`, которое генерируется каждый раз, когда AtomMind Server подтвердил успешную доставку события, сгенерированного Agent, и обработку путем вызова функции Agent confirmEvent() 689. Возможно добавить слушателя для события `eventConfirmed` и отреагировать на подтверждения доставки события, например, путем удаления доставленных событий Agent из внутренней базы данных Agent.

15.1.9 SDK компонента Виджет

SDK компонента Виджет AtomMind - это часть SDK AtomMind, которая позволяет реализовать компоненты Виджет AtomMind на языке программирования Java.

Компонент Виджет AtomMind - это особый тип плагина 207 AtomMind. Технически он включает в себя как минимум пять элементов:

- *Главный класс плагина* расширяет класс `ComponentPlugin`. Это главный класс плагина, который предоставляет доступ к классам компонента.
- *Класс компонента* расширяет класс `WAbstractComponent`. Он содержит свойства компонента.
- *Класс контекста* расширяет класс `WAbstractContext`. Этот класс предоставляет компонент Виджет как контекст.
- *Класс отрисовки* расширяет класс `DefaultSwingComponentRenderer`. Класс отрисовки используется для того, чтобы предоставлять компонент Виджет в стиле Swing.
- *Компонент дескриптор плагина* 1344, который определяет свойства плагина компонента и его место в иерархии плагинов AtomMind Server или AtomMind Client.

Пакет SDK AtomMind включает в себя пример реализации компонента Виджет с открытым исходным кодом AtomMind, который называется *пользовательский индикатор выполнения*. Он находится в пакете `examples.component` и содержит семь файлов:

- `CustomProgressBar.java` - исходный код *главного класса плагина*
- `CustomProgressBarSwingRenderer.java` - исходный код *класса отрисовки*
- `WCustomProgressBar.java` - исходный код *класса компонента*

- `WCustomProgressBarContext.java` - исходный код *класса контекста*
- `gb_custom_progress_bar.png` - иконка компонента
- `plugin.xml` - компонент *дескриптор плагина* 1344
- `build.xml` - файл Ant с единственной задачей по сборке JAR-файла компонента

Чтобы протестировать драйвер:

- Запустите `build.xml`, используя Ant, чтобы создать `custom-progress-bar.jar`
- Скопировать `custom-progress-bar.jar` в `%AtomMind Server Installation Folder/plugins/component`, когда AtomMind Server не запущен
- Скопировать `custom-progress-bar.jar` в `%AtomMind Client Installation Folder/plugins/component`, когда AtomMind Client не запущен
- Запустить AtomMind Server и AtomMind Client
- Создать новый виджет и редактировать его в редакторе виджетов 423
- Найти новый компонент во вкладке тулбара *Пользовательские компоненты*

15.1.9.1 Реализация компонента виджета

Чтобы создать новый компонент Виджет с нуля, вам необходимо создать несколько классов Java.

Класс компонента

У класса компонента Виджет нет логики, он содержит свойства компонентов и их геттеры (getters) и сеттеры (setters). Этот класс унаследован от `WAbstractComponent`, который определяет некоторые свойства (*ключ, описание, высота, ширина* и другие). Эти поля являются общими, подходят для большинства типов компонентов. *Класс компонентов* позволяет определить дополнительные свойства компонентов и их геттеров и сеттеров.

Пожалуйста, обратите внимание, что любой сеттер в преемнике класса `WAbstractComponent` должен создавать событие с информацией об изменениях.

Пример: Сеттер для *значения* свойства:

```
public void setValue(int value)
{
  int oldValue = this.value;
  this.value = value;
  firePropertyChange("value", oldValue, value);
}
```

Эти события необходимы для обработки изменений значений в видимых частях компонента.

Убедитесь, что вы реализовали метод `clone()`. Некоторые виды свойств требуют дополнительных шагов в клонировании, в этом случае вы можете их сделать. Но обычно реализация выглядит так:

Пример: метод `clone()` для `WNewComponent`, который унаследован у `WAbstractComponent`:

```
public WNewComponent clone()
{
  return (WNewComponent) super.clone();
}
```

Не забудьте реализовать метод `getComponentGroup()`. Он определяет группу компонентов для нового компонента. Предлагается *пользовательская* группа, но вы можете выбрать любую другую.

Пример: реализация для `getComponentGroup()` по умолчанию; подходит практически всегда:

```
public ComponentGroup getComponentGroup()
{
  return ComponentGroup.CUSTOM;
}
```

Класс контекстов

Оболочка контекста для компонента Виджет. Этот класс должен быть унаследован от `WAbstractContext`. Он предоставляет возможность работать с компонентом Виджет как с контекстом2102. Например, для того, чтобы представить свойства компонентов в редакторе свойств377. Реализация этого класса главным образом относится к определению переменных1386. Перепишите метод `createVariableDefinitions()`, чтобы добавить описание переменной в этот контекст.

Пример: *реализация для* `createVariableDefinitions()`:

```
protected void createVariableDefinitions()
{
  super.createVariableDefinitions();
  addDefaultVariableDefinition(VD_VALUE);
}
```

Используйте `addDefaultVariableDefinition(VariableDefinition vd)` только для свойств компонента по умолчанию1274. Для других свойств используйте `addVariableDefinition(VariableDefinition vd)`.

Класс отрисовщика

Класс отрисовщика используется для представления компонента Виджет в стиле Swing. Он унаследован от `DefaultSwingComponentRenderer`, который определяет общее поведение и функциональность для разных реализаций визуализации. Например, у него есть поле *Компонент,* которое будет использоваться в каждом *отрисовщике элемента*.

Класс отрисовщика должен реализовывать метод `createRenderer()`. Вы можете создать *компонент Swing,* настроить его и добавить слушателей, чтобы управлять событиями. Если некоторые изменения свойств требуют выполнения дополнительных действий, вы можете переписать метод `componentPropertyChanged(String property)`.

Пример: создание отрисовщика для *индикатора выполнения*:

```
public JProgressBar createRender()
{
  final JProgressBar progressBar = new JProgressBar();
  progressBar.setValue(getComponent().getValue());

  if (getRendererSupport().isInteractive())
  {
    progressBar.addChangeListener(new ChangeListener()
    {
      @Override
      public void stateChanged(ChangeEvent e)
      {
        getComponent().setValue(progressBar.getValue());
      }
    });
  }

  return progressBar;
}
```

Пример: использование метода `componentPropertyChanged(String property)` для выполнения дополнительных действий после изменения свойств:

```
public void componentPropertyChanged(String property)
{
  super.componentPropertyChanged(property);

  if (property.equals("value"))
  {
    progressBar.setForeground(getColor());
    getRendererSupport().renderChanged();
  }
}
```

Класс поддержки отрисовщика

Класс поддержки отрисовщика является опциональным. Он унаследован от `DefaultEditorComponentRendererSupport`. Его реализация может пригодиться, если вам нужно заставить отрисовщик представить изменения свойств в релевантном элементе в редакторе виджетов[423]. В большинстве случаев в этом нет необходимости, потому что `DefaultEditorComponentRendererSupport` реализует автоматическое выполнение соответствующих геттеров в элементе и сеттеров в объекте представления. Если автоматическая реакция не подходит для некоторых свойств, вы можете переписать метод `componentPropertyChanged(String property)` и реализовать собственного обработчика.

Главный класс плагинов

Главный класс плагинов предоставляет доступ к классам компонента. Он должен быть унаследован от `ComponentPlugin`. Необходимо переписать как минимум несколько методов, чтобы предоставить функциональность компонентов.

- `getWComponent()` должен возвращать *класс компонентов*
- `getSwingRenderer()` должен возвращать *класс отрисовщика*
- `getEditorRendererSupport()` должен опционально возвращать *класс поддержки отрисовщика*. Переписывание этого метода важно лишь в случае реализации *класса поддержки отрисовщика*.
- `getId()` должен возвращать ID компонента опционально. Реализация по умолчанию использует ID плагина из дескриптора, чтобы сгенерировать ID компонента.
- Методы `doStart()` и `doStop()` должны быть переписаны на ваше усмотрение.

Иконка компонента

Для представления новому компоненту Виджет нужна иконка. Редактор виджетов[423] AtomMind использует изображения формата PNG одного размера (20x20) как иконки компонентов. Таким образом, вы можете добавить изображение в файл компонента JAR и использовать с помощью конструктора `WAbstractComponent`. Обратите внимание, что параметр `iconId` должен быть таким, как и имя файла иконки без расширений.

Модификация дескриптора плагина

Чтобы создать дескриптор плагина[1344] для вашего пользовательского компонента, измените следующее в демо-файле компонента `plugin.xml`:

- Измените последнее слово в атрибуте `id` тэга `<plugin>` на ID нового плагина. ID должен содержать только строчные буквы, цифры и нижнее подчеркивание. Например, если требуемый ID вашего плагина `xyz`, установите атрибут ID на `com.tibbo.aggregate.common.plugin.component.xyz`
- Измените атрибут `class` тэга `<plugin>` на полное имя *главного класса плагина*
- Внесите описание компонента в тело тэга `<doc-text>`
- Измените атрибут `id` тэга `<extension>` на новый ID плагина

Создание и внедрение архива плагинов

Измените следующее в файле Ant `build.xml`:

- Измените имя проекта и имя файла назначения (файл плагина JAR)
- Запустите `build.xml`, используя Ant, чтобы создать файл плагина JAR
- Скопируйте файл JAR в `%AtomMind Server Installation Folder/plugins/component`, когда AtomMind Server не запущен
- Скопируйте файл JAR в `%AtomMind Client Installation Folder/plugins/component`, когда AtomMind Client не запущен
- Запустите AtomMind Server и AtomMind Client
- Создайте новый вмджет и измените его в редакторе виджетов[423]
- Найдите новый компонент во вкладке тулбара *Пользовательские компоненты*

15.1.10 SDK веб-приложения

Веб-приложение AtomMind - это часть набора средств разработки AtomMind, которая позволяет реализовать веб-приложения. Технически, в дополнение к веб-приложению, реализуется особая версия плагина 207 AtomMind, которая выполняет регистрацию веб-приложений. Таким образом, использование этого SDK дает результат в двух файлах: плагин AtomMind и веб-приложение на основе *технологии Java-сервлет*.

Java-сервлеты используются для увеличения возможностей серверов, которые размещают приложения, доступные посредством программной модели запроса-ответа. Хотя сервлеты могут отвечать на любой тип запроса, обычно они расширяют приложения, которые размещают веб-сервера. Для таких приложений технолоогия Java-сервлет определяет зависящие от HTTP классы сервлетов.

Реализованные веб-приложения могут взаимодействовать с контекстами AtomMind Server, могут быть размещены на том же интегрированном веб-сервере, как и Web UI 220 (поэтому они используют одинаковые настройки 354).

Пакет SDK AtomMind включает в себя пример веб-приложения с открытым кодом доступа AtomMind, который называется *демо веб-приложение*. Он находится в пакете `examples.webapp` и содержит следующие файлы:

- `DemoWebAppContextPlugin.java` - исходный код *класса плагина*
- `DemoWebApplication.java` - исходный код *класса веб-приложения*
- `DemoWebAppServlet.java` - исходный код *класса Java-сервлет*
- `build.xml` - файл Ant с однозадачным созданием файлов JAR и WAR
- `plugin.xml` - компонент *дескриптор плагина* 1344
- `web.xml` - дескриптор внедрения веб-приложения

Чтобы испытать драйвер:

- Запустите `build.xml`, используя Ant, чтобы создать `demo-web-app.jar` и `demo-web-app.war`
- Скопируйте `demo-web-app.jar` в `%AtomMind Server Installation Folder/plugins/context`, когда AtomMind Server не запущен
- Скопируйте `demo-web-app.war` в `%AtomMind Server Installation Folder/admin`, когда AtomMind Server не запущен
- Запустите AtomMind Server
- Выполните запрос, используя следующий URL: `http://localhost:8080/demo-web-app`

Создание веб-приложения требует дополнительные файлы в обеспечении пути в дополнение к `aggregate-api.jar` и `aggregate-api-libs.jar`:

- `extensions-libs.jar` может быть найден в `%AtomMind Server Installation Folder/jar`
- `server-core.jar` может быть найден в `%AtomMind Server Installation Folder/jar`
- `webserver.jar` может быть найден в `%AtomMind Server Installation Folder/plugins/contexts`

15.1.10.1 Реализация веб-приложения

Чтобы создать новое веб-приложение AtomMind с нуля, необходимо реализовать несколько классов Java.

Класс веб-приложения

Класс веб-приложения должен реализовывать интерфейс `WebApplication`. Перепишите методы `deploy()` и `stop()`, чтобы зарегистрировать и отменить регистрацию веб-приложения.

Пример: регистрация веб-приложения:

```
public void deploy() throws AggreGateException
{
  context = WebServerContextPlugin.registerWebApplication(WEBAPP_CONTEXT_PATH, WEBAPP_FI
}
```

Обратите внимание, что путь контекста используется в запросе к этому приложению. Второй параметр - это имя веб-приложения (файл WAR).

Пример: отмена регистрации веб-приложения:

```
public void stop() throws AggreGateException
{
  if (context != null)
  {
    WebServerContextPlugin.unregisterWebApplication(context);
  }
}
```

Используйте ранее сохраненный контекст, чтобы отменить регистрацию веб-приложения.

Класс плагина

Класс плагина должен быть унаследован у `AbstractContextPlugin`. Вам потребуется переписать один из наследуемых методов, чтобы добавить веб-приложение на веб-сервер.

Пример: использование метода *globalInit*, чтобы добавить веб-приложение:

```
public void globalInit(Context rootContext) throws PluginException
{
  WebServerContextPlugin.addAdditionalWebApplication(new DemoWebApplication());
}
```

Этого будет достаточно для многих веб-приложений. Для всех остальных это обычный класс плагина. Можно прочитать больше о его реализации в соответствующей теме 1342.

Класс Java-сервлет

Класс Java-сервлет должен быть унаследован от `HttpServlet`. Этот класс и его методы являются ответственными за процессы запросов и ответов. Детализированную информацию о реализации сервлета можно найти в соответствующей документации https://tomcat.apache.org/tomcat-7.0-doc/servletapi/javax/servlet/http/HttpServlet.html.

Пример: Простая реализация *doGet*:

```
public void doGet(HttpServletRequest request, HttpServletResponse response) throws Servl
{
  response.getWriter().println("test");
}
```

Особое внимание необходимо обратить на взаимодействие сервлета и AtomMind Server. Посмотрите на пробную версию веб-приложения в наборе средств разработки, чтобы найти примеры взаимодействия контекстов и использования прав доступа пользователей.

Измение дескриптора плагина

Чтобы создать дескриптор плагина 1344 для веб-приложения, измените следующее в демо-файле `plugin.xml`:

- Измените последнее слово в атрибуте `id` тэга `<plugin>` для нового ID плагина. ID должен содержать только строчные буквы, цифры и символы подчеркивания. Например, если ID вашего предполагаемого плагина - это `xyz`, вам понадобится установить ID атрибут в `com.tibbo.aggregate.common.plugin.context.xyz`
- Измените атриуб `class` тэга `<plugin>` на полное имя *класса плагина*
- Введите описание компонента в тело тэга `<doc-text>`
- Измените атрибут `id` тэга `<extension>` на новый ID плагина

Изменение дескриптора внедрения

Дескриптор внедрения описывает, как должно внедряться веб-приложение. Чтобы его создать, нужно изменить следующее в демо-файле `web.xml`:

- Изменить содержимое элемента `<display-name>`

- Изменить элемент `<servlet-name>` для элементов `<servlet>` и `<servlet-mapping>`
- Изменить `<servlet-class>` на полное имя *класса Java-сервлета*
- Индивидуализировать сопоставление сервлета и установить другие настройки внедрения при необходимости

Больше информации о внедрении веб-приложения можно найти на веб-сайте https://tomcat.apache.org/tomcat-7.0-doc/appdev/deployment.html.

Создание и внедрение архива плагинов

Следуйте этим шагам, предназначенным для файла `build.xml` из примера:

- Измените имя проекта и имена файлов назначения (файлы JAR и WAR)
- Запустите `build.xml`, используя Ant, чтобы создать файлы JAR и WAR
- Скопируйте файл JAR в `%AtomMind Server Installation Folder/plugins/context`, когда AtomMind Server не запущен
- Скопируйте файл WAR в `%AtomMind Server Installation Folder/admin`, когда AtomMind Server не запущен
- Запустите AtomMind Server
- Выполните требования для веб-приложения

Обратите внимание, что после первого запуска веб-приложение будет распаковано из файла WAR в папку с подходящим именем. Веб-сервер работает с нераспакованным приложением из папки. Таким образом, если файл WAR меняется, удалите папку со старой версией приложения до запуска AtomMind Server.

Создание веб-приложения требует дополнительные файлы в пути сборки в дополнение к `aggregate-api.jar` и `aggregate-api-libs.jar`:

- `extensions-libs.jar` может быть найден в `%AtomMind Server Installation Folder/jar`
- `server-core.jar` может быть найден в `%AtomMind Server Installation Folder/jar`
- `webserver.jar` может быть найден в `%AtomMind Server Installation Folder/plugins/contexts`

15.2 Руководство по разработке

Этот раздел освещает общие техники программирования на Java, помогающие освоить AtomMind SDK 1339. Эти техники полезны для:

- Разработки на сервере (плагины 1342, драйверы устройств 1345, скрипты сервера 879)
- Разработки на клиенте (скрипты виджетов 1308)
- Разработки сторонних приложений (AtomMind Server API 1340, Агенты 1360)

15.2.1 Основные концепции

SDK AtomMind предлагает определенный подход к управлению в системе данными любого типа, как сырыми данными, поступающими из устройств, так и подготовленными данными из различных системных элементов, таких как тревоги 779 или датчики 218. Этот подход также применяется к любому из решений, например, AtomMind Network Manager, AtomMind SCADA/HMI, AtomMind Access Control и другие.

Архитектура AtomMind

Технически AtomMind - это набор приложений и компонентов (библиотек). Далее следуют два основных приложения *Java,* которые создают платформу:

- AtomMind Server

- AtomMind Client

Автономные компоненты ПО могут встраиваться во внешнее ПО и бывают:

- API сервера 1340
- SDK агента 1360

Любой код, используемый другими частями SDK AtomMind (например, Набор для разработки драйверов 1345, SDK плагина 1342, SDK компонента виджета 1362), запускается или в AtomMind Server JVM, или в AtomMind Client JVM.

Формальное описание AtomMind доступно на странице Спецификация 2118.

Операции с таблицами данных

- Основной элемент всех данных в системе - Таблица данных 49. Даже простые скалярные значения (целые числа, строки, логические значения) представлены Таблицами данных с одной ячейкой с целью сохранения однородности. Запись Таблицы данных в java - это класс `DataTable`. Он включает в себя экземпляры от нуля до нескольких экземпляров `DataRecord`, которые представляют собой отдельные записи.
- Формат 49 `DataTable` представлен классом `TableFormat`. Он ссылается на ноль или несколько экземпляров `FieldFormat`, определяющих формат отдельных полей таблицы.

Работа через контексты

В целом, доступ ко всем данным должен осуществляться через различные контексты 41 сервера. Контексты организованы в виде иерархического дерева контекстов 41. SDK включает интерфейс `Context` для обращения с разными операциями контекста. Экземпляры контекстов можно выбрать из `ContextManager` по их пути (полным именам). У интерфейса `Context` также есть ряд методов доступа к родительскому или дочернему узлу.

Доступ к переменным, функциям и событиям контекста

- Каждый контекст предоставляет доступ к его переменным 61, функциям 70 и событиям 73. Существует ряд методов для их получения: `getVariableDefinition()`, `getVariableDefinitions()`, `getFuncitonDefinition()`, `getFuncitonDefinitions()`, `getEventDefinition()`, `getEventDefinitions()`. Классы, представляющие определения - это соответственно `VariableDefinition`, `FunctionDefinition` и `EventDefinition`.
- Интерфейс контекста предоставляет способ для чтения/записи данных переменных: `getVariable()` и `setVariable()`. Значение каждой переменной - это экземпляр `DataTable`.
- Контекст также предоставляет метод для вызова его функций: `callFunction()`. Входные и выходные параметры функции - это экземпляры `DataTable`.
- События обрабатываются путем добавления и удаления получателей события посредством методов `addEventListner()` и `removeEventListener()`. Получатели событий должны расширять интерфейс `ContextEventListerner`, но настоящие реализации в большинстве случаев расширяют `DefaultContextEventListener`. Получатели, работающие внутри плагинов и драйверов сервера, должны передать экземпляр `UncheckedCallerController` конструктору `DefaultContextEventListener`, чтобы обеспечить должные права доступа.

Метод `getVariable()` возвращает копию значения переменной `DataTable` даже в случае исполнения кода на стороне сервера (плагины, драйверы). Чтобы применить новое значение к переменной контекста, сначала создайте копию таблицы и после внесения в нее изменений вызовите `setVariable()` соответствующего контекста.

Прямые изменения `DataTable`, возвращенные `getVariable()`, приведут к неизбежной ошибке.

15.2.2 Основные классы и интерфейсы

Ниже приведенная таблица кратко описывает самые важные классы и интерфейсы AtomMind SDK.

Имя класса	Описание
`RemoteServer`	Контейнер для параметров соединения сервера (адрес, порт, имя пользователя и его пароль). Экземпляр этого класса передается в конструктор `RemoteServerController`.
`RemoteServerController`	Этот класс используется для установки соединения и осуществления контроля за ним с AtomMind Server. Он предоставляет доступ к интерфейсу `ContextManager`, который, в свою очередь, предоставляет доступ к контекстному дереву сервера.

ContextManager	Интерфейс, который предоставляет доступ к контекстному дереву. Экземпляр класса, реализующего этот интерфес, можно получить из RemoteServerController посредством метода getContextManager().
Context	Интерфейс, который позволяет работать с отдельным контекстом. Экземпляры классов, реализующих этот интерфейс, возвращаются различными методами ContextManager.
VariableDefinition	Определение переменной 61 контекста. Предоставляет доступ к его свойствам, формату, объектам получения и установки значений.
FunctionDefinition	Определение функции 70 контекста. Предоставляет доступ к его свойствам, форматам ввода/вывода и реализации
EventDefinition	Определение события 73 контекста. Предоставляет доступ к его свойствам и формату.
VariableGetter	Реализация данного интерфейса должна предоставлять логику чтения для значения пользовательской переменной.
VariableSetter	Реализация данного интерфейса должна предоставлять логику записи значения пользовательской переменной.
FunctionImplementation	Реализация данного интерфейса должна предоставлять логику исполнения пользовательской функции, т.е. обработки ввода и генерирования вывода.
ContextEventListener	Основной интерфейс для приемников событий контекста. Метод обращения приемника handle() вызывается, когда событие запускается в контексте, к которому был добавлен приемник.
DefaultContextEventListener	Реализация ContextEventListener по умолчанию. Как правило, большинство приемников должны расширять этот класс.
DataTable	Интерфейс, определяющий операции, применимые к Таблицам данных 49. Включает операции по предоставлению доступа к формату и записям таблицы. Экземпляры данного класса возвращают методы getVariable() и callFunction() , относящиеся к Context.
AbstractDataTable	Макетная реализация интерфейса DataTable. Реализует методы, которые считаются общими для большинства Таблиц данных.
SimpleDataTable	Первичная реализация DataTable на основе перечня записей данных.
DataRecord	Реализация одной записи Таблицы данных.
TableFormat	Класс, реализующий формат 49 Таблицы данных. Включает свойства таблицы и список полей.
FieldFormat	Класс, реализующий формат 49 отдельного поля Таблицы данных. Определяет имя, тип и другие параметры поля, включая *валидаторы*, *значения выбора* и пр.
CallerController	Экземпляр данного класса передается множеству операций контекста. Он включает в себя права доступа запрашивающей стороны.
UncheckedCallerController	Особый тип контроллера запрашиваюшего объекта, который запрещает любую проверку прав доступа, предоставляя полный доступ запрашивающей стороне. Должно использоваться для системных вызовов.
Permissions	Список всех прав доступа (объекты Permission), которые должны быть удовлетворены во время доступа к определенным ресурсам сервера.
ServerPermissionChecker	Этот класс предоставляет набор статических методов для получения Permissions, чтобы назначить для только что добавленных определений переменной/функции/события.

15.2.3 Нахождение и оценка произвольных данных

В этом разделе объясняется, как найти, прочитать или изменить данные в AtomMind.

1. Обнаружение необходимого контекста

Первый этап -- это определение места расположения контекста, который предоставляет необходимые настройки, операции или события. Существует несколько способов, как этого достичь:

- Проверьте ссылку контекста ядра сервера 1450 и/или ссылки контекста для вертикального решения (также являются частью этой документации)
- Найдите необходимый контекст в Системном Дереве AtomMind Client 359 или Web UI 220

2. Обнаружение необходимой переменной/функции/события

- Если контекст, содержащий необходимые данные, был обнаружен в ссылке контекста, проверьте раздел Открытые Переменные, Открытые Функции или Открытые События соответствующей главы по имени переменной/функции/события, содержащих необходимые данные.
- Всплывающая подсказка для рядов и ячеек компонента Редактор отображает имена переменных;
- Всплывающая подсказка для событий в компоненте Журнал Регистрации Событий отображает контекст ресурса события и его имя;
- Имена функций можно просмотреть, используя компонент Селектор Объектов;
- Редактор свойств предлагает операцию **Просмотр Информации о Переменной** в конекстном меню, в то время как Журнал Регистрации Событий включает в себя возможность **Просмотра Определения События**. Эти операции отображают все свойства определений, их формат и пр.

3. Определение имени и типа полей

Поскольку каждое значение является Таблицей Данных, необходимо определить, какие поля имеют этот формат. Это включает имена полей и их типы.

- Если нужное определение обнаружено в ссылке контекста, его описание будет содержать **таблицу формата**, описывающую имена полей, типы и назначение.
- При использовании UI, следует учитывать следующее:
  - Значения переменных, данные событий или функции ввода/вывода всегда открываются в компоненте Редактор Таблиц Данных. Всплывающие подсказки колонки, ряда и ячейки в Редакторе Таблиц Данных отображают имена и типы полей;
  - Операции Просмотр Информации о Переменной и Просмотр Определения Переменной (см выше) предоставляют подробную информацию обо всех свойствах полей, включая их значения по умолчанию, значения выборки и пр.

4. Осуществление доступа к данным программным путем

После нахождения необходимого контекста, перменной/функции/события и их полей, доступ к ним может осуществляться программным путем, как с плагинов сервера, так и используя удаленный API:

- Получить необходимый объект `Context`, используя метод `ContextManager.get()`, `Context.getChild()` или `Context.getParent()`.

Чтобы программным путем выстроить имена/пути контекста сервера, используйте методы статического помощника, определенные в классе `ContextUtils`, и константы, определенные в классе `Contexts`.

Пример: `String alertContextPath = ContextUtils.alertContextPath("admin", "myAlert");`

- Получить/установить переменную, вызвать функцию добавленного получателем события, используя методы `getVariable()`, `setVariable()`, `callFunction()`, `addEventListner()`.
- Чтобы построить новое значение переменной значения ввода функции с нуля, используйте следующий код: `new SimpleDataTable(VariableDefinition.getFormat())` или `new SimpleDataTable(FunctionDefinition.getInputFormat())`.
- При обработке `DataTable` можно получить определенный `DataRecord`, используя метод `getRecord()`. Максимально быстро получить первую запись можно, используя метод `rec()`.
- `DataRecord` предоставляет общий метод `getValue()`, а также метод `setValue()` для чтения/записи значений полей. Однако, существует ряд методов для получения значения поля разных типов: `getInt()`, `getString()`, `getBoolean()` и пр.

Для ссылки на упомянутые переменные/функции/события/действия сервера и их поля используйте константы, определенные в членах пакета `com.tibbo.aggregate.common.server`.

Пример: `String serverVersion = rootContext.getVariable(RootContextConstants.V_VERSION).rec().getString(RootContextConstants.VF_VERSION_VERSION);`

15.2.4 Журналирование

Библиотеки AtomMind ведут отчет о своей деятельности через журналирование[166]. Для того, чтобы активизировать и настроить журналирование в приложении Java, которое использует AtomMind Server API[1340] или Agent SDK[1339], следует вызвать метод `Log.start()` и передать ему `URL` конфигурационного файла журналирования[167]. Вы можете скопировать этот файл из директории установки AtomMind Server или AtomMind Client. Редактирование данного файла можно осуществлять путем изменения уровня журналирования в определенных категориях[168] с целью отладки.

Возможно также использование журналирования в серверных скриптах или скриптах виджета. Журналирование, созданное скриптами сервера, будет добавлено в серверный файл системного журнала (`logs/server.log`). Журналирование скрипта виджета будет добавлено в файл журнала AtomMind Client (`logs/client.log`) или в консоль *java*-апплет (при использовании Web UI[220]).

Все категории в файлах настройки журналирования по умолчанию настроены на передачу сообщений в INFO и на более высокие уровни. Поэтому чтобы написать определенную строку в файл журналирования, вы можете использовать следующий код:

```
Logger.getLogger("ag.mycategory").info("Hello world!");
```

Стандартные категории журналирования

Стандартные категории журналирования, которые используются основными модулями и некоторыми плагинами AtomMind, определяются в классе `Log`. В нем есть много преднастроенных статистических логгеров, которые могут быть использованы напрямую:

```
Log.DATABASE.info("Database problem detected");
```

Журналирование путем событий информации

Иногда удобным может быть использование событий[73] системы для журналирования какой-либо информации. У данных журналирования путем событий есть несколько преимуществ:

- Системные операторы смогут видеть вывод журналирования в журналах событий или даже пользовательских интерфейсах операторов
- Журналирование будет привязано к особому контексту[41], который представляет устройство, ресурс системы или определенный пользователем ресурс
- Вывод журналирования будет постоянно храниться в БД[692] сервера.

Используйте программное поколение событий информации[77], чтобы активизировать журналирование развития на базе событий. Каждый экземпляр события информации содержит строку производственной информации, которая может включать ваш собственный вывод.

15.2.5 Работа с таблицами данных

Этот раздел помогает освоить программное управление таблицами данных[49]. Он разделен на следующие части:

- Реализации таблиц данных[1372]
- Создание[1373] таблиц данных (определение формата, добавление записей, установление значений в ячейках и т.д.)
- Манипулирование[1376] таблицами данных (организация цикла записей, получение доступа к значениям ячеек, клонирование таблиц и т.д.)

15.2.5.1 Реализации таблиц данных

Существует несколько реализаций интерфейсов `DataTable`:

простая таблица данных

SimpleDataTable - это первичная реализация DataTable. В качестве базовой структуры данных она использует ArrayList записей данных.

Данная реализация поддерживает все операции из DataTable интерфейса, а также реализует интерфейс Cloneable.

В отличие от других реализаций DataTable, экземпляр SimpleDataTable всегда знает количество записей, которые в ней содержатся. Таким образом, getRecordCount() всегда возвращает ненулевое неотрицательное целое значение.

таблицы данных ResultSet

В качестве базовой структуры данных ResultSetDataTable использует экземпляр подкласса java.sql.ResultSet. По сути, это реализация DataTable вида реляционной БД, которая сама по себе представлена данным экземпляром ResultSet.

Данная реализация поддерживает все операции доступа и большинство операций модификации, определенных интерфейсом DataTable. Точный набор поддерживаемых операций зависит от движка БД, драйвера JDBC, а также от используемого типа результирующих данных и настроек многопоточности. Сортирующие операции не поддерживаются.

Количество записей в экземпляре ResultSetDataTable неизвестно до полного выполнения итерации над базовым набором результатов. Когда количество записей неизвестно, getRecordCount() возвращает null.

Таблица данных фильтрации

FilteringDataTable - это реализация DataTable, которая служит фильтрующей оболочкой для другого экземпляра DataTable. Экземпляр FilteringDataTable не хранит никаких данных, а использует в качестве источника данных другую DataTable. Только те данные исходной таблицы, которые удовлетворяют заданному Expression фильтру, будут считаться принадлежащими соответствующему экземпляру FilteringDataTable.

Данная реализация не поддерживание операции модификации.

Количество записей в экземпляре FilteringDataTable неизвестно до полного выполнения итерации над таблицей с исходными данными. Когда количество записей неизвестно, getRecordCount()возвращает null.

Прокси-таблицы данных

ProxyDataTable - это DataTable, которая представляет экземпляр ResultSetDataTable или FilteringDataTable на стороне Клиента 359. Основная цель использования ProxyDataTable - получение доступа к запрашиваемым записям соответствующей таблицы на сервере 144, не перенося содержание всей таблицы с сервера в клиент.

Данная реализация не поддерживание операции модификации

Количество записей в экземпляре ProxyDataTable неизвестно до полного выполнения итерации над ее записями данных. Когда количество записей неизвестно, getRecordCount() возвращает null.

15.2.5.2 Создание

Операции, связанные с созданием объекта DataTable, зависят от того, какой интерфейс DataTable реализован.

SimpleDataTable

Для создания SimpleDataTable необходимо определить ее формат путем создания экземпляра класса TableFormat:

```
TableFormat tf = new TableFormat(); // For multi-row tables

TableFormat tf = new TableFormat(1, 1); // For single-row tables
```

В большинстве случаев экземпляры TableFormat должны быть объявлены статически, должны конфигурироваться в статических блоках и быть повторно использованы между несколькими таблицами, переменными, функциями, событиями и т.д. Это обеспечивает меньшее потребление памяти в сравнении с нестатическими форматами таблиц.

ДОБАВЛЕНИЕ ПОЛЕЙ

После этого необходимо добавить дескрипторы полей к формату таблицы. Каждое поле представлено экземпляром `FieldFormat`. Существует множество способов создания дескриптора формата поля:

```
FieldFormat ff = FieldFormat.create("fieldName", FieldFormat.STRING_FIELD); // The simplest method
ff.setDescription("Field Description");
ff.setDefault("Default Value");
ff.setNullable(true);

FieldFormat ff = FieldFormat.create("fieldName", FieldFormat.STRING_FIELD, "Field Description", "D

FieldFormat ff = FieldFormat.create("<fieldName><S><D=Field Description><A=Default value><F=N>");

// After field format is created, we can fine-tune it
ff.addSelectionValue("v1", "First option");
ff.addSelectionValue("v2", "Second option");
ff.addValidator(new LimitsValidator(10, 100));
ff.setKeyField(true);

//Now we can add our field descriptor to field format:
TableFormat tf = new TableFormat();
tf.addField(ff);
```

Также возможно избежать создания `FieldFormat` и передать опции поля в `TableFormat.addField()`:

```
TableFormat tf = new TableFormat();

tf.addField("fieldName", FieldFormat.STRING_FIELD, "Field Description", "Default Value", true);
```

Установка опций формата

Когда добавляются все поля, мы можем настроить формат таблицы, добавить к ней несколько привязок 741 и валидаторов:

```
tf.setReorderable(true);

tf.addBinding("field2", "{field1} * 2");

tf.addTableValidator(new TableKeyFieldsValidator());
```

Создание таблицы

Когда формат таблицы готов, можно создать таблицу на его основе:

```
TableFormat tf = new TableFormat(1, 1, FieldFormat.create("fieldName", FieldFormat.STRING_FIELD));

DataTable table = new SimpleDataTable(tf);
```

Когда используется экземпляр `TableFormat` для создания таблицы, он становится **постоянным** и не может модифицироваться. Для модификации формата существующей `table`:

- Клонируйте формат таблицы: `TableFormat cloned = table.getFormat().clone();`
- Клонированный формат будет изменяемым. Внесите в него необходимые модификации.
- Создайте новую таблицу, используя клонированный формат: `DataTable newTable = new SimpleDataTable(cloned);`
- Скопируйте 58 данные из старой таблицы в новую, используя методы статического помощника `DataTableReplication.copy(table, newTable)` класса `DataTableReplication`

Добавление записей

Существует несколько способов создания записей.

Первый способ - установка значений поля:

```
DataRecord rec = table.addRecord();
```

```
rec.setValue("firstField", "Value of first field"); // First field is a string-type field
rec.setValue("secondField", true); // First field is a boolean-type field
rec.setValue("thirdField", 12345); // First field is an integer-type field
```

Второй способ - передача значений поля в `addRecord()`:

```
DataRecord rec = table.addRecord("Value of first field", true, 12345);
```

Третий способ - добавление значений по одному:

```
DataRecord rec = table.addRecord();
rec.addString("Value of first field");
rec.addBoolean(true);
rec.addInt(12345);

// Explicit type specification is not necessary
DataRecord rec = table.addRecord();
rec.addValue("Value of first field");
rec.addValue(true);
rec.addValue(12345);
```

АВТОМАТИЧЕСКИЙ ТИП ПРЕОБРАЗОВАНИЯ

Иногда значение, которое у вас есть, может не полностью подходить для ячейки таблицы. В этом случае используйте метод `setValueSmart()` от `DataRecord`. Он справится с задачей преобразования значения в подходящий тип.

Код внутри использует статические методы класса `Util`, который может быть напрямую использован, чтобы активировать преобразование необработанных объектов в булевые значения, числа и даты.

ResultSetDataTable

Для создания экземпляра `ResultSetDataTable`, необходим экземпляра класса, который реализует `java.sql.ResultSet`. Следующий пример показывает, как создать `ResultSetDataTable`, используя MySQL JDBC соединение:

```
DriverManager.registerDriver(((Driver) Class.forName("com.mysql.jdbc.Driver").newInstance())); //
Connection con = DriverManager.getConnection("jdbc:mysql://localhost/databasename?user=root&passwo
Statement stmt = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_UPDATABLE

String query = "SELECT * FROM TABLENAME;"; // here it is supposed that there is a table called "TA
ResultSet resultSet = stmt.executeQuery(query);

DataTable table = new ResultSetDataTable(resultSet);
// Do not forget to close the instances of ResultSet, Statement, and Connection when they are no l
```

Обратите внимание, что формат создаваемой таблицы данных формируется автоматически на основе схемы связей в наборе результатов.

ДОБАВЛЕНИЕ ЗАПИСЕЙ

Чтобы добавить записи в `ResultSetDataTable`, лежащий в основе набор результатов должен быть одновременно с прокруткой (то есть не должен быть типа `ResultSet.TYPE_FORWARD_ONLY`) и обновляемый (`ResultSet.CONCUR_UPDATABLE`).

Записи можно добавить теми же способами, что и для `SimpleDataTable`, с одним исключением: метод `addRecord()` (без аргументов) не поддерживается. Вместо него следует использовать метод `addRecord(DataRecord)`:

```
DataRecord rec = new DataRecord(table.getFormat);
rec.addString("Value of first field");
rec.addBoolean(true);
rec.addInt(12345);
table.addRecord(rec);
```

FilteringDataTable

В предыдущей главе говорилось, что экземпляр `FilteringDataTable` не хранит данные. Чтобы создать экземпляр `FilteringDataTable`, необходимы, как минимум, два объекта: таблица данных, которая будет служить источником данных, и выражение фильтрации, которое определит условие(я) для фильтра. Выражение фильтрации может быть либо типа `Expression`, либо `String`. Последнее будет автоматически преобразовано в `Expression` (при условии, что String - корректное Выражение[112], которое вычисляется в Boolean).

Обратите внимание, что экземпляр `DataTable`, переданный конструктору `FilteringDataTable`, становится постоянным.

Ниже приведен пример создания `FilteringDataTable`:

```
// Let's suppose we have a SimpleDataTable called "source" that has an Integer field called "intfi

String filterExpression = "{intfield} == 111 || {intfield} == 333";
DataTable filteringTable = new FilteringDataTable(source, filterExpression);

// The created FilteringDataTable will have only the records from source that satisfy the conditio
```

`FilteringDataTable` не поддерживает операции модификации, то есть добавление и настройка записей невозможны.

ProxyDataTable

Экземпляр `ProxyDataTable` создается автоматически в Клиенте при попытке получить доступ к экземпляру `ResultSetDataTable` или `FilteringDataTable` на Сервере.

`ProxyDataTable` не поддерживает операции модификации, то есть добавление и настройка записей невозможны.

15.2.5.3 Манипулирование

доступ к данным

Для получения доступа к данным ячеек сначала получите `DataRecord`:

```
DataRecord rec = table.getRecord(5); // The numbering of records is zero-based, so this will retur

DataRecord rec = table.rec(); // This returns first record (with zero index)
```

Теперь вы можете использовать методы, которые возвращают определенные типы значений, для получения значений ячеек:

```
String s1 = rec.getString("firstField");

Boolean b1 = rec.getBoolean("secondField");

Integer i1 = rec.getInt("thirdField");
```

итерация по записям

Часто бывает необходимо обработать все записи таблицы. `DataTable` легко выполняет эту задачу с помощью интерфейса `Iterable`:

```
for(DataRecord rec : table)
{
  // Process the record
}
```

Для экземпляра `ResultSetDataTable` на основе набора результатов типа `ResultSet.TYPE_FORWARD_ONLY`, разрешена только одна операция доступа, то есть только одна итерация по записям данных. Для таких таблиц невозможно получить запись данных каким-либо другим способом (например, методами `rec()` или `getRecord(int)`). Также невозможна любая другая операция, требующая изменения координат положения курсора в наборе результатов (например, методы `findIndex(DataTableQuery)` или `isOneCellTable()`).

Клонирование таблиц и элементов таблиц

Метод `clone()` работает по-разному для разных подтипов `DataTable`:

Для `SimpleDataTable`, метод возвращает полную копию экземпляра. Тип клонированного экземпляра также `SimpleDataTable`.

Для экземпляров таблиц `ResultSetDataTable`, `FilteringDataTable` или `ProxyDataTable`, метод возвращает `SimpleDataTable`, содержащую клоны записей, представленных в оригинальной таблице на момент клонирования.

Обратите внимание, что метод `clone()` в `ResultSetDataTable`, `FilteringDataTable` и `ProxyDataTable` не вызывает `super.clone()`. Помните, что если вы решите расширить любой из этих классов, вызов `super.clone()` в дочернем классе может не дать ожидаемого результата копирования.

Также возможно клонировать запись или формат таблицы:

```
DataTable clonedTable = table.clone();

DataRecord clonedRecord = table.rec().clone();
TableFormat clonedFormat = table.getFormat().clone();
```

Кодирование и декодирование таблиц

Кодирование таблиц данных работает по-разному для разных подтипов `DataTable`:

Любую `SimpleDataTable` можно закодировать в строку и восстановить без потери данных. Экземпляр `SimpleDataTable` кодируется совместно со всеми ее записями.

Для кодирования и декодирования `SimpleDataTable` используйте следующий код:

```
String encodedTable = table.encode(false); // false means don't use visible separators

DataTable restored = new SimpleDataTable(encodedTable);
```

Экземпляры `ResultSetDataTable` или `FilteringDataTable` кодируются по-другому. Их записи не кодируются, вместо этого кодируются их идентификаторы. Эти ID позволяют соответствующей `ProxyDataTable` расположить таблицу на Сервере и получить доступ к нужным записям.

Экземпляры `ProxyDataTable` не кодируются, так как не требуют такой операции.

Отбор яйчеек таблиц

Для нахождения первой записи, имеющей значение поля Integer `field1`, равное 123, используйте следующий код:

```
DataRecord rec = table.select("field1", 123);
```

Если такая запись не найдена, данный метод вернет `null`.

Сортировка таблиц

Сортировать модно только экземпляры `SimpleDataTable`. Для сортировки `SimpleDataTable` в соответствии со значениями поля `field1`, используйте этот код:

```
table.sort("field1", true); // Ascending sort order
```

Для сортировки таблицы данных отличного от `SimpleDataTable` типа, сначала клонируйте ее, чтобы получить `SimpleDataTable` копию оригинальной таблицы, а затем отсортируйте клонированный экземпляр.

Дублирование таблиц

Часто бывает необходимо копировать данные в разных таблицах. Прежде всего, `DataTable` можно клонировать (см. выше).

Однако более расширенные операции дублирования таблиц доступны как статические методы класса `DataTableReplication`:

- Методы семейств `copy()` копируют данные из одной таблицы в другую, сохраняя ключевые поля, доступные только для чтения и не предназначенные для дублирования поля, минимальное/максимальное количество записей и прочее
- Методы семейств `copyRecord()` дублируют данные из одной `DataRecord` в другую, также сохраняя большинство опций

Расширенные операции

Некоторые расширенные операции таблицы, такие как создание таблиц из списков значений или обработка привязок[735] таблицы, доступны в двух вспомогательных классах с открытым исходным кодом: `DataTableBuilding` и `DataTableUtils`.

15.2.6 Работа с контекстами

Первоначальный объект, предоставляющий доступ к дереву контекстов[41], реализует интерфейс `ContextManager`. Существует несколько методов получения менеджера контекстов.

Получение менеджера контекстов

Метод получения экземпляра *ContextManager* зависит от текущей среды:

- При разработке приложения с помощью AtomMind Server API[1340] используйте `RemoteServerController.getContextManager()` для получения дерева контекстов удаленного сервера
- При разработке драйвера устройства[1345] используйте `DeviceContext.getContextManager()` для получения менеджера дерева контекстов
- При разработке плагина[1342] менеджер контекстов передается как параметр при помощи методов `install(ContextManager cm)` и `deinstall(ContextManager cm)`. В случае применения методов `install(ServerContext context)` и `deinstall(ServerContext context)` используйте `ServerContext.getContextManager()`.
- При разработке Агента[1360] на Java помните, что у вашего Агента есть только один контекст. Однако получить доступ в его менеджер контекстов можно через `Agent.getContext().getContextManager()`.
- При написании скрипта сервера[879] вызовите `ScriptExecutionEnvironment.getScriptContext().getContextManager()` для получения доступа в менеджер контекстов сервера.
- При написании скрипта виджетов[1308] вызовите `WidgetScriptExecutionEnvironment.getEngine().getServerContextManager()` для получения доступа в менеджер контекстов удаленного сервера. Также можно вызвать `WidgetScriptExecutionEnvironment.getComponentContext("componentName").getContextManager()` для получения доступа в менеджер контекстов дерева контекстов компонентов виджета.

Права доступа

Экземпляр объекта `CallerController`, включающий текущие права доступа пользователя, **должен** передаваться в большинство вызовов, относящихся к контексту, если вы пишете код, работающий в виртуальной машине Java AtomMind Server (т.е. при разработке скрипта сервера или плагина). Если этого не сделать, результаты различных исключений отказа в доступе будут нулевыми.

Для получения более подробной информации см. Работа с правами доступа[1392].

Получение индивиуальных контекстов

Есть два вида основных путей, помогающие контексту иметь свой собственный путь[42]:

- Вызвать `get(String path)` или метод `get(String path, CallerController caller)` от `Context`
- Вызвать `get(String path)` или метод `get(String path, CallerController caller)` от `ContextManager`

Они обычно возвращают тот же самый результат, если только не будут использованы в распределенной архитектуре[1332]. Однако в распределенной среде их поведение будет полностью иным:

- Методы интерпретации ContextManager предоставляют строку path как локальный путь на сервере, которому принадлежит ContextManager.
- Методы интерпретации Context предоставляют строку path как путь на удаленном сервере, которому принадлежит Context. Он валиден, если контекст прокси представляет удаленное соседнее устройство посредством распределенной архитектуры.

Поэтому вызов методов get() от Context - это предпочтительное поведение. Вызов методов get() от ContextManager часто может быть причиной непредсказуемых результатов в распределенной инсталляции.

Пример получения контекста с указанным путем.

```
Context deviceContext = anotherContext.get("users.admin.devices.device1");
```

Метод get() вернет *null*, если запрашиваемый контекст не существует или недоступен с этими правами доступа.

Использование метода get() от Context, чтобы получить другой контекст, означает, что всегда должны быть контексты, из которых можно вызвать этот метод. Это может означать дополнительное планирование и обдумывание на протяжении фазы разработки, в то время как код, который использует методы get() от ContextManager, все еще хорошо работает в нераспределенной среде разработки. Однако в основном этот ошибочный код в дальнейшем перестает работать в нераспределенной рабочей среде.

Используйте Context.getParent() и Context.getChild(String name) при работе с особым контекстом, чтобы иметь доступ к их родительским и дочерним контекстам соответственно.

Используйте узел ContextManager.getRoot(), чтобы получить доступ к корню контекста дерева контекстов.

ИСПОЛЬЗОВАНИЕ ВСПОМОГАТЕЛЬНЫХ МЕТОДОВ

Константы, определенные в классе Contexts, и статические методы класса ContextUtils обеспечивают удобные способы для построения путей различных контекстов сервера, таких как *Тревоги* или контексты Device.

Пример: `Context alertContext = contextManager.get(ContextUtils.alertContextPath("admin", "myalert"));`

ОБРАБОТКА СПИСКА ДОЧЕРНИХ КОНТЕКСТОВ

Для получения списка дочерних контекстов определенного контекста вызовите Context.getChildren().

1. Вызов сервера getChildren(CallerController caller) вернет только те дочерние контексты, которые доступны вызывающему пользователю (представленному в виде CallerController).
2. Вызов getChildren() прокси-контекста (т.е. при использовании удаленного API или разработке Агента) вернет только те дочерние контексты, которые доступны пользователю, аутентифицированному в текущем соединении сервера.

Получение контекстной информации

Интерфейс Context имеет несколько методов получения основной информации о контексте:

- getName()
- getPath()
- getDescription()
- getType()
- getGroup()

Контексты прокси

Если вы разрабатываете код, работающий в AtomMind Client или в любом другом приложении и получаете Context посредством API, результатом будет получение *контекста прокси*, который соединяется с удаленным соседним устройством на стороне сервера. Таким образом, даже наипростейшие вызовы метода могут стать результатом ввода/вывода сети и надолго отложатся или даже прекратят работу из-за ошибок сети.

15.2.6.1 Манипулирование переменными

Этот раздел объясняет, как манипулировать переменными[61] контекста программным путем.

Нахождение доступных переменных

Существует несколько способов получения списка переменных из `Context`:

- `getVariableDefinitions()` - этот метод вернет список всех переменных в контексте
- `getVariableDefinitions(CallerController caller)` - этот метод вернет список переменных, доступных *вызывающему* (как для чтения, так и для записи)
- `getVariableDefinitions(String group)` - этот метод вернет список переменных, принадлежащих определенной *группе* или ее подгруппе
- `getVariableDefinitions(CallerController caller, String group)` - этот метод вернет список переменных, принадлежащих определенной *группе* и доступных *вызывающему*

```
List<VariableDefinition> variables = context.getVariableDefinitions(ContextUtils.GROUP_REMOTE); //
```

Для извлечения одного определения переменной по ее имени вызовите метод `getVariableDefinition(String name)`. Метод `getVariableDefinition(String name, CallerController caller)` вернет определение или ноль, если *переменная* недоступна *вызывающему*.

Переменные чтения и записи

Интерфейс `Context` имеет два основных метода манипулирования значениями переменных:

- `getVariable()`
- `setVariable()`

Эти методы позволяют читать и записывать значения переменных контекста.

ЧТЕНИЕ ПЕРЕМЕННЫХ

Метод `getVariable()` читает значения перменной и возвращает `DataTable`, представляющую эти значения. Она принимает один или два аргумента: имя переменной для чтения и опциональный экземпляр `CallerController`, который включает права доступа вызывающей стороны (см. Работа с правами доступа[1392]).

При получения таблицы значений переменной вы получаете доступ к ее данным. См. Манипулирование таблицами данных[1376].

Вот пример чтения значения переменной:

```
DataTable variableValue = context.getVariable("tabularVariable");
```

ЗАПИСЬ ПЕРМЕННЫХ

Для настройки переменной существует два основных способа:

- Путем предоставления предварительно подготовленного объекта `DataTable`. Этот объект можно создать с нуля программным способом или получить и модифицировать через `getVariable()`.
- Путем предоставления отдельных значений ячеек.

Пример 1 - настройка значения переменной с использованием встроенной таблицы данных:

```
DataTable variableValue = new SimpleDataTable(context.getVariableDefinition("tabularVariable").get

variableValue.addRecord().addString("str1").addInt(111); // Adding a record
variableValue.addRecord().addString("str2").addInt(222); // Adding another one

context.setVariable("tabularVariable", variableValue); // Setting variable value
```

Пример 2 - модификация предыдущего значения переменной:

```
DataTable variableValue = context.getVariable("tabularVariable");

variableValue.addRecord().addString("str1").addInt(111); // Adding a record
```

```
variableValue.addRecord().addString("str2").addInt(222); // Adding another one

context.setVariable("tabularVariable", variableValue); // Setting variable value
```

Пример 3 - настройка значения переменной с использованием списка значений ячеек:

```
context.setVariable("tabularVariable", "str1", 111); // Supplying cell values for the first record
```

Если вы хотите изменить лишь одну ячейку или запись значения переменной, недостаточно прочитать ее значение и изменить полученную `DataTable`. Необходимо записать измененное значение обратно в исходный контекст при помощи вызова `setVariable()`.

Проще говоря, **любая** модификация переменной требует вызова `setVariable()`.

Получение доступа к истории переменных

Существует два способа получить доступ к историческим значениям переменных, хранящихся в базе данных:

- Вызвать функцию variableHistory 1613 из контекста Утилиты 1613. Этот способ работает как локально (внутри драйверов/плагинов сервера), так и удаленно (через API сервера).
- Вызвать статический метод Java `ServerContextUtils.getVariableHistory()`. Этот способ чуть быстрее предыдущего, но работать он будет только внутри виртуальной машины Java (т.е. в драйверах/плагинах).

Полная сигнатура метода `ServerContextUtils.getVariableHistory()`:

```
public static Iterator<Pair<Date, DataTable>> getVariableHistory(ServerContextManager cm, CallerController caller,
String context, String variable, Date fromDate, Date toDate, Integer maxResults,

    boolean sortAscending) throws DaoException
```

ВЫЗОВ ФУНКЦИИ VARIABLEHISTORY

Функция `variableHistory` контекста **Утилиты** возвращает таблицу данных 49, каждый ряд которой содержит временную метку и историческое значение переменной. См. ее описание здесь 1613.

```
DataTable history = utilsContext.callFunction(UtilitiesContextConstants.F_VARIABLE_HISTORY, getCal

for (DataRecord rec : history)
{
  // Process historical values
}
```

ИСПОЛЬЗОВАНИЕ КЛАССА УТИЛИТ КОНТЕКСТА СЕРВЕРА

Статический способ класса ServerContextUtils, который возвращает исторические значения переменной, имеет следующую сигнатуру:

```
public static Map<Date, DataTable> getVariableHistory(CallerController caller, String context, Str
      throws DaoException, ContextException, DataTableException
```

Он возвращает карту временных меток, указывающую время, когда были сохранены значения и сами значения.

Использование вспомогательных констант

Интерфейсы, которые расположены в пакете `com.tibbo.aggregate.common.server`, предоставляют константы строк, которые подходят именам большинства переменных контекста сервера и их полям ввода/вывода.

Использование этих констант всегда предпочтительнее, чем использование констант строк, определенных в вашем собственном коде. Это обеспечит безошибочный код, если некоторые переменные или поля будут переименованы или пермещены в будущих версиях AtomMind Server.

15.2.6.2 Вызов функций

Этот раздел объясняет, как составить список и вызвать функции[70] контекста программным путем.

Нахождение доступных функций

Существует несколько способов получения списка функций из `Context`:

- `getFunctionDefinitions()` - этот метод вернет список всех функций в контексте
- `getFunctionDefinitions(CallerController caller)` - этот метод вернет список функций, доступных *вызывающему*
- `getFunctionDefinitions(String group)` - этот метод вернет список функций, принадлежащих определенной *группе* его подгрупп
- `getFunctionDefinitions(CallerController caller, String group)` - этот метод вернет список функций, принадлежащих определенной *группе* и доступных ее *вызывающему*

```
List<FunctionDefinition> functions = context.getFunctionDefinitions(ContextUtils.GROUP_REMOTE); //
```

Для извлечения одного определения функции по ее имени вызовите метод `getFunctionDefinition(String name)`. Метод `getFunctionDefinition(String name, CallerController caller)` вернет определение или ноль, если *функция* недоступна *вызывающему*.

Выполнение функций

Для вызова функции используйте метод `Context.callFunction()`. Он принимает следующие аргументы: имя функции для вызова, опциональный экземпляр `CallerController`, который включает права доступа вызывающей стороны (см. Работа с правами доступа[1392]) и опциональные параметры функции.

Существует два способа определения параметров ввода функции:

- Путем предоставления встроенного объекта `DataTable`. Этот объект может быть с нуля создан программным путем при помощи формата, получаемого через `FunctionDefinition.getInputFormat()`.
- Путем предоставления отдельных значений ячеек таблицы данных ввода функции.

Метод `callFunction()` возвращает `DataTable`, представляющую вывод функции.

Пример 1 - вызов функции путем предоставления встроенной таблицы данных:

```
DataTable input = new SimpleDataTable(context.getFunctionDefinition("sendSms").getInputFormat());

input.addRecord().addString("+12345678900").addInt("First message"); // Adding a record
input.addRecord().addString("+98765432100").addInt("Second message"); // Adding another one

DataTable output = context.callFunction("sendSms", input); // Calling function
```

Пример 2 - вызов функции путем предоставления списка значений ячеек таблицы ввода:

```
DataTable output = context.callFunction("sendSms", "12345678900", "First message"); // Supplying c
```

Использование вспомогательных констант

Интерфейсы, которые расположены в пакете `com.tibbo.aggregate.common.server`, предоставляют константы строк, которые подходят именам большинства функций контекста сервера и их полям ввода/вывода.

Использование этих констант всегда предпочтительнее, чем использование констант строк, определенных в вашем собственном коде. Это обеспечит безошибочный код, если некоторые функции или поля будут переименованы или перемещены в будущих версиях AtomMind Server.

15.2.6.3 Прослушивание и обработка событий

Этот раздел объясняет, как программным образом составить список событий[73] контекста, подписаться на него и обработать.

Выяснение доступных событий

Существует несколько способов получения списка событий из `Context`:

- `getEventDefinitions()` - этот метод вернет список всех событий в контексте
- `getEventDefinitions(CallerController caller)` - этот метод вернет список событий, доступных *вызывающему*
- `getEventDefinitions(String group)` - этот метод вернет список событий, принадлежащих определенной *группе* его подгрупп
- `getEventDefinitions(CallerController caller, String group)` - этот метод вернет список событий, принадлежащих определенной *группе* и доступных *вызывающему*

```
List<EventDefinition> events = context.getEventDefinitions(ContextUtils.GROUP_REMOTE); // Finding
```

Для извлечения одного определения события по его имени вызовите метод `getEventDefinition(String name)`. Метод `getEventDefinition(String name, CallerController caller)` вернет определение или ноль, если событие недоступно *вызывающему*.

Инициирование событий

Как только определенный компонент системы (например, драйвер устройства) определяет, что что-то произошло, он может *инициировать событие контекста*. Событие может быть запущено асинхронно из любого потока. Имейте в виду, что не существует общего способа удаленного запуска события (из AtomMind Client или используя AtomMind Server API).

Для запуска события вызовите метод `Context.fireEvent()`. Этот метод принимает следующие параметры:

- Имя события
- Уровень события
- Параметры события (в форме встроенной `DataTable` или списка значений ее ячеек)

Пример 1: запуск события путем подачи события `DataTable`:

```
DataTable eventData = new SimpleDataTable(context.getEventDefinition("myEvent").getFormat()); // C

eventData.addRecord().addString("str1").addInt(111); // Adding a record

context.fireEvent("myEvent", EventLevel.INFO, eventData);
```

Пример 2: запуск события путем подачи значения ячейки таблицы событий:

```
context.fireEvent("randomValue", EventLevel.INFO, new Float(Math.random() * 1000000));
```

Подписка на события

Модули сервера могут подписываться на события контекста, чтобы получать и обрабатывать их экземпляры. В большинстве случаев эта функциональность требуется при реализации настраиваемых плагинов[207] сервера.

Чтобы подписаться на событие, нужно:

- Создать экземпляр класса, реализующего интерфейс `ContextEventListener`
- Передать этот экземпляр методу `Context.addEventListener()`

Интерфейс `ContextEventListener` имеет множество методов, и в большинстве случаев разумнее наследовать слушателя вашего события из существующей реализации:

- При написании кода сервера (например, плагина) наследуйте слушателя вашего события из `ServerEventListener` для обеспечения правильного уровня прав доступа.
- В других случаях используйте `DefaultContextEventListener` в качестве базы.

Использование `DefaultContextEventListener` из кода, запущенного на сервере виртуальной машины Java, приведет к отсутствию получения каких-либо событий. Это происходит из-за того, что слушатель по умолчанию не получает необходимый уровень прав доступа.

МАССОВАЯ ПОДПИСКА ПУТЕМ МЕНЕДЖЕРА КОНТЕКСТОВ

Также возможно подписаться на события путем void `addMaskEventListener(String mask, String event, ContextEventListener listener)` метода `ContextManager`. У этого пути прдписки есть большое количество различий от подписки путем интерфейса `Context`:

- Возможно получать события от различных ресурсов, указывая маску 44 контекстов
- Если подписка была сделана путем `ContextManager`, определенный контекст был удален, а новый был создан с таким же именем, тот же самый слушатель будет автоматически добавлен в новый контекст

СЛАБЫЕ СЛУШАТЕЛИ

У большинства методов подписки есть подписи, которые принимают флажок `boolean weak`. Если определенный пользователь добавлен в качестве слабого, подписчик автоматически будет сборщиком мусора, если нет сильных ссылок на это.

Все слушатели по умолчанию не являются слабыми.

Обработка событий

Как только вы подписались на событие путем добавления слушателя, система будет вызывать метод слушателя `handle()` каждый раз, когда происходит событие. Таким образом, вся логика обработки должна добавляться к реализации этого метода:

```
DefaultContextEventListener listener = new DefaultContextEventListener()
{
  @Override
  public void handle(Event event) throws EventHandlingException
    {
       // Handling event here

       int level = event.getLevel();

       List<Acknowledgement> acknowledgements = event.getAcknowledgements();

       // Accessing event data
       DataTable eventData = event.getData();
    }
};

context.addEventListener("eventName", listener);
```

Получение доступа к историческим событиям

Существует два способа получить доступ к историческим событиям, хранящимся в базе данных сервера:

- Вызвать функцию получить 1516 из контекста события 1513. Этот метод сработает как локально (внутри драйверов/плагинов сервера), так и удаленно (через сервер API).
- Вызвать метод Java `Server.getDaoFactory().getEventDao().getEvents()`. Этот способ чуть быстрее, чем предыдущий, но он работает только внутри сервера виртуальной машины Java (т.е. в драйверах/плагинах).

ВЫЗОВ ФУНКЦИИ GET

Функция `get` контекста **События** возвращает таблицу данных 49, каждая строка которой содержит информацию об одном историческом событии. См. ее описание здесь 1516.

```
DataTable history = eventsContext.callFunction(EventsContextConstants.F_GET, getCallerController()

for (DataRecord rec : history)
{
  // Process historical events
}
```

ИСПОЛЬЗОВАНИЕ ПРЯМОГО ДОСТУПА К БАЗЕ ДАННЫХ

Метод класса `EventDao`, выполняющий загрузку событий из базы данных, имеет следующую сигнатуру:

```
public Iterator<Event> getEvents(ContextManager cm, CallerController caller, EntityList eventList,
      boolean sortAscending, Object... additionalCriteria) throws DaoException;
```

Этот метод принимает список типов событий для загрузки, предварительной фильтрации (additionalCriteria) и постфильтрации (filter), даты начала/окончания периода появления событий, максимальное количество событий для загрузки и опции сортировки. Параметры startDate, endDate и maxResults могут быть цстановлны на null для отключения временного диапазона и ограничения количества.

Он возвращает список объектов Event, представляющих хронологические события.

Пример использования:

```
EntityList events = new EntityList("users.admin.devices.my_device", "my_device_event");

Iterator<Event> eventItr = Server.getDaoFactory().getEventDao().getEvents(cm, caller, entities, nu

while (eventItr.hasNext())
{
  Event ev = eventItr.next();
  Date eventTime = ev.getCreationtime();

  DataTable eventData = ev.getData();

  // Processing
}
```

Использование вспомогательных констант

Интерфейс, который находится в пакете com.tibbo.aggregate.common.server, предоставляет константы строк, которые соединяются с большинством событий контекстов сервера и их полями.

Использование этих констант предпочтительнее, чем использование констант строк, определенных в вашем собственном коде. Это обеспечит безоткатные коды, если некоторые события или поля будут переименованы или перемещены в AtomMind Server в будущих версиях.

15.2.7 Объявление переменных, функций, событий и действи

Плагины 1342 и драйверы 1345 сервера вместе с Агентами на базе Java 1360 обычно создают их собственные определения переменных, функций и событий. Эти определения добавляются к контекстам при помощи методов addVariableDefinition(), addFunctionDefinition() и addEventDefinition(). Также существуют соответствующие методы удаления: removeVariableDefinition(), removeFunctionDefinition() и removeEventDefinition().

Не существует общего способа удаленно управлять/модифицировать структуру дерева контекстов сервера и определения переменных/функций/событий.

Таким образом, addChild(), removeChild(), addVariableDefinition(), removeVariableDefinition(), addFunctionDefinition(), removeFunctionDefinition(), addEventDefinition(), removeEventDefinition() и иные подобные методы, вызванные в прокси-контексте (например, через AtomMind Server API), не влияют на дерево контекстов сервера.

Использование статических форматов

Объявление переменных, функций и событий допускает уточнение TableFormat, т.е. формат значения переменных, функций ввода/вывода и данные событий.

В целом создание форматов описано в статье Создание таблиц данных 1373.

Однако важно понимать, что в большинстве экземпляров класса TableFormat класс должен быть повторно использован между значениями тех же самых переменных/функций/событий, добавленных во множественные контексты.

Поэтому в большинстве случаев экземпляры форматов таблицы должны быть статическими. Это существенно уменьшит использование памяти в случае, если, допустим, это же самое значение переменной было добавлено к десяти тысячам контекстов.

Типичный экземпляр TableFormat может занимать один килобайт памяти. Поэтому добавление десяти переменных с нестатическими форматами к десяти тысячам контекстов (например, экземпляры пользовательских ресурсов) займет сто мегабайт памяти, которую можно было бы сберечь при использовании единственного экземпляра в статическом формате.

15.2.7.1 Определение и реализация переменных

При определении переменных контекста ваших устройств/агентов/серверов необходимо задать свойства их определений[61], т.е. имя, описание, формат, флажки чтения/записи, уровень прав доступа, текст справки и группу. Для объявления новой переменной создайте экземпляр объекта `VariableDefinition` и задайте его свойства. Вот пример:

```
// Creating String field format
FieldFormat ff = FieldFormat.create("demoSettingField", FieldFormat.STRING_FIELD);

// Creating single-cell (scalar) setting
TableFormat format = new TableFormat(1, 1, ff);

// Creating variable (setting) definition. Note that variable group should not be changed.
VariableDefinition vd = new VariableDefinition("demoSetting", format, true, true, "Demo Setting",

// Setting permission level
vd.setPermissions(ServerPermissionChecker.getManagerPermissions());
```

После этого добавьте определение переменной к контексту:

- Переменные контекста драйвера устройства, соответсвующие настройкам доступа устройства, должны добавляться вызовом `Context.addVariableDefinition()` из метода `DeviceDriver.setupDeviceContext()`.
- Определения переменных контекста драйвера устройства, соответствующие свойствам устройства (переменные настроек устройства), должны возвращаться переопределенным методом `DeviceDriver.readVariableDefinitions()`.
- Плагины[207] сервера должны добавлять переменные из методов `install()` и `start()`.
- Агенты на базе Java должны добавлять переменные после создания объекта `Agent` через `Agent.getContext().addVariableDefinition()`.
- Наконец, скрипты (как скрипты сервера, так и виджета) обычно не должны добавлять никаких переменных.

Для переменных сервера также необходимо задать уровень прав доступа для чтения/записи[1392]:

```
// Setting permission level
vd.setPermissions(ServerPermissionChecker.getManagerPermissions());
```

Переменные настроек устройства, предоставляемые драйвером устройства[518], вместе с переменными Agent должны принадлежать группе `remote`. Если вы хотите сгруппировать их во вкладки, добавьте описание вкладки к имени группы, используя разделитель "|", например `remote|Power Management Settings`. Также возможно использовать вложенные группы, например `remote|Power Management Settings|Auto Power-off`.

Группа переменных определяется вызовом `VariableDefinition.setGroup()`. Можно использовать следующий синтаксис:

```
vd.setGroup(ContextUtils.createGroup(ContextUtils.GROUP_REMOTE, "Power Management Settings
```

Переменные, добавляемые вручную (кроме переменных настроек устройства), должны также иметь ненулевой *метод чтения*, реализующий интерфейс `VariableGetter`. Цель метода чтения - обеспечить пользовательский код чтения переменной. Записываемые переменные должны также иметь ненулевой *метод установщика*, реализующий интерфейс `VariableSetter`. Метод установщика должен каким-то образом сохранять новое значение переменной, измененнное системными операторами или различными интсрументами сервера.

Формат таблицы данных, возвращенный методом установщика переменной, должен соответствовать формату, содержащемуся в ее определении. Несоблюдение данного правила приведет к потере данных, поскольку система попытается сконвертировать таблицу данных в формат, предоставленный определением, сохраняя как можно большее количество данных.

Реализация чтения значения переменной

Этот раздел подробно описывает, как `Context` выстраивает и возвращает `DataTable`, представляющую значение переменной, когда кто-то вызывает метод `getVariable()`.

Вся логика, описанная здесь, реализуется в классе `AbstractContext`, включенном в комплект разработчика с открытым исходным кодом.

Чтобы вернуть результирующую DataTable из метода getVariable(), AbstractContext выполняет следующее:

1. Применяет *блокировку чтения переменной* для предотвращения одновременных операций чтения. Если любой другой поток читает переменную, текущий поток заблокируется до завершения предыдущей операции чтения.
2. Проверяет, имеет ли вызывающая сторона (определенная объектом CallerController) необходимый эффективный уровень прав доступа 486 в текущем контексте для чтения этой переменной.
3. Ищет текущий контекст (используя технологию отражения Java) и проверяет его на наличие "метода чтения" со следующей сигнатурой:

```
public DataTable getVariable (VariableDefinition def, CallerController caller,
RequestController request) throws ContextException
```

Variable - это имя читаемой переменной.

Если находится метод с такой сигнатурой, он вызывается. Этот метод должен вернуть DataTable, представляющую значение переменной.

4. Если "метод чтения" не был найден, наличие VariableGetter проверяется вызовом VariableDefinition.getGetter(). Если VariableGetter определен (не ноль), вызывается метод VariableGetter.get(). Метод чтения должен вернуть DataTable, представляющую значение переменной.
5. Если метод чтения переменной не определен (т.е. VariableDefinition.getGetter() возвращает ноль), AbstractContext вызывает метод getVariableImpl(). Этот метод может быть переопределен в подклассах AbstractContext. Если реализация переопределения "понимает" переменную (т.е. "знает" ее имя), она должна вернуть DataTable, представляющую значение переменной. В других случаях метод getVariableImpl() должен вернуть ноль.
6. Если метод getVariableImpl() вернул ноль, AbstractContext вызывает метод executeDefaultGetter(). Метод чтения по умолчанию возвращает значение переменной, которая хранилась в базе данных (для кода сервера) или была кэширована в память (для всех других случаев).
7. Если значение перменной до этого не кэшировалось, метод executeDefaultGetter() проверяет, определяется ли значение переменной по умолчанию (ненулевое) вызовом VariableDefinition.getDefaultValue(). Это значение должно быть ранее задано через VariableDefinition.setDefaultValue(DataTable value).
8. И, наконец, если значение по умолчанию не определяется в VariableDefinition, метод чтения по умолчанию выстраивает и возвращает значение по умолчанию вызовом new SimpleDataTable(variableDefinition.getFormat(), true).
9. DataTable, полученная из "метода чтения", методов VariableGetter, getVariableImpl() или метода чтения по умолчанию не сразу возвращается из getVariable(). Сначала она проверяется на соответствие формату переменной (доступно через VariableDefinition.getFormat(), этот шаг опускается, если возвращается ноль). Если формат таблицы данных соответствует формату, предоставленному определением (или расширяет его, добавляя некоторые поля), эта таблица возвращается в исходном виде. Если формат таблицы не соответствует и не расширяет формат определения, AbstractContext конвертирует таблицу в "нужный" формат и сохраняет максимальное количество данных. Это делается вызовом метода DataTableReplication.copy().

Если коду, реализующему "метод чтения", VariableGetter или getVariableImpl(), нужен доступ к значению переменной, хранящейся в базе данных или памяти, этот код должен напрямую вызывать метод executeDefaultGetter().

Реализация установки значения переменной

Этот раздел подробно описывает, как Context выстраивает и возвращает DataTable, представляющую значение переменной, когда кто-то вызывает метод setVariable().

Вся логика, описанная здесь, реализуется в классе AbstractContext, включенном в комплект разработчика с открытым исходным кодом.

Чтобы обработать DataTable, представляющую новое значение переменной при вызове метода setVariable(), AbstractContext выполняет следующее:

1. Применяет *блокировку записи переменной* для предотвращения одновременных операций записи. Если любой другой поток записывает переменную, текущий поток заблокируется до завершения предыдущей операции записи.
2. Проверяет, имеет ли вызывающая сторона (определенная объектом CallerController) необходимый эффективный уровень прав доступа 486 в текущем контексте для записи этой переменной.

3. DataTable, представляющая новое значение переменной, сначала проверяется на соответствие формату переменной (доступному через VariableDefinition.getFormat(), этот шаг опускается, если возвращается ноль). Если формат таблицы данных значения соответствует формату, предоставленному определением (или расширяет его путем добавления полей), эта таблица остается нетронутой. Если формат таблицы значений не соответствует и не расширяет формат определения, AbstractContext коновертирует таблицу в "нужный" формат и сохраняет максимальное количество данных. Это выполняется вызовом метода DataTableReplication.copy().

4. Если у вызывающей стороны включена проверка уровня прав доступа (т.е. как у обычного системного пользователь 478), система проверяет, чтобы значения всех полей "только для чтения" в новой таблице были равны значениям соответствующих полей в старой (текущей) таблице, которая внутренне извлекается вызовом getVariable(). Значения всех измененных полей "только для чтения" переустанавливаются на те, которые взяты из старой (текущей) таблицы.

5. Ищет текущий контекст (используя технологию отражения Java) и проверяет его на наличие "метода установщика" со следующей сигнатурой:

```
public void setVariable  (VariableDefinition def, CallerController caller,
RequestController request, DataTable value) throws ContextException
```

Variable - это имя записываемой переменной.

Если находится метод с такой сигнатурой, он вызывается. Этот метод должен вернуть DataTable, представляющую новое значение переменной.

6. Если "метод установщика" не находится, наличие VariableSetter проверяется при помощи VariableDefinition.getSetter(). Если определяется VariableSetter (не ноль), вызывается метод VariableSetter.set(). Метод установщика должен обработать DataTable,представляющую новое значение переменной.

7. Если метод установщика переменной не определен (т.е. VariableDefinition.getSetter() возвращает ноль) или метод установщика возвращает false, AbstractContext вызывает метод setVariableImpl(). Этот метод может быть переопределен в подклассах AbstractContext. Если реализация переопределения "понимает" переменную (т.е. "знает" ее имя), она должна обрабатывать DataTable, представляющую новое значение переменной и возвращать true. В ином случае метод setVariableImpl() должен вернуть false.

8. Если метод setVariableImpl() вернул false, AbstractContext вызывает метод executeDefaultSetter(). Метод установщика по умолчанию сохраняет новое значение переменной в базе данных (для кода сервера) или в кэше памяти (для всех других случаев).

Если коду, реализующему "метод установщика", VariableSetter или setVariableImpl(), нужно хранить значение новой переменной в базе данных или памяти, этот код должен напрямую вызывать метод executeDefaultSetter().

15.2.7.2 Определение и реализация функций

При определении функций контекста ваших устройств/агентов/серверов необходимо задать свойства их определений 70, т.е. имя, описание, формат ввода и вывода, уровень прав доступа, текст справки и группу. Для объявления новой функции создайте экземпляр объекта FunctionDefinition и задайте его свойства. Вот пример:

```
// Creating function input format (scalar, integer)
FieldFormat iff = FieldFormat.create("demoOperationInputField", FieldFormat.INTEGER_FIELD);
TableFormat inputFormat = new TableFormat(1, 1, iff);

// Creating function output format (scalar, string)
FieldFormat off = FieldFormat.create("demoOperationOutputField", FieldFormat.STRING_FIELD);
TableFormat outputFormat = new TableFormat(1, 1, off);

// Creating function (operation) definition. Note that function group should not be changed.
FunctionDefinition fd = new FunctionDefinition("demoOperation", inputFormat, outputFormat, "Demo O
```

После этого добавьте определение функции к контексту:

- Определения функций контекста драйвера устройства, соответствующие операциям устройства, должны возвращаться переопределенным методом DeviceDriver.readFunctionDefinitions(). Сервер добавит определения функции к контексту устройства и создаст для них определения действия 87.
- Плагины 207 сервера должны добавлять функции из методов install() и start().
- Агенты на базе Java должны добавлять функции после создания объекта Agent через Agent.getContext().addFunctionDefinition().

- Наконец, скрипты (как скрипты сервера, так и виджета) обычно не должны добавлять никаких функций.

Для функций сервера также необходимо определить уровень прав доступа[1392]:

```
// Setting permission level
fd.setPermissions(ServerPermissionChecker.getManagerPermissions());
```

Функции работы устройства, предоставляемые драйвером устройства[518], вместе с функциями Agent должны принадлежать группе `remote`. Если вы хотите сгруппировать их логически (и поместить во вложенное меню контекстов), добавьте описание группы к имени группы, используя "|" в качестве разделителя, например `remote|Maintenance Operations`. Также возможно использовать вложенные группы, например, `remote|Maintenance Operations|Daily`.

Группа функций определяется вызовом `FunctionDefinition.setGroup()`. Можно использовать следующий синтаксис:

```
fd.setGroup(ContextUtils.createGroup(ContextUtils.GROUP_REMOTE, "Maintenance Operations",
```

Функции, добавляемые вручную, должны иметь ненулевую *реализацию*, полученную из интерфейса `FunctionImplementation`. Реализация должна анализировать ввод функции, обрабатывать его и генерировать вывод.

Формат таблицы данных, возвращенный методом реализации функции, должен соответствовать формату вывода, содержащемуся в ее определении. Несоблюдение данного правила приведет к потере данных, поскольку система попытается сконвертировать таблицу данных в формат, предоставленный определением, сохраняя как можно большее количество данных.

15.2.7.3 Определение и реализация событий

При определении событий контекста ваших устройств/агентов/серверов необходимо задать свойства их определений[73], т.е. имя, описание, формат, текст справки, уровень прав доступа и группу. Для объявления нового события создайте экземпляр объекта `EventDefinition` и задайте его свойства. Вот пример:

```
// Creating event data format (scalar, string)
FieldFormat ff = FieldFormat.create("demoEventField", FieldFormat.STRING_FIELD);
TableFormat format = new TableFormat(1, 1, ff);

// Creating event definition
EventDefinition ed = new EventDefinition("demoEvent", format, "Demo Event", ContextUtils.GROUP_DEF
```

После этого добавьте определение событий к контексту:

- Определение событий контекста драйвера устройства, соответствующее событию устройства, должно возвращаться переопределенным методом `DeviceDriver.readEventDefinitions()`.
- Плагины[207] сервера должны добавлять события из методов `install()` и `start()`.
- Агенты на базе Java должны добавлять события после создания объекта `Agent` через `Agent.getContext().addEventDefinition()`.
- Наконец, скрипты (как сервера, так и виджетов) обычно не должны добавлять какие-либо события.

Для событий сервера также необходимо определить период их действия и уровень прав доступа[1392]:

```
// Setting permission level
ed.setPermissions(ServerPermissionChecker.getManagerPermissions());
```

События устройства, предоставленные драйвером устройства[518], вместе с событиями Agent должны принадлежать группе `remote`.

Группа событий определяется вызовом `EventDefinition.setGroup()`. Можно использовать следующий синтаксис:

```
ed.setGroup(ContextUtils.GROUP_REMOTE);
```

Генерация событий

Плагины, драйвера и Агенты могут генерировать события контекста с помощью метода `fireEvent()` интерфейса `Context`. Вот пример:

```
context.fireEvent("eventName", EventLevel.INFO, new Float(Math.random() * 1000000));
```

Формат таблицы данных, передаваемой методу `fireEvent()`, должен соответствовать формату в определении событий. При несоблюдении этого правила данные могут быть утеряны, поскольку система попытается конвертировать таблицу данных в формат, предоставленный определением, сохраняя как можно большее количество данных.

15.2.7.4 Определение и реализация действий

При определении действий контекста вашего сервера необходимо задать свойства определений, т.е. имя, описание, формат, уровень прав доступа, текст справки и группу.

Многие классы, которые относятся к действиям, чаще являются частями AtomMind Server, нежели Java SDK с открытым исходным кодом. Эти классы недоступны в исходном коде. Чтобы получить доступ к этим классам, необходимо добавить следующие файлы JAR к пути класса:

- *aggregate-commons.jar*
- *server-core.jar*

Эти файлы JAR можно найти в подпапке /jar установочной папки AtomMind Server.

Для объявления нового действия создайте экземпляр объекта `ServerActionDefinition` и задайте его свойства. Пример:

```
// Creating the definition and providing action implementation class
ServerActionDefinition ad = new ServerActionDefinition("configurationWizard", ActionImplementation

// Setting action description
ad.setDescription("Configuration Wizard");

// Setting permission level
ad.setPermissions(ServerPermissionChecker.getManagerPermissions());
```

После завершения добавьте определение действия к контексту путем вызова метода `Context.addActionDefinition()`. Плагины[207] и драйверы[518] сервера должны добавлять действия из методов `install()` и `start()`.

Реализация действий

Реализация действий - это класс, включенный в класс `ServerAction`. Он должен заменить метод `execute()`, выполняющий данное действие, т.е. скомбинировать операции сервера и взаимодействия с человеком-оператором.

Как только действие запускается оператором (или в автономном режиме), создается новый экземпляр объекта реализации действия и вызывается его метод `execute()`.

У иерархии `ServerAction` есть следующие методы предоставления доступа в среду действий, доступных во время выполнения:

- `getDefiningContext()` возвращает `Context` действие, которое было вызвано.
- `getActionDefintion()` возвращает `ActionDefinition` действия.
- `getActionContext()` возвращает экземпляр `ServerActionContext`, который содержит расширенную информацию о среде выполнения.

Вот пример простой реализации действия:

```
public class ShowReportAction extends ServerAction
{
  @Override
  public ActionResult execute(ServerActionInput parameters) throws ContextException
  {
    DataTable deviceStatusTable = getDefiningContext().getVariable("status", getCallerController()

    String status = deviceStatusTable.rec().getString("statusMessage");
```

```
    getProcessor().showMessage("Current Device Status", "Device Status: " + status);
  }
}
```

Взаимодействие с оператором

Когда запускается действие, оно произвольно решает "общаться" с человеком-оператором, который его запустил. Эти взаимодействия называются процедуры пользовательского интерфейса 88.

Все процедуры пользовательского интерфейса начинаются при помощи вызова различных методов объекта `ServerActionCommandProcessor`, который доступен из действия через метод `getProcessor()`.

Вот неполный список доступных вызовов процедур пользовательского интерфейса:

Метод ServerActionCommandProcessor	Процедура пользовательского интерфейса
browse()	Просмотреть 89
confirm()	Подтвердить 89
editCode()	Редактировать код 90
editData()	Редактировать данные 90
editProperties()	Редактировать свойства 91
editText()	Редактировать текст 93
launchWidget()	Запустить виджет 93
selectContext()	Выбрать объекты 94 (только выбор контекста активен)
selectEntities()	Выбрать объекты 94
showEventLog()	Показать журнал событий 96
showError()	Показать ошибку 95
showGuide()	Запустить интерактивные практические уроки 99
showMessage()	Показать сообщение 97
showReport()	Показать отчет 97
showSystemTree()	Показать системное дерево 98

15.2.8 Работа с выражениями

Выражения 112 широко используются в AtomMind, поэтому важно разбираться в обработке выражений с точки зрения программирования.

Выражение представлено классом `Expression`, который может быть создан из строки. Однако экземпляры `Expression` инкапсулируют анализируемое синтаксическое дерево. Поэтому если единственный экземпляр `Expression` вычисляется несколько раз (например, с разной средой), то разбор этого исходного текста предоставляется только раз.

Вычисление выражения, например, вычисление результатов, выполняется методами `evaluate()` класса `Evaluator`. Эти методы возвращают необработанные экземпляры `Object`, которые представляют собой результирующие значения. Однако есть методы, включающие преобразование результирующего значения в определенный тип: `evaluateToBoolean()`, `evaluateToColor()`, `evaluateToData()`, `evaluateToDataTable()`, `evaluateToDate()`, `evaluateToNumber()`, `evaluateToString()` и другие.

У объекта `Evaluator` есть карта распознавателей ссылок 117 (представленных интерфейсом `ReferenceResolver`). Большинство распознавателей пользователей являются стандартными распознавателями 118 (доступными посредством метода `getDefaultResolver()`) и распознавателями среды (доступными посредством метода `getEnvironmentResolver()`).

Большинство выражений запрашивает указание определенных настроек распознавателя по умолчанию, поэтому у класса `Evaluator` есть большое количество особых методов конструкторов и конфигураций, которые передают функциональность стандартному распознавателю и позволяют указывать:

- ContextManager, который используется для нахождения контекстов
- Context по умолчанию
- DataTable по умолчанию
- CallerController, представляющий права доступа пользователя/модуля, вычисляющего выражение

15.2.9 Управление правами доступа

В этом разделе говорится о программном подходе в решении вопросов безопасности и прав доступа 477 для AtomMind Server.

Доступ к контекстным данным

Многие методы интерфейсов Context и ContextManager допускают параметры типа CallerController. Контроллер вызывающего объекта - это объект, который включает в себя эффективные права доступа вызывающей стороны. Существует два общих правила использования контроллеров вызывающего объекта:

- Любой код сервера (драйверы устройства 1345 и плагины 1342) должны передать экземпляр CallerController каждому методу, который его способен принять. Если это не удается выполнить, на вызывающей стороне уровень права доступа не будет определен, и, таким образом, большинство вызовов завершаться ошибкой или возвратом нулевых значений.
- Любой удаленный код (приложения, использующие API AtomMind Server 1340 или Агенты на Java 1360) должен использовать методы, которые не принимают контроллеры вызывающего объекта или передают нулевые значения. Это будет отлично работать, поскольку код клиента не выполняет проверки прав доступа, в то время как запрошенная операция удаленного сервера будет тем не менее учитывать уровень прав доступа, полученный во время авторизации (вызывает функцию login() корневого контекста).

Получение контроллера вызывающего объекта

Код сервера может получить контроллеры вызывающего объекта несколькими способами:

- Создать экземпляр UncheckedCallerController, чтобы запретить любое право проверки и полного доступа. Это следует выполнить при помощи системного кода, который, как предполагается, не выполняет операции с правами доступа пользователя системы.
- Получить Контроллер вызывающего объекта, вызывая метод getCallerController() от ServerContext (или метод getCallerController() от DeviceContext). Этот контроллер будет разрешать права доступа системного пользователя 478, которому принадлежит учетная запись устройства 497. Этот способ получения контроллеров вызывающего объекта в большей степени применимо для кода Драйвера устройства 1345.

Расширение контекстов

Когда драйвер устройства плагина пользователя добавляет определения переменных, функции или события в контекст сервера, он может назначить для них пользовательские уровни доступа, используя методы VariableDefinition.setReadPermissions(), VariableDefinition.setWritePermissions(), FunctionDefinition.setPermissions() и EventDefinition.setPermissions(). Если уровень прав доступа для определения равно нулю, оно будет соответствовать уровню доступа родительского контекста. Обратите внимание, что настройка уровней доступа к определению ниже или равному уровню доступа к конексту является бессмысленным.

Чтобы получить определенные заранее значения уровней с различными правами доступа, используйте следующий код:

Уровень доступа	Код
Отсутствует	LinkServerPermissionChecker.getNullPermissions()
Наблюдатель	LinkServerPermissionChecker.getObserverPermissions()
Оператор	LinkServerPermissionChecker.getOperatorPermissions()
Администратор	LinkServerPermissionChecker.getManagerPermissions()
Инженер	LinkServerPermissionChecker.getEngineerPermissions()
Администратор	LinkServerPermissionChecker.getAdminPermissions()

Проверка прав доступа вручную

Иногда нужно проверить права доступа особых авторизованных сущностей (представленных `CallerController`). Это можно сделать с помощью инструмента проверки серверных прав доступа, представленным интерфейсом `PermissionChecker`.

Экземпляр инструмента проверки прав доступа может быть получен с помощью метода `getPermissionChecker()` от `ServerContextManager`, который, в свою очередь, возвращен методом `getContextManager()` правильно настроенных контекстов сервера (например, контексты, унаследованные от `BaseServerContext`, `EditableChildrenContext` или `EditableChildContext`).

Действующий уровень прав доступа[486] `CallerController` в особом контексте можно проверить с помощью метода `(CallerController caller, Permissions requiredPermissions, Context accessedContext)` от `PermissionChecker`. Далее представлен пример:

```
boolean checking = caller.isPermissionCheckingEnabled();

boolean admin = getContextManager().getPermissionChecker().has(caller, ServerPermissionChecker.get

if (!admin && checking)
{
  throw new ContextException("Not an administrator");
}
```

15.2.10 Жизненный цикл AtomMind Server

AtomMind Server - это автономное приложение Java, которое является ядром AtomMind. Его технические и программные требования прописаны на странице Требования[146].

15.2.10.1 Загрузка сервера

Процесс загрузки AtomMind Server оптимизирован для высокой производительности. Поэтому большинство стадий, которые занимают значительное время, задействуют многопоточное выполнение заданий.

Процесс загрузки содержит следующие основные стадии:

- Инициализация системы журналирования[166]. Журналирование ошибок файлов и консоли доступны с этого момента.
- Загрузка менеджера ресурсов. Она обеспечивает работу операций по ребрендированию фирменного стиля.
- Загрузка внешних библиотек JAR и JNI. Архивы Java не включают в себя плагины сервера.
- Загрузка и обработка настроек сервера[177].
- Проверка обновлений. Эта стадия может быть пропущена, если нет Интернет соединения.
- Инициализация системы безопасности[477].
- Загрузка и инициализация плагинов[207]. Вызывается метод всех плагинов контекста `ContextPlugin.initialize()`.
- Инициализация доступа к хранилищу данных сервера[692].
- Загрузка Отказоустойчивого кластера[1326]. Ведомые узлы отказоустойчивого кластера active-passive находятся на этой стадии до тех пор, пока работает первичный узел.
- Загрузка дерева контекстов[41]. Вызывается метод всех контекстов `Context.setupMyself()`. Все дочерние контексты любого зависящего контекста контейнера загружаются в этот же момент несколькими потоками.
- Фаза установки плагинов контекста, вызывается метод `ContextPlugin.install(ContextManager)`.
- Загрузка дерева контекстов, вызывается метод всех контекстов `Context.start()`. Все дочерние контексты любого зависящего контекста контейнера загружаются в этот же момент несколькими потоками.
- Загрузка плагинов[207], вызывается метод всех плагинов контекста `ContextPlugin.launch()`. Он включает в себя загрузку встроенного web сервера при условии, что он реализован как плагин контекста.
- Инициализация соединения. Большинтсво сокетов сервера, которые принимают входящие соединения, открыты на этой стадии.

- Создание встроенных ресурсов[208]. Эта стадия происходит только при первой загрузке сервера.

15.2.10.2 Выключение сервера

Выключение сервера - это полностью противоположный процесс включению сервера. Далее указаны основные шаги:

- Выключение соединения. Большинство слушающих сервер сокетов закрываются на этом этапе.
- Выключение плагинов. Вызывается метод всех плагинов контекста `ContextPlugin.shutdown()`. Он включает в себя выключение встроенного веб-сервера.
- Деинсталляция плагинов контекста, вызывается метод `ContextPlugin.deinstall(ContextManager)`.
- Выключение дерева контекстов, вызывается метод всех контекстов `Context.stop()`.
- Деинициализация плагинов, вызывается метод всех плагинов контекста `ContextPlugin.deinitialize()`.
- Выключение соединения отказоустойчивых кластеров.
- Выключение хранения данных сервера.

15.2.11 Работа в распределенной архитектуре

Эти разделы описывают особенности разработки, относящейся к распределенной архитектуре[1332]. Далее представлен список основных факторов, которые необходимо учитывать при создании совместимого с распределенной архитуктурой модуля:

- У каждого контекста есть три пути: локальный путь, соседний путь и удаленный путь. Их различия указаны ниже.
- Каждый раз, когда вы получаете путь контекста в качестве строки, вы должны тщательно выбирать между использованием методов `getPath()` и `getRemotePath()`, зависящих от того, как используется путь строки. В большинстве случаев, но не всегда, `getPath()` будет верным выбором.
- Есть разница между методами `get()` от `Context` и `ContextManager`. Второй имеет доступ к локальному пути, в то время как первый имеет доступ к удаленному пути, тем самым методы возвращают разные результаты.
- Разработка кода на стороне сервера и получение `Context` могут привести к созданию *контекста прокси,* который соединяется с удаленным соседним устройством в распределенной установке. Поэтому даже самые простые вызовы метода могут быть значительно отложены или даже остановлены из-за сетевых проблем.

Контексты в распределенной архитектуре

У интерфейса `Context` есть количество методов, которые связаны с функционированием в распределенной инсталляции. Представленный далее список описывает некоторые из методов:

- Метод `isDistributed()` определяет, является ли на самом деле экземпляр класса, включающего интерфейс `Context`, контекстом прокси, у которого есть удаленное одноранговое устройство в распределенной архитектуре.
- Метод `getPath()` всегда возвращает путь контекста в локальное дерево контекстов.
- `getPeerPath()` возвращает путь контекста на сервер сразу же, соединяясь с текущим сервером (текущим одноранговым устройством сервера). Очень редко есть необходимость использовать этот метод в пользовательском коде.
- `getRemotePath()` возвращает путь контекста на сервер, где он был определен. Этот вывод отличается от вывода `getPeerPath()` только в случае трех и более уровней серверов распределенной архитектуры. В нераспределенной архитектуре выводы этих двух методов равняются `getPath()`.
- Методы `getRemoteRoot()` и `getPeerRoot()` возвращают информацию о локальных корневых путях контекстов ветвей, которые соединены в распределенной архитектуре. Пока *javadocs* предоставляет необходимое количество деталей, нет необходимости вызывать эти методы напрямую.

15.2.12 Дополнительные темы

Этот раздел описывает функции, применяемые опытными разработчиками AtomMind.

15.2.12.1 Работа с привязками

Привязки 735 позволяют устанавливать и управлять связями между разными частями единой модели данных 41 сервера, например, ячейками в таблице БД или переменными/функциями/событиями разных контекстов.

Несмотря на то, что привязки обычно не требуют кастомизации, есть возможность реализовать и использовать ваш собственный пользовательский движок с того момента, как соответсвующие классы стали частью API с открытым исходным кодом.

Движок привязки базируется на нескольких основных классах и интерфейсах:

- `BindingProcessor` управляет выполнением привязок и соответствующих потоков.
- `BindingProvider` определяющий действия для выполнения привязок и использования их результата.
- `Binding` определяет саму привязку.

Реализация процессора привязок

Процессор привязок не требует кастомизации. Стандартный класс `DefaultBindingProcessor` может быть использован в большинстве сучаев.

Реализация поставщика привязок

В отличие от процессора, у поставщика привязок должны быть пользовательские реализации в каждом движке привязок.

Следующий минимальный набор методов должен быть переписан для кастомизации:

- `createBindings()`, возвращающий `Map<Binding, EvaluationOptions>` с объектами привязок и их опциями вычисления
- `writeReference(int method, Binding binding, Reference cause, Object value, ChangeCache cache)`, которые выполняют изменение цели `binding` в предоставленное `value`

`Method` должен быть одним из:

- `EvaluationOptions.STARTUP`
- `EvaluationOptions.EVENT`
- `EvaluationOptions.PERIODIC`

Если пользовательская реализация не основывается на стандартном `ContextBindingProvider`, также будет важно реализовать методы, которые добавляют и удаляют слушателей ссылки, которые вызываются при изменении состояния выполнения определенной привязки. Методы следующие:

- `addReferenceListener(Reference ref, ReferenceListener<T> listener)`
- `removeReferenceListener(ReferenceListener<T> listener)`

Пользовательский поставщик привязок должен информировать слушателей об определенном изменении, которое должно вызывать выполнение привязки с помощью события. Это происходит путем вызова метода `referenceChanged(final Reference cause, final Map<String, Object> environment, ChangeCache cache, boolean asynchronousProcessing)`, ранее добавленного `ReferenceListener`.

Использование движка привязки

Использование пользовательского движка привязки является несложным:

- Первым делом должен быть создан `BindingProcessor`. Он обычно принимает экземпляр пользовательского `BindingProvider` как аргумент. Также пользовательский пул потоков используется для параллельно работающего процесса привязки, который может быть представлен аргументом конструктора.
- Дальше процесс привязки должен быть инициирован путем вызова метода `start()`.

- Как только движок прерывается, метод процессора привязки `stop()` должен быть применен.

15.2.12.2 Взаимодействие по протоколу AtomMind

В этой главе подробно описывается, как работают реализации протокола AtomMind на стороне сервера и клиента. Обратите внимание, что сервер и клиент - это логические роли, в связи с чем могут быть различные сценарии:

- В AtomMind Client или AtomMind Server API-подключение к AtomMind Server, AtomMind Server играет роль сервера, а AtomMind Client / API играют роль клиента
- В агенте подключение к AtomMind Server, агент играет роль сервера, а AtomMind Server выступает как клиент
- в распределенной архитектуре поставщик соединения выступает в роли сервера, в потребитель - в роли клиента

Реализация серверной стороны

Основной класс сервера подключений по протоколу AtomMind - какой-либо класс, наследованный от `DefaultClientController`, например, `AgentImplementationController`. Обычно этот класс создается из контролирующего потока, запущенного при принятии нового TCP подключения, т.е. данный поток имеет доступ к сокету подключения (инкапсулированный в объект `BlockingChannel`).

Как только создан клиентский контроллер, контролирующий поток должен начать периодически вызывать метод контроллера `run()`. Этот метод выполняет основную логику обработки данных:

- Сначала, он вызывает `AggreGateCommandParser.readCommand()` для чтения полностью определенного протокольного блока AtomMind из сокета. Вызов анализатора может блокироваться до получения достаточных данных, либо если в системе не хватает памяти для обработки поступающих данных.
- Если анализатор возвращает завершенную команду `IncomingAggreGateCommand`, контроллер ставит эту команду в очередь в `CommandQueueManager` и предъявляет задачу на обработку команды своему `ExecutorService`. Задача будет обрабатываться в отдельном пуле потоков, который был предоставлен в качестве аргумента конструктору клиентского контроллера.

Каждая задача обработки команды делает следующее:

- Выполняет запрошенную операцию (**Get/Set Variable, Call Function** или **Subscribe/Unsubscribe to Events**) и формирует ответ `OutgoingAggreGateCommand`
- Обрабатывает очередь ожидающих событий (если какие-то события, на которые подписан текущий клиент, были добавлены в очередь после последней обработки команд) и отправляет каждое событие клиенту как отдельную `OutgoingAggreGateCommand`
- Отправляет ответ оригинальной команде

Реализация клиентской стороны

Основной класс клиента подключений по протоколу AtomMind - специфичная реализация `AbstractAggreGateDeviceController`, такая как `RemoteServerController`. Контроллер отвечает за:

- установку TCP соединения с сервером, разбор входящих команд через `CommandParser` и отправку ответов
- реализацию логики авторизации (входа)
- поддержание прокси контекстного дерева контекстов удаленного сервера (представленных экземплярами `ProxyContext`)
- работу в качеству потока `AsyncCommandProcessor`, который читает и обрабатывает входящие данные

Обработчик асинхронных команд работает все время, пока есть соединение с сервером. Поток выполняет следующий цикл:

- Использование `AggreGateCommandParser.readCommand()` для чтения полностью определенного протокольного блока AtomMind из сокета
- Если команда асинхронная, (т.е. получен экземпляр события), задача на ее обработку немедленно добавляется к пулу потоков обработки отдельных событий (обычно это пул из одного потока)
- Если команда синхронная, обработчик команд проверяет свою очередь ранее отправленных команд, чтобы найти команду с таким же ID, как и полученный ответ от сервера
- Как только такая команда найдена, ее `ReplyMonitor` получает оповещение, которое разблокирует поток, инициировавший какую-либо синхронную операцию сервера (**Get/Set Variable, Call Function** или **Subscribe/Unsubscribe to Events**)

Прокси контексты контекстного дерева контроллера сервера обеспечивает перенаправление всех основных вызовов на сервер и блокировку до получения ответа с использованием описанного механизма.

15.2.12.3 Кэширование формата

В любой инсталляции множество таблиц данных[49] передаются между серверами AtomMind, клиентами и агентами. Каждая таблица имеет описание формата[50], которое может включать значительное количество данных, таких как подробные описания (тексты подсказки) полей, строковые и табличные значения по умолчанию, валидаторы и их сообщения об ошибках, и пр. Чаще всего, если таблица из 1-10 строк кодируется в строку[2123] для передачи по сети, ее формат занимает больше места, чем сами данные.

В то же время, есть много таблиц с одинаковым форматом. Например, может быть тысяча пользователей[478], с разными свойствами,[481] но все таблицы этих свойств имеют один формат.

При отправке таблиц одного формата на удаленный сервер (например, когда клиент открывает свойства множества пользователей для редактирования) важно избегать многократной отправки общего формата таблиц. Для этого существует технология AtomMind протокола[2119] *кэширование формата*.

Кэширование формата работает по следующим правилам:

- Когда таблица данных с определенным форматом впервые кодируется[2123] для передачи по сети, ее формат сохраняется в центральном кэше формата AtomMind Server, реализованном экземпляром объекта `FormatCache` и доступном через статический метод `Server.getFormatCache()`. Формат получает уникальный ID сразу после кэширования.
- Если таблица с упомянутым ID впервые отправляется через соединение по AtomMind протоколу[2119], отправляется и сам формат, и его ID. Клиент подключения сохраняет формат и его ID в локальном кэше соединения, доступном через `AbstractAggreGateDeviceController.getFormatCache()`.
- При повторной отправке таблицы с таким же форматом через это же соединение, кодируется и отправляется только ID формата (сам формат не отправляется). Клиент соединения берет формат таблицы из своего локального кэша сразу при получении такой таблицы.

15.2.12.4 Выполнение действий программно

В редких случаях может понадобиться выполнить действия[87] сервера программно. Это позволяет заменить реализации UI процедур по умолчанию кастомизированными процедурами и/или автоответами.

Чтобы выполнить действие программно:

- Получите объект `Context` действия из `ContextManager`
- Вызовите `ActionsUtils.initAction()` статический метод, предоставляющий имя `Context` и действия. `ActionExecutionMode` должна быть выставлена либо на `NORMAL`, либо на `HEADLESS`, если UI процедуры фактически не выполняются человеком-оператором. `initAction()` вернет `ActionIdentifier`, которое позднее будет использоваться для степпинга действия.
- Вызовите `ActionsUtils.stepAction()` в цикле, пока не вернет null. Каждая итерация `stepAction()` будет возвращать определенную UI процедуру для ручного выполнения как `GenericActionCommand`. Каждый вызов, начиная со второго, должен включать ваш сформированный вручную запрос `GenericActionResponse` к ранее полученному `GenericActionCommand`.
- Обработайте каждую `GenericActionCommand` согласно ее **типу**, **параметрам** и **заголовку**.

Каждое событие запускается как отдельный поток AtomMind Server. Если по какой-то причине, вы прекратите вызывать `stepAction()` до того, как вернется null, поток действия на стороне сервера зациклится до прекращения соединения (или навсегда, если действие выполнено вручную изнутри плагина[207] сервера). Действие сервера невозможно остановить, пока оно не дойдет до логического конца.

15.3 API для .NET

Интерфейс Программирования Приложений (API) AtomMind Server для .NET и .NET Compact (.NET API <%ls%) - это удобная библиотека для доступа AtomMind Server из любого приложения, базирующегося на .NET, которая использует Веб-сервис[1417] или Протокол связи AtomMind[2119].

Интерфейс Программирования Приложений (API) -- это интерфейс, который программа или библиотека предоставляют для других компьютерных программ, чтобы они могли выполнять запросы к ней.

API .NET Compact недоступен для прямой загрузки на сайт AtomMind. Пожалуйста, свяжитесь с ТВЭЛ для получения пакета API .NET Compact.

Возможности API для .NET

Используя API, можно:

- иметь доступ к ресурсам сервера и подсоединенных устройств;
- изменять конфигурацию сервера и устройства;
- выполнять операции сервера и устройства;
- получать события сервера и устройства;
- подключаться к AtomMind Server в "режиме устройства", т.е. писать код на .NET, который **эмулирует или реализует устройство аппаратного оборудования, подключенное к AtomMind**.

Технически API для .NET предоставляет следующую функциональность:

- полный доступ к контекстам 41 сервера через так называемое *контекстное proxy-дерево;*
- получение и установка значений переменных 61 контекста 41;
- вызов функций 70 контекста;
- прослушивание событий 73 контекста (только при использовании Протокола AtomMind);
- создание Таблиц данных 49 и работа с ними;
- эмулирование Агента.

Дистрибутивный пакет API для .NET

AtomMind API для .NET доступен как архивный файл в формате ZIP, который содержит:

- исходный код
- примеры
- решение под Microsoft Visual Studio

Основные классы и интерфейсы API для .NET

Имя класса	Описание
`RemoteLinkServer`	Параметры контейнера серверного соединения (адрес, порт, имя пользователя и пароль). Экземпляр этого класса передается в конструктор `RemoteLinkServerController`.
`RemoteLinkServerController`	Этот класс используется для установки и контроля за соединением с AtomMind Server. Он предоставляет доступ к интерфейсу `ContextManager`, который, свою очередь, предоставляет доступ к контекстному дереву сервера.
`ContextManager`	Интерфейс, который предоставляет доступ к контекстному дереву. Экземпляр класса, реализующего этот интерфес, может быть получен из `RemoteLinkServerController` посредством метода `getContextManager()`
`Context`	Интерфейс, позволяющий работать и одним контекстом. Экземпляры класса, реализующие этот интерфейс, возвращаются различными методами `ContextManager`.
`DataTable`	Реализация Таблицы Данных 49. Предоставляет доступ к формату и записям таблицы. Экземпляры этого класса возвращаются следующими методами класса `Context: getVariable()` и `callFunction()`.
`DataRecord`	Реализация одной записи Таблицы Данных.
`TableFormat`	Класс, реализующий формат 49 Таблицы Данных. Он включает свойства таблицы и список полей.
`FieldFormat`	Класс, реализующий формат 49 одного поля Таблицы Данных. Определяет имя, тип и другие параметры поля, включая *валидаторы*, *значения выбора* и пр.

Доступ к AtomMind Server из приложения .NET через Веб-сервис

Чтобы подключиться к AtomMind Server из приложения .NET через Веб-сервис, в Вашем приложении следует создать объект `LinkServerClient`, вызвав его конструктор с адресом веб-сервиса и именем пользователя/паролем AtomMind Server.

Ниже следует объявление конструктора LinkServerClient:

```
public LinkServerClient(String username, String password, String url)
```

Пример: создание клиентского объекта AtomMind Server на C#:

```
String webServiceAddress = "https://localhost:8443/ws/services/LinkServerWeb";
String userName = "admin";
String userPassword = "admin";
LinkServerClient lsc = new LinkServerClient(userName, userPassword, webServiceAddress);
```

Чтобы получить доступ к Веб-сервису по HTTPS, необходимо сначала установить сертификат. Пример (на C#):

```
System.Net.ServicePointManager.CertificatePolicy = new TrustAllCertificatePolicy();

public class TrustAllCertificatePolicy : System.Net.ICertificatePolicy
{
  public TrustAllCertificatePolicy()
  { }
  public bool CheckValidationResult(ServicePoint sp, X509Certificate cert, WebRequest r
  {
     return true;
  }
}
```

Объект `LinkServerClient` реализует все методы, определенные Веб-сервисом AtomMind Server:

- `getXML`
- `setXML`
- `callXML`
- `setByStringArray`
- `callByStringArray`
- `get`
- `set`
- `call`

Методы **getXML, setXML, callXML, setByStringArray** и **callByStringArray** всесторонне описаны в главе Веб-сервисы. В большинстве случае удобнее использовать методы **get**, **set** и **call**, которые представляют Таблицы Данных как объекты типа `DataTable` , определеные в области имен `AtomMind`. Ниже представлены объявления всех методов, предоставляемых LinkServerClient:

МЕТОДЫ, ИСПОЛЬЗУЮЩИЕ ДЛЯ КОДИРОВАНИЯ ТАБЛИЦ ДАННЫХ XML

```
public String getXML(String context, String variable)
public void setXML(String context, String variable, String value)
public String callXML(String context, String variable, String parameters)
```

МЕТОДЫ, ИСПОЛЬЗУЮЩИЕ ДЛЯ КОДИРОВАНИЯ ТАБЛИЦ ДАННЫХ СТРОКОВЫЙ МАССИВ

```
public void setByStringArray(String context, String variable, String[] values)
public DataTable callByStringArray(String context, String variable, String[] values)
```

МЕТОДЫ, КОТОРЫЕ ОБРАБАТЫВАЮТ ТАБЛИЦЫ ДАННЫХ КАК ОБЪЕКТЫ

```
public DataTable get(String context, String variable)
public void set(String context, String variable, DataTable value)
public DataTable call(String context, String variable, DataTable parameters)
```

15.4 API для C/C++

Интерфейс Программирования Приложений (API) AtomMind Server для C/C++ (C/C++ API AtomMind Server) – это удобная библиотека для доступа AtomMind Server из любого приложения, базирующегося на C/C++, которая использует Протокол связи AtomMind 2119.

Возможности API для C/C++

Используя API, можно:

- иметь доступ к ресурсам сервера и подсоединенных устройств;
- изменять конфигурацию сервера и устройства;
- выполнять операции сервера и устройства;
- получать события сервера и устройства;
- подключаться к AtomMind Server в "режиме устройства", т.е. писать код на C/C++, который **эмулирует или реализует устройство аппаратного оборудования, подключенное к AtomMind**.

Технически API для C/C++ предоставляет следующую функциональность:

- полный доступ к контекстам 41 сервера через так называемое *контекстное proxy-дерево;*
- получение и установка значений переменных 61 контекста 41;
- вызов функций 70 контекста;
- прослушивание событий 73 контекста;
- эмулирование Агента.

15.5 API для JavaScript/TypeScript

Интерфейс Программирования Приложений (API) AtomMind Server для JavaScript/TypeScript (AtomMind Server JS/TS API) - то удобная библиотека для доступа к AtomMind Server из любого приложения на базе JavaScript или TypeScript, которая использует Протокол связи AtomMind 2119.

Источники API можно скачать с GitHub. Демо-приложение, показывающее, как взаимодействовать с AtomMind Server из приложения на базе React, также доступно на GitHub.

Возможности API для JavaScript/TypeScript

Используя API, можно:

- иметь доступ к ресурсам сервера и подсоединенных устройств;
- изменять конфигурацию сервера и устройства;
- выполнять операции сервера и устройства;
- получать события сервера и устройства.

Технически API для JavaScript/TypeScript API предоставляет следующий функционал:

- полный доступ к контекстам 41 сервера через так называемое *контекстное proxy-дерево;*
- получение и установка значений переменных 61 контекста 41;

- вызов функций 70 контекста;
- прослушивание событий 73 контекста;
- создание таблиц данных 49 и работа с ними.

15.6 REST API

AtomMind Server поддерживает REST API для интеграции со сторонними системами. Вы можете использовать запросы REST API для работы с переменными контекста 41, функциями 70 и событиями 73, а также чтобы вычислять выражения 112.

В терминах программирования, REST (Representational State Transfer) является архитектурным стилем ПО во всемирной сети Интернет.

ВЕРСИИ REST API

Текущая версия REST API - **v1**. Предыдущая версия REST API на базе XML может быть устаревшей для будущих версий AtomMind. Если вам нужна документация для старой версии REST API, пожалуйста, напишите на info@tvel.ru.

ФОРМАТ СООБЩЕНИЯ

REST API использует JSON для представления данных. Передаваемые в запросах и ответах данные имеют формат JSON.

ПРОТОКОЛЫ HTTP

REST API может использовать протоколы HTTP и HTTPS.

Рекомендуется использовать протокол HTTPS, так как аутентификация 1403 REST API происходит на основе токена. Любой, кто отслеживает трафик HTTP, сможет узнать токен и получить те же права доступа, что и владелец токена.

МЕТОДЫ HTTP

REST API поддерживает **GET**, **POST**, **PUT** и **PATCH** HTTP-методы. Данные методы поддерживают операции **retrieve** и **update** на переменных контекста, функциях и выражениях. Операции **create** и **delete** нельзя выполнить при помощи REST API.

В программировании, create, read, update, и delete (CRUD) - четыре базовые функции постоянного хранения.

КОДЫ СОСТОЯНИЙ HTTP

REST API использует стандартные коды состояний HTTP для обозначения результата запроса.

- **1XX** — Информационные ответы
- **2XX** — Успешно
- **3XX** — Перенаправление
- **4XX** — Ошибки клиента
- **5XX** — Ошибки сервера

15.6.1 Настройка REST API

Для настройки REST API на уровне сети, можно использовать следующие опции настройки веб-сервера[354]:

- Список алиасов имени хоста сервера, разделенных запятыми[355]
- Номер порта для приема защищенных соединений (HTTPS)[355]
- Номер порта для приема незащищенных соединений (HTTP)[355]
- Включить незащищенный доступ к веб-приложениям[354]

15.6.2 Доступ к REST API

REST API доступен по адресам, указанным в настройках AtomMind Server для REST API[1402].

REST API имеет следующие параметры:

- **Время жизни токена** определяет, сколько времени токен остается действительным.
- **Секретный ключ**, чье название говорит само за себя. Используется для подписи токенов доступа.
- **Сгенерировать секретный ключ** - чекбокс, который активирует генерацию нового Ключа сервера.
- **Разрешенные значения Origin (для проверок CORS)** - разделённый запятыми список источников (хостов, портов и схем), с которых разрешены запросы к серверу AtomMind Server. Специальное значение `*` означает доступность из любых источников. Выставленное здесь значение будет возвращаться сервером в заголовке `Access-Control-Allow-Origin` при ответах на CORS-запросы. При отсутствии значения запросы из любых сторонних источников будут запрещены.

Токены REST API становятся недействительными после перезагрузки сервера. Чтобы токены оставались действительными после перезагрузки, отключите параметр **Сгенерировать секретный ключ** и убедитесь, что заполнено поле **Секретный ключ**. С такими настройками прежние токены останутся действительными после перезагрузки, но чтобы они работали, необходимо снова залогиниться на сервере под тем же именем пользователя, для которого были получены прежние токены.

REST API работает через HTTP или HTTPS соединение и использует авторизацию. По умолчанию, можно получить доступ к REST API по следующим адресам и портам локального хоста:

- `https://localhost:8443/rest/{request}` для HTTPS соединений
- `http://localhost:8080/rest/{request}` для HTTP соединений

В примерах выше, замените `{request}` запросом, как описано в Справочнике REST API[1405].

Чтобы получить результат такого запроса, необходимо авторизоваться[1403].

Пример:

Если AtomMind Server работает на локальном хосте, он имеет виртуальное устройство, и все показатели настроек выставлены по умолчанию, вы можете вызвать REST API, использую следующий URL:

`https://localhost:8443/rest/v1/contexts/users.admin.devices.virtual`

Этот запрос получает информацию о контексте `users.admin.devices.virtual`, который является контекстом виртуального устройства.

15.6.3 Аутентификация

REST API использует аутентификацию на основе токенов.

Токен доступа имеет такое свойство, что любой владелец токена ("предъявитель") может использовать токен любым способом, каким его используют другие владельцы. Для использования токена на предъявителя не требуется, чтобы предъявитель подтверждал правильность криптографического ключа (proof-of-possession).

ПРОЦЕСС АУТЕНТИФИКАЦИИ

Аутентификация состоит из двух шагов. Сначала вы делаете запрос на аутентификацию, используя ваши идентификационные данные. Ответом на этот вопрос является токен. Далее вы используете этот токен для отправки всех остальных запросов.

1. Сделайте запрос auth 1405. Ответ вернут токен в поле **токен**.
2. Включайте полученный токен в заголовки запросов. Для этого используйте заголовок **Авторизация**. Его значение должно начинаться со строки `"Bearer "`, а далее - полученный токен.

Пример:

```
Authorization: Bearer
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiAZJhZG1pbiIsImV4cCI6MTU1NjE4MTI3NywiaWF0IjoxNTU2MT
c3Njc3fQ.LkGOWmgiRn3O1vREbRzDCigCSMeGbI337fsioTgK1p-
kMIYx1R4d_lcUOQYN3gpFBazvvmR3mO2j4fb5J3YFug
```

ОБНОВЛЕНИЕ ТОКЕНОВ

Если срок действия вашего токена закончился, вы можете получить новый через запрос refresh 1406. Укажите ваш старый токен, в ответ придет новый.

15.6.4 STOMP API

AtomMind Server поддерживает STOMP API для интеграции со сторонними системами ПО. Вы можете использовать соединения по протоколу STOMP, чтобы подписаться и получать уведомления при изменении переменных 61 или возникновении событий 73 в интересующем контексте.

Simple (or Streaming) Text Orientated Messaging Protocol (STOMP) - это текстовый протокол, схожий с HTTP. STOMP обеспечивает интероперабельный формат проводной связи, чтобы клиенты STOMP могли общаться с любым брокером сообщений STOMP. Это обеспечивает простое и обширное взаимодействие по передаче сообщений между многими языками, платформами и брокерами.

настройка STOMP API

Чтобы установить соединение, STOMP-запросы используют те же URL и номер порта, что и REST API 1401.

Аутентификация осуществляется через REST API-запрос 1403.

доступ к STOMP API

Чтобы работать с STOMP API в качестве клиента, вам необходимо получить токен аутентификации, создать экземпляр клиента, а затем установить соединение между этим клиентом и сервером, использующим токен. После установления соединения между STOMP-клиентом и STOMP-сервером будут выполняться все поддерживаемые операции.

Получить токен можно с помощью запроса аутентификации 1405. После этого подключите ваш клиент к серверу через передачу запроса **Connect** с использованием токена.

ЗАПРОС CONNECT

Устанавливает соединение между экземпляром STOMP-клиента и STOMP-сервером. Если во время выполнения запроса не происходит ошибок, соединение в итоге будет установлено, и вы получите представление STOMP-сессии. Передавайте все остальные STOMP-запросы через эту сессию.

Токен аутентификации нужно включить в заголовки запроса. Для получения токена выполните запрос аутентификации 1405.

Запрос

```
/rest/v1/stomp
```

Файлы заголовка запроса

```
Authorization: {authentication_token}
```

Замените `{authentication_token}` на токен, полученный в результате запроса аутентификации.

Пример запроса

Запрос с использованием локальной машины в качества хоста STOMP-сервера и номером HTTP порта по умолчанию:

```
ws://localhost:8080/rest/v1/stomp
```

Запрос с использованием локальной машины в качества хоста STOMP-сервера и номером HTTPS порта по умолчанию:

```
ws://localhost:8443/rest/v1/stomp
```

Заголовок:

```
Authorization: Bearer
eyJhbGciOiJIUzI1NiJ9.eyJqdGkiOiJjOTQ1NTk0ZC01NGVmLTQ0ODItYWQxZi00NThhYWQwNzZlZDciLCJzd
WIiOiJhZG1pbiIsImlhdCI6MTU5MDQ3OTQ2MywiZXhwIjoxNTkwNDgzMDYzfQ.AeaaXg7_m651JGquQ-
WEzh_8Z7avL14s709dSd8wV4c
```

ЗАПРОС ПОДПИСКИ НА ИЗМЕНЕНИЯ ЗНАЧЕНИЙ

Подписывается на оповещения об изменениях значения переменной. В качестве результата вы получите представление подписки. Чтобы отписаться от таких оповещений, вызовите соответствующий метод экземпляра подписки.

Запрос

```
/contexts/{context}/variables/{variable}
```

Замените `{context}` на полный путь контекста.

Замените `{variable}` на имя переменной.

Параметры

В качестве параметра вы должны передать обработчик уведомлений. Каждый раз при изменении значения переменной обработчик будет получать новые значения в качестве входных данных.

Пример

Подписка на изменения переменной info контекста users.admin.devices.virtual:

```
/v1/contexts/users.admin.devices.virtual/variables/info
```

Пример строкового представления входных данных при изменении значения переменной:

```
[{"updateOriginator":0,"variable":"info","value":
[{"localRoot":"","remotePath":"users.admin.devices.device1","icon":"st_device","remote
Root":null,"mapped":false,"description":"Virtual Device
2","peerRoot":null,"type":"device.virtual","peerPrimaryRoot":null,"group":"default"}],
"user":null}]
```

ЗАПРОС ПОДПИСКИ НА СОБЫТИЕ

Подписывается на оповещения о событии. В качестве результата вы получите представление подписки. Чтобы отписаться от таких оповещений, вызовите соответствующий метод экземпляра подписки.

Запрос

```
/v1/contexts/{context}/events/{event}
```

Замените `{context}` на полный путь контекста.

Замените `{event}` на имя события.

Параметры

В качестве параметра вы должны передать обработчик уведомлений. Каждый раз при возникновении события обработчик будет получать строковое представление события в качестве входных данных.

Пример

Подписка на событие 1 контекста users.admin.devices.virtual:

```
/v1/contexts/users.admin.devices.virtual/events/event1
```

Пример строкового представления входных данных при возникновении события:

```
[{"date":"2000-02-01 12:00:00.000","string":"","int":0}]
```

15.6.5 Справочник REST API

В этом разделе описываются все REST API-запросы AtomMind Server.

15.6.5.1 Auth

Отправляет имя пользователя и пароль на сервер. Ответ содержит токен. Используйте этот токен в других запросах на сервер.

Более подробно об аутентификации см. раздел Аутентификация 1403.

ЗАПРОС

`POST /auth`

Запрос аутентификации должен иметь HTTP-заголовок с типом содержимого: `Content-Type:application/json`

ПОЛЯ ТЕЛА ЗАПРОСА

- *username* — Имя пользователя.
- *password* — Пароль.

ПОЛЯ ТЕЛА ОТВЕТА

- *token* — Токен аутентификации.

ПРИМЕР ЗАПРОСА

Запрос:

```
POST https://localhost:8443/rest/auth
```

Пример тела запроса JSON:

```
{
```

```
        "username":"admin",
        "password":"admin"
    }
```

Пример тела ответа JSON:

```
    {
        "token":
"eyJhbGciOiJEOfUxMiJ9.eyJzdWIiOiJhZG1pbiIsImV4cRT6MTU1NjAOMzA4OCwiaWFOIjoxNTU2MDM5NDg4
fQ.7S_OOFL1mjGVAafBWQ2lDsJguUqCng9_OTfCsiU9RckQTRsI3H86ZytJkyjKtzTF4VYQ_3A3ZSa7RPhAttu
5NQ"
    }
```

15.6.5.2 Refresh

Получает новый токен. Запрос не берет никаких параметров, кроме заголовка аутентификации. Ответ содержит новый токен.

ЗАПРОС

```
GET /refresh
```

REQUEST HEADER FIELDS

- *Authorization* — Токен аутентификации. Более подробно об аутентификации см. раздел Аутентификация[1403].

ПОЛЯ ТЕЛА ЗАПРОСА

- *token* — Новый токен аутентификации.

ПРИМЕР ЗАПРОСА

Запрос:

```
    GET https://localhost:8443/rest/refresh
```

Пример тела ответа JSON:

```
    {
        "token":
"eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbiIsImV4cCI6MTU1NjIxMTQ5NSwiaWFOIjoxNTU2MjA3ODk1
fQ.VYhyM5-K-AAmFwC2OVep1Zz8qWhAF1C4PgmnQhk1hyGzSs-Wj6X2gVjz8x83R-
zrXFJt1PxbOx19Bvo7RsyzFQ"
    }
```

15.6.5.3 Evaluate

Отправляет выражение для вычисления. Ответ содержит результат вычисления.

ЗАПРОС

```
POST /v1/evaluate
```

ПОЛЯ ЗАГОЛОВКА ЗАПРОСА

- *Authorization* — Токен аутентификации. Более подробно об аутентификации см. раздел Аутентификация[1403].

ПОЛЯ ТЕЛА ЗАПРОСА

- *expression* — Выражение для вычисления.
- *defaultTable* — Таблица по умолчанию.
- defaultContext — Контекст по умолчанию.

ПОЛЯ ТЕЛА ОТВЕТА

- *result* — Результат вычисления.

ПРИМЕР ЗАПРОСА

Запрос:

```
POST https://localhost:8443/rest/v1/evaluate
```

Пример тела запроса JSON:

```
{
    "expression": "10+10",
    "defaultTable":null,
    "defaultContext":""
}
```

Пример тела ответа:

```
{
"result": 20
}
```

15.6.5.4 Context

Получает информацию о контексте. Ответ содержит информацию о самом контексте и всех его дочерних контекстах.

ЗАПРОС

```
GET /v1/contexts/{contextPath}
```

Замените `{contextPath}` на полный путь контекста.

Чтобы получить все контексты, используйте путь контекста `server`.

ПОЛЯ ЗАГОЛОВКА ЗАПРОСА

- *Authorization* — Токен аутентификации. Более подробно об аутентификации см. раздел Аутентификация 1403.

ПОЛЯ ТЕЛА ОТВЕТА

- *name* — Имя контекста.
- *description* — Описание контекста.
- *path* — Путь контекста.
- *parent* — Родительский контекст данного контекста.
- *children* — Дочерние контексты данного контекста. Это поле может содержать описания вложенных контекстов.

ПРИМЕР ЗАПРОСА

Запрос:

```
GET https://localhost:8443/rest/v1/contexts/users.admin.devices.virtual
```

Пример тела ответа JSON:

```
{
    "name": "virtual",
    "description": "virtual",
    "path": "users.admin.devices.virtual",
    "parent": "users.admin.devices",
    "children": []
```

```
}
```

15.6.5.5 Variables

Получает информацию обо всех переменных контекста. Ответ содержит информацию о переменных 61. Можно использовать дополнительный параметр, чтобы включить в ответ информацию о формате таблицы.

ЗАПРОС

```
GET /v1/contexts/{contextPath}/variables
```

Замените `{contextPath}` на полный путь контекста.

ПОЛЯ ЗАГОЛОВКА ЗАПРОСА

- *Authorization* — Токен аутентификации. Более подробно об аутентификации см. раздел Аутентификация 1403.

ПАРАМЕТРЫ ЗАПРОСА

- *includeFormat* — Необязательный параметр. Ответ включает информацию о формате таблицы 50. Эта информация предоставляется в поле *tableFormat*. Возможные значения: `true`, `false`.

ТЕЛО ОТВЕТА

Список переменных контекста. Каждая запись содержит информацию об одной переменной.

ПОЛЯ ТЕЛА ОТВЕТА

- *name* — Имя переменной.
- *group* — Группа переменной.
- *tableFormat* — Информация о формате таблицы.
- *defaultValue* — Значение по умолчанию.
- *readable* — Если переменная читается. Возможные значения: `true`, `false`.
- *writable* — Если переменная записывается. Возможные значения: `true`, `false`.

ПРИМЕР ЗАПРОСА

Запрос:

```
    GET https://localhost:8443/rest/v1/contexts/users.admin.models.example/variables?
includeFormat=false
```

Пример тела ответа JSON:

```
    [
        {
            "name": "granulator",
            "group": "default",
            "tableFormat": null,
            "defaultValue": [],
            "readable": true,
            "writable": true
        },
        {
            "name": "exampleTable",
            "group": "custom",
            "tableFormat": null,
            "defaultValue": [],
            "readable": true,
```

```
        "writable": true
    }
]
```

15.6.5.6 Variable (GET)

Получает информацию о переменной контекста. Ответ содержит информацию об этой переменной. Можно использовать дополнительные параметры, чтобы ограничить количество полей в выходных данных и задать смещение для полей.

ЗАПРОС

```
GET /v1/contexts/{contextPath}/variables/{variable}
```

Замените `{contextPath}` на полный путь контекста.

Замените `{variable}` на имя переменной.

ПОЛЯ ЗАГОЛОВКА ЗАПРОСА

- *Authorization* — Токен аутентификации. Более подробно об аутентификации см. раздел Аутентификация 1403.

ПАРАМЕТРЫ ЗАПРОСА

- *limit* — Необязательный параметр. Максимальное число возвращаемых записей.
- *offset* — Начало смещения для возвращаемых записей. Необязательный параметр.

ТЕЛО ОТВЕТА

Список записей переменной.

ПРИМЕР ЗАПРОСА

Запрос:

```
    GET
https://localhost:8443/rest/v1/contexts/users.admin.models.example/variables/exampleTa
ble?limit=2&offset=2
```

Пример тела запроса JSON:

```
    [
        {
            "name": "Room_3",
            "description": "Storage Area"
        },
        {
            "name": "Room_4",
            "description": "Boiler Room"
        }
    ]
```

15.6.5.7 Variable (PUT)

Заменяет переменную контекста. Ответ пустой.

ЗАПРОС

```
PUT /v1/contexts/{contextPath}/variables/{variable}
```

Замените `{contextPath}` на полный путь контекста.

Замените `{variable}` на имя переменной.

ПОЛЯ ЗАГОЛОВКА ЗАПРОСА

- *Authorization* — Токен аутентификации. Более подробно об аутентификации см. раздел Аутентификация 1403.

ТЕЛО ОТВЕТА

Новое значение переменной контекста.

ПРИМЕР ЗАПРОСА

Запрос:

```
    PUT
https://localhost:8443/rest/v1/contexts/users.admin.models.example/variables/exampleTa
ble
```

Пример тела запроса JSON:

```
    [
        {
            "name": "Room_1",
            "description": "Machinery Storage"
        },
        {
            "name": "Room_2",
            "description": "Workshop"
        }
    ]
```

15.6.5.8 Variable (PATCH)

Корректирует переменную контекста, заменяя определенную запись. Можно использовать дополнительный параметр, чтобы задать запись для корректировки. Ответ пустой.

ЗАПРОС

`PATCH /v1/contexts/{contextPath}/variables/{variable}`

Замените `{contextPath}` на полный путь контекста.

Замените `{variable}` на имя переменной.

ПОЛЯ ЗАГОЛОВКА ЗАПРОСА

- *Authorization* — Токен аутентификации. Более подробно об аутентификации см. раздел Аутентификация 1403.

ПАРАМЕТРЫ ЗАПРОСА

- *record* — Запись для корректировки. По умолчанию данный параметр `0`, и корректируется первая запись.

ТЕЛО ЗАПРОСА

Новое значение для записи.

ПРИМЕР ЗАПРОСА

Запрос:

```
    PATCH
https://localhost:8443/rest/v1/contexts/users.admin.models.example/variables/exampleTa
ble?record=2
```

Пример тела запроса JSON:

```
{
    "name": "Room_3",
    "description": "Oxygen Plant"
}
```

15.6.5.9 Events

Получает информацию о событиях 73 контекста. Ответ содержит список событий. Можно использовать дополнительный параметр, чтобы включить в ответ информацию о формате.

ЗАПРОС

```
GET /v1/contexts/{contextPath}/events
```

Замените `{contextPath}` на полный путь контекста.

ПОЛЯ ЗАГОЛОВКА ЗАПРОСА

- *Authorization* — Токен аутентификации. Более подробно об аутентификации см. раздел Аутентификация 1403.

ПАРАМЕТРЫ ЗАПРОСА

- *includeFormat* — Необязательный параметр. Ответ включает информацию о формате таблицы 50. Эта информация предоставляется в поле *tableFormat*. Возможные значения: `true`, `false`.

ТЕЛО ОТВЕТА

Список событий контекста. Каждая запись содержит информацию об одном событии.

ПОЛЯ ТЕЛА ОТВЕТА

- *name* — Имя события.
- *group* — Группа события.
- *level* — Уровень события.
- *tableFormat* — Информация о формате таблицы.

ПРИМЕР ЗАПРОСА

Запрос:

```
GET https://localhost:8443/rest/v1/contexts/users.admin.devices.virtual/events?includeFormat=false
```

Пример тела ответа JSON:

```
[
    {
        "name": "info",
        "group": "default",
        "level": 2,
        "tableFormat": null
    },
    {
        "name": "visibleInfoChanged",
        "group": null,
        "level": 0,
        "tableFormat": null
    }
```

```
]
```

15.6.5.10 Functions

Получает информацию о функциях 70 контекста. Ответ содержит список функций. Можно использовать дополнительный параметр, чтобы включить в ответ входящую и выходящую информацию.

ЗАПРОС

```
GET /v1/contexts/{contextPath}/functions
```

Замените `{contextPath}` на полный путь контекста.

ПОЛЯ ЗАГОЛОВКА ЗАПРОСА

- *Authorization* — Токен аутентификации. Более подробно об аутентификации см. раздел Аутентификация 1403.

ПАРАМЕТРЫ ЗАПРОСА

- *includeFormat* — Необязательный параметр. Ответ включает информацию о формате. Эта информация предоставляется в поле *tableFormat*. Возможные значения: `true`, `false`.

ТЕЛО ОТВЕТА

Список функций контекста. Каждая запись содержит информацию об одной функции.

ПОЛЯ ТЕЛА ОТВЕТА

- *name* — Имя функции.
- *group* — Группа функции.
- *concurrent* — Возможность одновременного выполнения нескольких вызовов этой функции.
- *inputFormat* — Входной формат функции.
- *outputFormat* — Выходной формат функции.

ПРИМЕР ЗАПРОСА

Запрос:

```
    GET https://localhost:8443/rest/v1/contexts/users.admin.devices.virtual/functions?
includeFormat=false
```

Пример тела ответа JSON:

```
    [
        {
            "name": "updateVariable",
            "group": "system",
            "concurrent": true,
            "inputFormat": null,
            "outputFormat": null
        },
        {
            "name": "initAction",
            "group": null,
            "concurrent": true,
            "inputFormat": null,
            "outputFormat": null
        }
```

```
]
```

15.6.5.11 Function

Вызывает функцию. Ответ содержит результат выполнения функции.

ЗАПРОС

```
POST /{contextPath}/functions/{function}
```

Замените `{contextPath}` на полный путь контекста.

Замените `{function}` на имя функции.

ПОЛЯ ЗАГОЛОВКА ЗАПРОСА

- *Authorization* — Токен аутентификации. Более подробно об аутентификации см. раздел Аутентификация 1403.

ТЕЛО ЗАПРОСА

Объект JSON с параметрами функции.

Если у функции пустой входной формат (т.е. не имеет полей), передайте пустой документ JSON (`{}`) в теле запроса.

ПОЛЯ ТЕЛА ОТВЕТА

- *result* — Результат функции.

ПРИМЕР ЗАПРОСА

Запрос:

```
POST
https://localhost:8443/rest/v1/contexts/users.admin.devices.virtual/functions/calculat
e
```

Пример тела запроса JSON:

```
[
    {
        "leftOperand": 10.0,
        "rightOperand": 20.0,
        "operation": "+"
    }
]
```

Пример тела ответа JSON:

```
[
    {
        "result": 30
    }
]
```

15.7 HTTP сервер

У AtomMind есть встроенный **Сервер HTTP** для взаимодействия с внешними системами с использованием HTTP.

Сервер HTTP - это сервер, который получает HTTP-запросы от клиентов (обычно веб браузеров) и дает им HTTP-ответы.

AtomMind Сервер HTTP позволяет получать HTTP-запросы , обрабатывать их и посылать ответы. Обработка запроса реализуется выражениями, в которых таблицы по умолчанию предоставляют доступ к параметрам запроса и ответа.

15.7.1 Настройка HTTP сервера

Общие настройки **HTTP сервера** доступны в глобальной конфигурации плагина 207 HTTP сервер. Чтобы получить доступ к этим настройкам, необходимо:

- Развернуть узел **Драйверы/Плагины** () в Системном дереве 370.
- Дважды щелкнуть мышью на значок плагина **HTTP сервера** ().

Настройки HTTP сервера

Таблица Настроек HTTP сервера содержит следующие поля:

Имя поля	Тип	Описание
URI	String	Строка идентифицирует выражение, которое должно обрабатывать соответствующий запрос.
Выражение	String	Выражение обрабатывает запрос.
Параметры	DataTable	Параметры запроса, доступные в выражении через его таблицу по умолчанию.
Предварительно авторизованный пользователь	String	Пользователь с правами доступа для вычисления выражения.

В таблице хранятся выражения для обработки запроса и соответствующие им URI. Порядок записей в таблице имеет значение. Если URI запроса соответствует нескольким выражениям, будет использоваться первое выражение.

URI

Этот параметр позволяет HTTP серверу находить выражение для обработки запроса. URI не должен включать в себя имя хоста, порт или строку запроса. Он не может быть пустой строкой или `/`. Если URI заканчивается символом `*`, то URI из запроса может заканчиваться любым суффиксом вместо `*`. Другие специальные символы не поддерживаются.

Пример: возможные варианты URI:

`/req` или `req` соответствуют запросу следующего типа `http://localhost:8080/req?<request>`

ВЫРАЖЕНИЕ

Это выражение 112 позволяет обрабатывать запрос. Параметры запроса и ответа представлены как поля таблицы данных по умолчанию 120. Полный список поддерживаемых параметров можно найти здесь 1415.

Пример: установка значения *"test"* для параметра тела ответа:

```
set(dt(), "responseBody", 0, "test")
```

Для более сложных случаев и пошаговой обработки запросов лучше использовать модели и их наборы правил:

```
{users.admin.models.model:request(dt())}
```

См. подробнее в примерах 1416.

ПАРАМЕТРЫ

Активные параметры[1415] будут доступны в выражении через таблицу данных по умолчанию[120]. Отключение неиспользуемых параметров может увеличить скорость обработки запроса. Все параметры активны по умолчанию.

ПРЕДВАРИТЕЛЬНО АВТОРИЗОВАННЫЙ ПОЛЬЗОВАТЕЛЬ

Пользователь[478], чьи права используются для вычисления выражение. Пожалуйста, будьте внимательны при выборе пользователя для этой настройки. Набор прав доступа пользователя и правильная обработка запроса помогут избежать нежелательного доступа к конфиденциальной информации. В некоторых случаях может быть полезным создать отдельного пользователя специально для обработки запросов.

15.7.2 Параметры HTTP сервера

Обработка запроса выполняется выражениями, в которых таблицы данных по умолчанию дают доступ к параметрам запросов и ответов. Эти параметры представлены полями таблицы данных по умолчанию.

ПАРАМЕТРЫ ЗАПРОСА

Следующие параметры запроса доступны в таблице данных по умолчанию:

Имя	Тип	Описание
authType	String	Имя соответствующей схемы.
characterEncoding	String	Имя кодировки символов, используемой в теле запроса.
contentLength	Integer	Длина тела запроса в байтах, предоставляемая входящим потоком, либо -1, если длина неизвестна.
contentType	String	MIME-тип тела запроса или *null*, если тип неизвестен.
contextPath	String	Часть URI запроса, которая указывает на контекст запроса.
headers	DataTable	Таблица данных заголовков запроса.
isAsyncStarted	Boolean	Проверяет, был ли запрос сделан в асинхронном режиме.
isAsyncSupported	Boolean	Проверяет, поддерживает ли этот запрос асинхронное выполнение.
isSecure	Boolean	Логический параметр, который показывает, был ли запрос сделан через защищенный канал, такой как HTTPS.
localAddr	String	IP адрес интерфейса, на котором был получен запрос.
localName	String	Имя хоста интерфейса с данным IP, на котором был получен запрос.
localPort	Integer	Номер порта IP, на котором был получен запрос.
method	String	Имя метода HTTP, с помощью которого был сделан запрос, например GET, POST или PUT.
pathInfo	String	Дополнительный информации в пути, относящаяся к URL, отправленному клиентом при создании запроса.
protocol	String	Имя и версия протокола, используемого для запроса, в форме *protocol/majorVersion.minorVersion*, например HTTP/1.1.
queryString	Строка	Строка запроса, которая содержится в URL запроса после пути.
remoteAddr	Строка	IP адрес клиента или последнего прокси, отправившего запрос.
remoteHost	Строка	Полное имя клиента или последнего прокси, отправившего запрос.
remotePort	Integer	Порт источника IP клиента или последнего прокси, отправившего запрос.
remoteUser	String	Логин пользователя, делающего запрос, если пользователь авторизован, либо *null*, если не авторизован.
requestBody	String	Тело запроса в виде строки.
requestedSessionId	String	ID сессии, указанный клиентом.

requestParameters	DataTable	Таблица с параметрами запроса. Может быть пустой, если считывается после тела запроса, или если вид содержимого отличается от *application/x-www-form-urlencoded*.
requestURI	String	Часть URI запроса из имени протокола.
requestURL	String	Реконструированный URL, использованный клиентом при создании запроса.
scheme	String	Имя схемы, используемой для создания запроса, например *http*, *https*, или *ftp*.
serverName	String	Имя хоста сервера, которому был отправлен запрос.
serverPort	Integer	Номер порта, которому был отправлен запрос.

ПАРАМЕТРЫ ОТВЕТА

Параметры ответа представлены в той же таблицы данных по умолчанию. Настраивая их, вы можете создать пользовательский HTTP ответ. Доступны следующие параметры ответа:

Имя	**Тип**	**Описание**
responseBody	String	Тело ответа в виде строки.
responseCharacterEncoding	String	Имя кодировки символов (набор символов MIME), используемой в теле отправляемого ответа.
responseContentLength	Integer	Длина тела содержания ответа (HTTP заголовок Content-Length)
responseContentType	String	Тип содержимого, используемый для тела ответа в формате MIME.
responseError	Integer	Сообщение клиенту об ошибке, использующее указанный код состояния и очищающее буфер.
responseRedirect	String	Отправляет в ответ клиенту сообщение о временной переадресации с использованием URL указанного расположения переадресации.
responseStatus	Integer	Код состояния для ответа.

15.7.3 Примеры HTTP сервера

Пример №1: Получение версии AtomMind Server

Этот пример показывает, как получить версию AtomMind Server, используя HTTP запрос.

- Найдите узел Драйверы/Плагины в Системном дереве и дважды кликните мышью на плагин HTTP сервера .
- Добавьте новую запись в Таблицу настроек HTTP сервера.
- Введите `version` в поле **URI**.
- Введите `set(dt(), "responseBody", 0, {:status$version})` в поле **выражение**.
- Выберите пользователя для вычисления выражения. Например: `users.admin`.
- Перейдите на `http://localhost:8080/version`

Если все сделано правильно, получившаяся страница покажет версию AtomMind Server.

Пример №2: Получение информации о статусе AtomMind Server

Для обработки запросов с параметрами и для других сложных случаев, где одного выражения недостаточно, лучше использовать модели и их наборы правил. Использование переменных среды для хранения промежуточных результатов значительно упростит обработку запросов.

Этот пример показывает, как получить любую информацию из таблицы статусов AtomMind Server.

- Найдите узел Драйверы/Плагины в Системном дереве и дважды кликните мышью на плагин HTTP сервера.

- Добавьте новую запись в Таблицу настроек HTTP сервера.
- Введите `req` в поле **URI**.
- Введите `{users.admin.models.model:request(dt())}` в поле **выражение**.
- Выберите пользователя для вычисления выражения. Например: `users.admin`.

Второй шаг включает создание модели и настройка ее набора правил.

- Создайте новую модель с именем *"model"*.
- Выберите вкладку *Наборы Правил*.
- Добавьте новый набор правил с именем *"request"*.
- Добавьте к этому набору следующие правила:

Цель	Выражение	Условие	Комментарий
parameterValues	cell({requestParameters}, "values")		
value	cell({env/parameterValues} ,"value")		
result	cell({:status} ,{env/value})		
Final Rule Set Result	set(dt(), "responseBody", 0, {env/result})		

Сохраните изменения. Теперь вы можете попробовать следующие запросы:

- `http://localhost:8080/req?parameter=version`
- `http://localhost:8080/req?parameter=name`
- `http://localhost:8080/req?parameter=uptime`

В этих запросах значение параметра может быть именем любого поля таблицы статусов.

15.8 SOAP веб-сервис

AtomMind Server поддерживает технологию *Веб-сервисов* для интеграции со сторонними ERP, CRM и иным программным обеспечением. Он поддерживает SOAP-протокол, используемый для отправки закодированных в XML данных через безопасные соединения по HTTP.

Веб Сервисы - это набор протоколов для обмена общими данными между разными системами.

Веб-сервис AtomMind Server разрешает осуществлять полный контроль за сервером. Можно остановить, перезапустить сервер, управлять пользователями, конфигурировать и контролировать Device, просматривать события Device, его статус и пр.

Технически Веб-сервис предоставляет методы для получения и установки значений переменных 61 контекста 41 и вызова функций 70 контекста. Прослушивание событий 73 контекста в реальном времени не поддерживается, но есть доступ к истории событий.

15.8.1 Доступ к веб-сервису

Веб-сервис AtomMind Server доступен по следующим адресам:

- `https://server_host_name:server_ssl_port/ws/services/ServerWebService` (для безопасных соединений по HTTPS)
- `http://server_host_name:server_non_ssl_port/ws/services/ServerWebService` (для небезопасных соединений по HTTP)

`server_host_name` - это IP-адрес или имя хоста сервера. Оно может совпадать с одним из псевдонимов, определенных в списке псевдонимов для имен хоста в Настройках Глобальной Конфигурации.

server_ssl_port - это номер порта, с которого AtomMind Server принимает входящие безопасные соединения по HTTP (HTTPS). Значение по умолчанию **8443**. Этот порт определен настройками Глобальной Конфигурации в номер порта для прослушивания (безопасных) соединений по HTTPS.

server_non_ssl_port - это номер порта, с которого AtomMind Server принимает входящие соединения по HTTP. Значение по умолчанию **8080**. Этот порт определен настройками Глобальной Конфигурации номер порта для прослушивания (небезопасных) соединений по HTTP.

Небезопасный доступ по HTTP к веб-сервису отключен по умолчанию. Он может быть активирован включением настройки активировать небезопасный доступ к веб-приложениям в Настройках Глобальной Конфигурации.

Пример: Если AtomMind Server запущен на локальном хосте и все значения в настройках установлены по умолчанию, можно подключиться к веб-сервису через следующий URL:

`https://localhost:8443/ws/services/ServerWebService`

15.8.2 Функции веб-сервиса

В этом разделе приводится список всех функций, доступных через Веб-сервис.

Функции, которые используют XML-кодирование Таблиц Данных

1. GETXML

Веб-сервис **getXML** используется для получения значений переменных[61] контекста[41] AtomMind Servera.

Параметры функции:

Имя	Тип	Описание
имя пользователя	Строка	Имя учетной записи пользователя[478] AtomMind Servera, которое будет использоваться для регистрации на сервере. Операция **Получить Переменную** будет выполняться с правами доступа[477] данного пользователя.
пароль	Строка	Пароль для учетной записи пользователя.
контекст	Строка	Путь к контексту для переменной.
переменная	Строка	Имя переменной.

Возвращаемое значение: эта функция возвращает **Строку**, т.е. Таблицу Данных[49], содержащую значение запрашиваемой переменной. Это значение закодировано в формате XML. См. Кодирование Таблиц Данных в формате XML[2130]. Полученная строка XML закодирована вновь согласно стандарту кодирования URL, определенного RFC 1738 (Унифицированный Указатель Ресурсов). Это необходимо для уверенности, что полученная строка не содержит небезопасные для протокола SOAP символы.

2. SETXML

Функция **setXML** позволяет изменять значение для переменной[61] контекста[41] AtomMind Servera.

Параметры функции:

Имя	Тип	Описание
имя пользователя	Строка	Имя учетной записи пользователя[478] AtomMind Server, которое будет использоваться для регистрации на сервере. Операция **Установить переменную** будет выполняться с правами доступа[477] данного пользователя.
пароль	Строка	Пароль для учетной записи пользователя.
контекст	Строка	Путь к контексту переменной.
переменная	Строка	Имя переменной.
значение	Строка	Новое значение переменной, закодированной в формате XML. См. Кодирование Таблиц Данных в формате XML[2130]. Полученный в результате XML-документы закодирован вновь согласно стандарту шифрования URL, определенного RFC 1738 (Унифицированный Указатель Ресурсов). Это необходимо для уверенности в том, что полученная строка не содержит небезопасные для протокола SOAP символы.

Возвращаемое значение: никакое.

В большинстве случаев **значением** аргумента, переданного этой функции, является Таблица Данных, которая была до этого извлечена функцией **getXML**.

3. CALLXML

Функция **callXML** используется для выполнения функций[70] контекста[41] AtomMind Servera посредством передачи в функцию закодированной в формате XML Таблицы Данных в качестве аргумента.

Параметры функции:

Имя	Тип	Описание
имя пользователя	Строка	Имя учетной записи пользователя[478] AtomMind Servera, которое будет использоваться для регистрации на сервере. Функция будет выполняться с правами доступа[477] данного пользователя
пароль	Строка	Пароль учетной записи пользователя.
контекст	Строка	Путь к контексту из которого функция должна вызываться.
переменная	Строка	Имя функции.
значение	Строка	Таблица Данных, содержащая параметры ввода функции. Эта таблица закодирована в формате XML. См. Кодирование Таблиц Данных в формате XML[2130]. Полученный в формате XML документ кодируется вновь согласно стандарту кодирования URL, определенного RFC 1738 (Унифицированный Указатель Ресурсов). Это необходимо для уверенности в том, что полученная строка не содержит небезопасные для протокола SOAP символы.

Возвращаемое значение: эта функция веб-сервиса возвращает строку, которая является Таблицей Данных[49], возвращенной выполненной функцией. Это значение закодировано аналогичным образом, что и входной параметр, упомянутый выше.

Функции, представляющие Таблицы Данных как массив строк

1. SETBYSTRINGARRAY

Функция **setByStringArray** используется для изменения значений переменных[41] контекста[61] AtomMind Servera. Ячейки Таблицы Данных, содержащие новое значение, передаются в нее в виде массива **Строк**.

Параметры функции:

Имя	Тип	Описание
имя пользователя	Строка	Имя учетной записи пользователя[478] AtomMind Servera, которое будет использоваться для регистрации на сервере. Операция **Установить переменную** будет выполняться с правами доступа[477] данного пользователя
пароль	Строка	Пароль учетной записи пользователя.
контекст	Строка	Путь к контексту, где находится переменная.
переменная	Строка	Имя переменной.
значения	Строка[]	Массив строк, используемых для заполнения Таблицы Данных с новыми значениями переменной, как описано здесь.

Возвращаемое значение: нет.

2. CALLBYSTRINGARRAY

Функция **callByStringArray** позволяет выполнять функцию[41] контекста[70] AtomMind Servera путем передачи ее аргумента в виде массива Строк.

Параметры функции:

Имя	Тип	Описание
имя пользователя	Строка	Имя учетной записи пользователя[478] AtomMind Server, которое будет использоваться для регистрации на сервере. **Операция вызова** будет выполняться с правами доступа[477] данного пользователя

пароль	Строка	Пароль учетной записи пользователя.
контекст	Строка	Путь к контексту, из которого вызывается функция.
функция	Строка	Имя функции.
параметры	Строка[]	Массив Строк, который используется для заполнения Таблицы Данных, представляющей значение ввода функции, как описано здесь.

Возвращаемое значение: эта функция Веб-сервиса возвращает **Строку** Таблицы Данных 49, содержащую значение запрашиваемой переменной. Это значение закодировано в формате XML. См. Кодирование Таблиц Данных в формате XML 2130. Полученная строка XML кодируется вновь согласно стандарту шифрования URL, определенного RFC 1738 (Унифицированный Указатель Ресурсов). Это необходимо для уверенности, что полученная строка не содержит небезопасные для протокола SOAP символы.

3. ADDRECORDBYSTRINGSRRAY

Функция **addRecordByStringSrray** используется для создания новых записей в табличной переменной 61 контекста 41. Поля новой записи Таблицы Данных передаются в нее в виде массива **Строк**.

Технически функция читает значение переменной, добавляет в нее новую запись и записывает новое значение назад в контекст.

Параметры функции:

Имя	Тип	Описание
имя пользователя	Строка	Имя учетной записи пользователя 478 AtomMind Server, которое будет использоваться для регистрации на сервере. Операции **Получить переменную**/**Настроить переменную** будут выполняться с правами доступа 477 данного пользователя.
пароль	Строка	Пароль учетной записи пользователя.
контекст	Строка	Путь к контексту, где находится переменная.
переменная	Строка	Имя переменной.
значение	Строка[]	Массив строк, используемый для заполнения новой записи Таблицы Данных со значениями, описанными здесь.

Возвращаемое значение: нет.

4. SETVARIABLEFIELD

Функция **setVariableField** используется для изменения одной ячейки (задано именем поля и номером ряда) переменной 61 контекста 41. Новое значение ячейки передается в виде **Строки**.

Технически эта функция читает значение переменной, меняет значение ее ячейки (переменная типа Таблица данных 49), и записывает новое значение обратно в контекст.

Параметры функции:

Имя	Тип	Описание
имя пользователя	Строка	Имя учетной записи пользователя 478 AtomMind Servera, которое будет использоваться для регистрации на сервере. Операции **Получить переменную**/**Настроить переменную** будут выполняться с правами доступа 477 данного пользователя.
пароль	Строка	пароль учетной записи пользователя.
контекст	Строка	Путь к контексту, где находится переменная.
переменная	Строка	Имя переменной.
поле	Строка	Имя поля.
ряд	int	Номер ряда.
значение	Строка	Новое значение ячейки в форме Строки.

Возвращаемое значение: никакое.

Функции, которые используют собственные кодировки Таблиц Данных

Существуют три функции Веб-сервиса, которые используют собственное кодирование Таблиц Данных 2123:

- **get**
- **set**
- **call**

Эти функции используются для получения/установки значений переменных контекста AtomMind Servera и вызова соответствующих функций контекста.

15.8.3 Примеры веб-сервиса

Пример 1: Доступ к веб-сервису AtomMind Servera из PHP-скрипта

Этот пример показывает, как получить значение переменной контекста AtomMind Servera из скрипта PHP при помощи Веб-сервиса.

Полный исходный код этого примера доступен в разделе **Загрузки** на вебсайте AtomMind.

```
<?php
// Creating SOAP client for the LinkServer running on localhost
$client = new SoapClient("https://localhost:8443/ws/?wsdl");

$username = "admin";
$password = "admin";
$context = "users.admin";
$variable = "childInfo";

try
{
	// Calling getXML Web Service function and decoding its output by URL decoder
	$userInfoXML = urldecode($client->getXML($username, $password, $context, $variable));
}
catch (SoapFault $fault)
{
	echo "Error occurred while calling remote function: ".$fault->faultcode." (".$fault->faults
	exit();
}

if (is_soap_fault($userInfoXML))
{
	echo "Error occurred while calling remote function: ".$fault->faultcode." (".$fault->faults
	exit();
}

// Creating DOM document from XML
$xml = new DomDocument;
$xml->loadXML( $userInfoXML );

// Loading XSLT transformation
$xsl = new DomDocument;
$xsl->load( 'resources/xslt/dataTable.xslt' );

// Creating XSLT processor
$proc = new xsltprocessor;
$proc->importStyleSheet( $xsl );

// Processing out XML document and and showing the output
echo $proc->transformToXML( $xml );
?>
```

Пример 2: Вызов функции контекста

Это пример показывает, как создавать новую учетную запись пользователя 478 AtomMind Servera путем вызова функции **регистрации** из корневого контекста 1559. Создание клиента SOAP и проверка ошибок опущены в этом примере.

```
$params[0]="phpuser"; // Username (value for record 0, field 0 of function input Data Table)
$params[1]="test"; // Password (value for record 0, field 1)
$params[2]="test"; // Repeat Password (value for record 0, field 2)
urldecode($client->callByStringArray("admin", "admin", "", "register", $params)); // Calling funct
```

16 Расширенные функции

Этот раздел описывает расширенные модули и функции AtomMind Server.

16.1 Сессии и переменные сессии

Сессии позволяют серверу и компонентам UI обмениваться данными, специфичными для одного текущего пользователя системы, например, оператора или стороннего приложения, работающего через API.

Взаимодействия внутри сессии реализуются через *переменные сессии,* существующие, пока длится сессия.

Для записи и чтения переменных сессии можно использовать специальный набор функций и событий внутри корневого контекста 1559.

Функция `sessionSet()` задает значение переменной. Переменная доступна, пока открыта текущая сессия пользователя. Можно запросить ее значение через виджет или инструментальную панель.

Например, можно использовать это выражение.

```
callFunction({:}, "sessionSet", "test1", table("<<strTest><S>>","newStr1",
"newStr2"))
```

Функция `sessionGet()` получает значение упомянутой переменной сессии.

Пример выражения:

```
callFunction({:}, "sessionGet", "test1")
```

Событие `sessionVariableUpdated` возникает при каждом обновлении переменной сессии, поэтому его можно использовать как активатор привязки виджета или модели для дальнейшей обработки данных.

Также можно использовать стандартные переменные сессии `type, login, username`. Они помогают создавать разные человеко-машинные интерфейсы для разных типов пользовательских подключений. Например, внутри выражения пригодности 917 инструментальной панели.

- Переменная `type` хранит данные о подключении, если пользователь подключился через Web UI или AtomMind Client.
- Переменная `username` содержит имя пользователя AtomMind, используемое для аутентификации в текущей сессии.
- Переменная `login` показывает, какие данные были введены на экране входа с систему. Ее значение может отличаться от переменной `username`. Такое может произойти при использовании метода внешней аутентификации.
- `displayName` показывает предпочтительное имя для объекта при отображении записей. Это обычно комбинация имени пользователя, первой буквы его отчества и фамилии.
- Переменная `company` хранит имя компании или организации пользователя.
- `department` содержит название департамента, в котором работает пользователь.
- `title` определяет должность пользователя. Используется для отображения формальной должности, а не рода занятий, например, Senior Programmer, а не просто программист. Обычно не используется для должностей с сокращениями, например, Esq. или DDS.

Пример выражения пригодности:

```
cell(callFunction({:}, "sessionGet", "type"), "value") == "Web"
```

16.2 Универсальный поиск

В AtomMind Server доступен интегрированный поиск на основе индексации единой модели данных 41. Интерфейс универсального поиска может использоваться для нахождения различных контекстов, переменных, функций и событий.

Вызов окна универсального поиска осуществляется через действие **Search**, доступное в корневом контексте 1559. В диалоговом окне 382 строка из текстового поля поиска используется для поиска по всему дереву контекстов 41 и базе данных 692 AtomMind Server.

Конфигурация индексации

Настройки индексации находятся в узле Поиск раздела конфигурации плагинов 207. Настройки включают в себя несколько элементов:

НАСТРОЙКИ ИНДЕКСАЦИИ

Эта таблица задает правила индексации различных групп контекстов. Каждая запись содержит:

- **Маску контекстов** для определения группы контекстов данного правила.

Если доступный путь контекста 41 ресурса совпадает 45 или расширяет 44 маску контекста, определенную текущей строкой таблицы параметров индексации, определенные в этой строке настройки индексации будут использованы как *настройки индексации* для пути. Как только найдена соответствующая строка, ни одна другая строка не сопоставляется. Так, например, если вы начали таблицу с контекста * (т.е, это первая строка в таблице), другие строки проверяться не будут, так как данная строка соответствует всем контекстам. Именно поэтому * всегда должен быть последней строкой таблицы.

- Флажок **контекста**, который разрешает индексирование этой группы и основных свойств контекстов этой группы (такие как имя, описание, тип, и т.д.)
- Флажок **переменных**, который разрешает индексирование метаданных переменных (имена, описания, параметры поля, и т.д.)
- Флажок **функций**, который разрешает индексирование метаданных функций (имена, описания, параметры ввода/вывода и т.д.)
- Флажок **событий**, который разрешает индексирование метаданных событий (имена, описания, параметры полей, и т.д.)
- Флажок **действий**, который разрешает индексирование метаданных действий (имена, описания, и т.д.)
- Таблицу **Имена переменных**, которая содержит список переменных, чьи значения будут добавлены в индекс.

Данный параметр может быть изменен, только если **Переменные** активированы.

НАСТРОЙКИ ПОИСКА

Данное свойство определяет параметр **Папка хранения индекса**, который определяет месоположение папки с поисковым индексом. Путь может быть абсолютным или относительным папки установки AtomMind Server.

НАСТРОЙКИ ИНДЕКСАЦИИ КЛАССА

В этой таблице вы можете настроить, какие классы и экземпляры классов индексировать. Таблица вводит следующие параметры:

- **Контекст класса** используется для определения контекста класса, чьи экземпляры необходимо индексировать.
- **Поля класса** используется для определения поля, которые необходимо индексировать.

16.3 Динамическая система доменных имен (DNS)

AtomMind Server может регистрировать любой Agent, который соединяется с ним в системе доменных имен (DNS), используя обновления динамических DNS (RFC 2136). Это также позволяет соединиться с устройством, используя имя хоста вместо его IP-адреса, который останется динамическим.

Функциональность динамической DNS доставляется в отдельный плагин 207 динамической DNS. Настройки плагина описаны здесь 1425.

См. больше информации о том, как работает динамическая DNS, в статье сервис динамической DNS 2082.

16.3.1 Настройка динамической DNS

Опции в этой группе отвечают за обслуживание динамической DNS.

Активировать обновления динамической DNS

Ключевое имя в файле конфигурации: ddnsEnabled

Тип значения: Boolean

Возможные значения: true или false

Значение по умолчанию: false

Позволяет соединениям Серверов устройств 2087 регистрироваться в DNS. Для самостоятельной регистрация сервера устройства в dDNS, регистрация dDNS должна быть также активирована в настройках.

Имя зоны

Ключевое имя в файле конфигурации: ddnsZone

Тип значения: String

Возможные значения: Любое действующее имя домена DNS

Значение по умолчанию: "" (пусто)

Эта опция определяет имя зоны (домена) DNS, к которой AtomMind Server добавит хосты, относящиеся к Серверам устройств. Это может быть доменное имя в локальной DNS вашей компании (в сети (LAN) или Intranet) или глобальное имя домена, например, "*company.com*" (если вы им владеете).

IP-адрес сервера DNS

Ключевое имя в файле конфигурации: ddnsServerIp

Тип значения: String

Возможные значения: Любой IP-адрес

Значение по умолчанию: "" (пусто)

Определяет IP-адрес сервера DNS, используемый для обновлений динамической DNS.

Номер порта для взаимодействия на сервере DNS

Ключевое имя в файле конфигурации: ddnsServerPort

Тип значения: Integer

Возможные значения: 1-65535

Значение по умолчанию: 53

Определяет номер конечного порта на сервере DNS, который используется для обновлений динамической DNS.

Активировать ключи Transaction Signature (TSIG)

Ключевое имя в файле конфигурации: ddnsUseTsigKeys

Тип значения: Boolean

Возможные значения: true или false

Значение по умолчанию: false

Эта опция активирует использование ключей Transaction Signature (TSIG) для защиты взаимодействия с DNS сервером. Ключи TSIG - это форма безопасности DNS. Подробнее об этом можно узнать, пройдя, пройдя по ссылке http://en.wikipedia.org/wiki/TSIG

Имя ключа TSIG

Ключевое имя в файле конфигурации: ddnsTsigKeyName

Тип значения: String

Возможные значения: Любое строковое значение, соответствующее спецификации динамической DNS

Значение по умолчанию: "" (пусто)

Определяет имя ключа, используемого для подписей транзакции (TSIG).

Значение ключа TSIG

Ключевое имя в файле конфигурации: : ddnsTsigKeyValue

Тип значения: String

Возможные значения: Любое строковое значение, соответствующее спецификации динамической DNS

Значение по умолчанию: "" (пусто)

Определяет значение ключа, используемого для подписей транзакции (TSIG).

Активирвоать режим TCP

Ключевое имя в файле конфигурации: ddnsTcpMode

Тип значения: Boolean

Возможные значения: true или false

Значение по умолчанию: false

Если эта опция установлена на значение `true`, используется протокол TCP для обновлений, иначе все транзакции выполняются, используя UDP.

Таймаут операций для DNS в секундах

Ключевое имя в файле конфигурации: ddnsTimeout

Тип значения: Integer

Возможные значения: Любое положительное целое число

Значение по умолчанию: 10

Определяет таймаут для операций обновлений DNS. Если в течение этого времени нет ответа от сервера DNS, операция обновления DNS считается неудачной.

TTL для новых записей DNS в секундах

Ключевое имя в файле конфигурации: ddnsTtl

Тип значения: Integer

Возможные значения: Любое положительное целое число

Значение по умолчанию: 600

Определяет интервал Time To Live (TTL), который устанавливается для всех недавно созданных записей DNS. Когда кэширующий (рекурсивный) DNS-сервер запрашивает рекурсивную запись у доверенного DNS-сервера, то он будет кэшировать запись за время (в секундах), установленное TTL. Низкое значение этой настройки приводит к высокому количеству обновлений dDNS. Высокое значение влечет за собой более медленное обновление информации (и поэтому меньше запросов обновления). Это значение указывается в секундах (600 секунд соответствует 10 минутам).

Шаблон хоста для регистрации внешних IP

Ключевое имя в файле конфигурации: ddnsHostPatternForExternalIPs

Тип значения: String

Возможные значения: Строка, содержащая особые маркеры

Значение по умолчанию: $n.$o.ext

Определяет шаблон имени хоста, который будет использован для внешних Серверов устройств. Строка обрабатывается, создаются следующие замены:

- `$n` изменяется относительно имени устройства отдельного Сервера устройства (как определено в его внутренних настройках)
- `$o` изменяется относительно имени владельца отдельного Сервера устройства (так же, как определено в настройках устройства)

Результат используется в качестве префикса к имени зоны полного имени домена. Таким образом, например, со значением по умолчанию для этой настройки, если Сервер устройства называется `LightSaber`, имя владельца `Luke`, а имя домена `DeathStar.com`, доступ к Серверу устройства будет возможен, используя адрес DNS `lightsaber.luke.deathstar.com`.

Шаблон хоста для регистрации внутренних IP

Ключевое имя в файле конфигурации: ddnsHostPatternForInternalIPs

Тип значения: String

Возможные значения: Строка, содержащая особые маркеры

Значение по умолчанию: $n.$o.int

Определяет шаблон имени хоста, который будет использован для регистрации Серверов устройств, чьи IP-адреса являются внутренними (в вашей сети (LAN)). Строка обрабатывается, создаются следующие замены:

- `$n` изменяется на имя устройства отдельного Сервера устройства
- `$o` изменяется на имя владельца отдельного Сервера устройстваег

Результат используется в качестве префикса к имени зоны полного имени домена (см. пример выше).

16.3.2 Справочник по контекстам динамической DNS

Модуль динамической DNS добавляет следующие сущности в дерево контекстов 1450 AtomMind Server.

Корневой контекст

Следующие действия 87 добавляются в корневой 1559 контекст:

ОБНОВИТЬ ХОСТ

Вызов функции 103 дает команду AtomMind Server добавить или удалить хост из DNS, используя обновление динамического DNS.

Тип действия:	Вызов функции 103
Имя действия:	updateHost
Права доступа:	Доступно на уровне 486 прав доступа *Администратор*
Группа действий:	Дополнительные действия

Контекст утилитов

Следующие функции 103 добавляются к контексту утилитов 1613:

ОБНОВИТЬ ХОСТ

Обновляет хост в динамической DNS.

Имя функции:	updateHost
Права доступа:	Доступно на уровне 486 прав доступа *Оператор*
Записи входа:	1

Формат 50 входа:

Имя	Тип	Описание
host	String	Имя хоста для обновления. Зона DNS для обновлений может быть конфигурирована в глобальных настройках динамической DNS 1424.
ip	String	IP-адрес хоста.

remove	Boolean	Отменяет регистрацию хоста в зоне DNS, если значение false, и регистрирует, если значение true.

Записи выхода: 0

Формат 50 выхода: Отсутствует

16.4 E-Mail сообщения

Этот раздел описывает, как отправлять и получать электронные сообщения, используя AtomMind Server.

Например, сообщения могут отправляться относительно тревог 779, а ответы на них позволяют распознать тревогу без соединения с сервером.

Функциональность управления e-mail предоставляется в отдельном плагине 207 E-mail. Конфигурация плагина описана здесь 1428.

16.4.1 Настройки почты

Опции в этой группе используются для того, чтобы определить, как AtomMind Server взаимодействует с серверами E-mail (SMTP и POP3).

Активировать отправку почты

Ключевое имя в файле конфигурации: mailSendingEnabled

Тип значения: Boolean

Возможные значения: true или false

Значение по умолчанию: false

Позволяет AtomMind Server отправлять и получать электронные сообщения.

Адрес сервера исходящей почты (IP или имя хоста сервера SMTP)

Ключевое имя в файле конфигурации: mailOutgoingAddress

Тип значения: String

Возможные значения: Любой существующий IP-адрес или имя хоста

Значение по умолчанию: localhost

Опция определяет адрес сервера почты SMTP, используемый AtomMind Server.

Пример: Чтобы активировать отправку сообщения с помощью сервера Google Mail (@gmail.com), установите это свойство на `smtp.gmail.com`.

Серверное имя пользователя исходящей почты (логин)

Ключевое имя в файле конфигурации: mailOutgoingUsername

Тип значения: String

Возможные значения: Любое подходящее имя пользователя для сервера почты

Значение по умолчанию: "" (пусто)

Эта опция определяет имя пользователя, использованное во время аутентификации SMTP.

Пароль сервера исходящей почты

Ключевое имя в файле конфигурации: mailOutgoingPassword

Тип значения: String

Возможные значения: Любое подходящий пароль для сервера почты

Значение по умолчанию: "" (пусто)

Эта опция определяет пароль, использованный во время аутентификации SMTP.

Адрес отправителя для электронных сообщений

Ключевое имя в файле конфигурации: mailSenderAddress

Тип значения: String

Возможные значения: Любой подходящий адрес e-mail

Значение по умолчанию: AtomMind Server@your-domain.com

Эта опция определяет e-mail адрес, указанный в графе FROM email сообщений, отправленных AtomMind Server.

Имя отправителя для сообщений почты

Ключевое имя в файле конфигурации: mailSenderName

Тип значения: String

Возможные значения: Любое подходящее имя отправителя e-mail

Значение по умолчанию: AtomMind Server

Эта опция определяет имя отправителя, указанное в графе FROM email сообщений, отправленных AtomMind Server.

Адрес сервера входящей почты (IP или имя хоста сервера POP3)

Ключевое имя в файле конфигурации: mailIncomingAddress

Тип значения: String

Возможные значения: Любой подходящий IP-адрес или имя хоста

Значение по умолчанию: localhost

Эта опция определяет адрес сервера почты POP3, используемый AtomMind Server.

Пример: Чтобы активировать отправку сообщения с помощью сервера Google Mail (@gmail.com), установите это свойство на `pop.gmail.com`.

Имя пользователя сервера входящей почты (логин)

Ключевое имя в файле конфигурации: mailIncomingUsername

Тип значения: String

Возможные значения: Любое подходящее имя пользователя для сервера

Значение по умолчанию: "" (пусто)

Эта опция определяет имя пользователя, использованное во время аутентификации POP3.

Серверный пароль входящей почты

Ключевое имя в файле конфигурации: mailIncomingPassword

Тип значения: String

Возможные значения: Любой подходящий пароль для сервера почты

Значение по умолчанию: "" (пусто)

Эта опция определяет пароль, использованный во время аутентификации POP3.

16.4.2 Дополнительные настройки почты

Таблица дополнительных настроек почты позволяет производить тонкую настройку параметров отправки и получения почты. Каждый параметр определен **Именем** и **Значением**. Однако все параметры вносятся в форме строки, а **Тип** параметров в таблице ниже просто объясняет, как интерпретируется значение параметра. Например, если параметр логический, его строковое значение должно быть `true` или `false`.

В большинстве случаев достаточно использовать только базовые опции 1428 для должной настройки e-mail операций. Используйте дополнительные настройки только если ваши базовые настройки не работают, и вы знакомы с e-mail операциями.

Пример: Чтобы включить отправку почты через SSL SMTP сервер, установите следующие дополнительные опции:

mail.smtp.port=465

mail.smtp.auth=true

mail.smtp.ssl.enable=true

Пример: Чтобы включить отправку почты через сервер Google Mail (@gmail.com), используйте следующие дополнительные опции:

mail.smtp.port=587

mail.smtp.auth=true

mail.smtp.ssl.enable=true

mail.smtp.ssl.protocols=TLSv1

Общие параметры

Имя	Тип	Описание
mail.debug	Boolean	Предварительный режим отладки. По умолчанию false.
mail.from	String	Обратный e-mail адрес текущего пользователя.
mail.mime.address.strict	Boolean	Активирует точный разбор строк заголовков сообщений. По умолчанию true.
mail.host	String	Имя хоста почтового сервера по умолчанию. Используется, если свойство mail.*protocol*.host не задано.
mail.store.protocol	String	Определяет протокол доступа к сообщениям по умолчанию ("pop3", "pop3s", "imap" или "imaps").
mail.transport.protocol	String	Определяет протокол передачи сообщений по умолчанию ("smtp" или "smtps").
mail.user	String	Имя пользователя по умолчанию для подключения к почтовому серверу. Используется, если свойство mail.*protocol*.user не задано.
mail.password	String	Пароль по умолчанию для подключения к почтовому серверу. Используется, если свойство mail.*protocol*.password не задано.
mail.*protocol*.host	String	Имя хоста почтового сервера для определенного протокола. Переопределяет свойство mail.host.
mail.*protocol*.port	Integer	Номер порта почтового сервера для определенного протокола. Если протокол не определен, используется номер порта протокола по умолчанию.
mail.*protocol*.user	String	Имя пользователя для подключения к почтовым серверам по определенному протоколу. Переопределяет свойство mail.user.
mail.*protocol*.password	String	Пароль для подключения к почтовым серверам по определенному протоколу. Переопределяет свойство mail.password.

SMTP параметры

Обратите внимание, что при использовании "smtps" протокола для доступа SMTP через SSL, все свойства будут носить имя "mail.smtps.*".

Имя	Тип	Описание
mail.smtp.port	Integer	Порт SMTP сервера для подключения. По умолчанию 25.
mail.smtp.connectiontimeout	Integer	Значение таймаута сокета соединения в миллисекундах. Бесконечный таймаут по умолчанию.
mail.smtp.timeout	Integer	Значение таймаута сокета ввода/вывода в миллисекундах. Бесконечный таймаут по умолчанию.
mail.smtp.localhost	String	Имя локального хоста, используемого для SMTP команд HELO или EHLO. Обычно не требует настройки, если пространство имен сконфигурировано правильно.
mail.smtp.localaddress	String	Локальный адрес (имя хоста) для привязки при создании SMTP соединения. Обычно не требует настройки, но бывает полезным в случае многосетевых хостов, когда важно выбрать определенный локальный адрес для привязки.
mail.smtp.localport	Integer	Номер локального порта для привязки при создании SMTP соединения.
mail.smtp.ehlo	Boolean	Если false, не пытается войти с помощью команды EHLO. Значение по умолчанию true. Обычно, сбой команды EHLO приводит к использованию команды HELO; это свойство существует только для серверов, которые не выдают должным образом сбой команды EHLO или не реализуют должным образом команду EHLO.
mail.smtp.auth	Boolean	Если true, пытается авторизовать пользователя с помощью команды AUTH. Значение по умолчанию false.
mail.smtp.auth.mechanisms	String	Если установлен, выдает список механизмов аутентификации для рассмотрения, а также порядок их рассмотрения. Используются только механизмы, поддерживаемые сервером и поддерживаемые текущей реализацией. По умолчанию LOGIN PLAIN DIGEST-MD5 NTLM, которое включает все механизмы аутентификации, поддерживаемые текущей реализацией.
mail.smtp.auth.login.disable	Boolean	Если true, предотвращает использование команды AUTH LOGIN. По умолчанию false.
mail.smtp.auth.plain.disable	Boolean	Если true, предотвращает использование команды AUTH PLAIN. По умолчанию false.
mail.smtp.auth.digest-md5.disable	Boolean	Если true, предотвращает использование команды AUTH DIGEST-MD5. По умолчанию false.
mail.smtp.auth.ntlm.disable	Boolean	Если true, предотвращает использование команды AUTH NTLM. По умолчанию false.
mail.smtp.auth.ntlm.domain	String	NTLM домен аутентификации.
mail.smtp.auth.ntlm.flags	Integer	Флажки, специфичные для NTLM протокола. Для более подробной информации см. http://curl.haxx.se/rfc/ntlm.html#theNtlmFlags.
mail.smtp.submitter	String	Отправитель, используемый в теге AUTH команды MAIL FROM. Обычно используется почтовым транслятором для передачи информации об оригинальном отправителе сообщения. Обычно не используется почтовыми клиентами.
mail.smtp.dsn.notify	String	Опция NOTIFY команды RCPT. Либо NEVER, либо комбинация из SUCCESS, FAILURE, и DELAY (разделенные запятыми).
mail.smtp.dsn.ret	String	Опция RET команды MAIL. Либо FULL, либо HDRS.
mail.smtp.allow8bitmime	Boolean	Если установлено в true, и сервер поддерживает расширение 8BITMIME, текстовые части сообщения, использующие кодировки "quoted-printable" или "base64", переводятся на использование кодировки "8bit", если они соответствуют правилам RFC2045 для 8-разрядного текста.
mail.smtp.sendpartial	Boolean	Если установлено в true, а сообщение содержит как верные, так и неверные адреса, производит отправку сообщения в любом случае, сообщая о частичном сбое с исключением. Если установлено в false (по умолчанию), сообщение не отправляется ни одному из получателей при наличии хотя бы одного неверного адреса.

mail.smtp.sasl.enable	Boolean	Если установлено в true, производит попытку использовать SASL для выбора механизма аутентификации при входе. По умолчанию false.
mail.smtp.sasl.mechanisms	String	Список имен SASL механизмов, разделенных пробелом или запятой, для возможного использования.
mail.smtp.sasl.authorizationid	String	ID авторизации для использования при SASL аутентификации. Если не установлено, используется ID аутентификации (имя пользователя).
mail.smtp.sasl.realm	String	Область, используемая при DIGEST-MD5 аутентификации.
mail.smtp.quitwait	Boolean	Если установлено в false, отправляется команда QUIT, и соединение немедленно отключается. Если установлено в true (по умолчанию), приводит к ожиданию передачей ответа на команду QUIT.
mail.smtp.ssl.enable	Boolean	Если установлено в true, использует SSL для соединения и SSL порт по умолчанию. По умолчанию false для протокола "smtp" и true протокола "smtps".
mail.smtp.ssl.checkserveridentity	Boolean	Если установлено в true, проверяет идентификацию сервера как указано RFC 2595. Эти дополнительные проверки на основе содержания сертификата сервера нацелены предотвратить атаки "перехватчика". По умолчанию false.
mail.smtp.ssl.trust	String	Если установлено в "*", доверяет всем хостам. Если настроен на разделенный пробелами список хостов, доверяет всем этим хостам. В противном случае, доверие хостам зависит от сертификата сервера.
mail.smtp.ssl.socketFactory.port	Integer	Определяет порт для подключения при использовании SSL. Если не задано, используется порт по умолчанию.
mail.smtp.ssl.protocols	String	Определяет SSL протоколы, которые будут активированы для SSL подключения. Значение свойства - разделенный пробелами список токенов. Допустимые значения SSLv3 и TLSv1.
mail.smtp.ssl.ciphersuites	String	Определяет наборы шифров SSL, которые будут активированы для SSL соединения. Значение свойства - разделенный пробелами список токенов.
mail.smtp.mailextension	String	Строка расширения для присоединения к команде MAIL. Строка расширения может использоваться для определения стандартных расширений сервиса SMTP, а также расширений, специфичных для производителя. Например, RFC 1869 и другие расширения RFC, которые определяют специфичные расширения.
mail.smtp.starttls.enable	Boolean	Если true, активирует использование команды STARTTLS (если поддерживается сервером) для переключения соединения на TLS защищенное соединение до выдачи каких-либо команд авторизации. Обратите внимание, что соответствующее хранилище сертификатов должно быть сконфигурировано, чтобы клиент доверял сертификату сервера. По умолчанию false.
mail.smtp.starttls.required	Boolean	Если true, требует использовать команду STARTTLS. Если сервер не поддерживает команду STARTTLS, или если команда выдает сбой, метод подключения также выдаст сбой. По умолчанию false.
mail.smtp.userset	Boolean	Если установлено в true, использует команду RSET вместо команды NOOP для проверки состояния соединения. В некоторых случаях отправка почты будет отвечать медленно после многократного применения команд NOOP. Избежать этой проблемы с отправкой почты помогает использование RSET. По умолчанию false.
mail.smtp.noop.strict	Boolean	Если установлено в true (по умолчанию), требует кода ответа 250 от команды NOOP для индикации успешного выполнения. Команда NOOP используется для определения, сохранилось ли соединение подключенным. Некоторые более старые серверы возвращают неверный код ответа при успешном выполнении, некоторые серверы вообще не осуществляют команду NOOP и всегда возвращают код отказа. Установите это свойство в false для работы с серверами с подобными недочетами. Обычно, когда время соединения с сервером истекает, сервер посылает код ответа 421, который становится видимым клиенту как ответ на следующую производимую команду. Некоторые серверы отправляют неверный код ответа на отказ, когда истекает время соединения. Не устанавливайте это свойство в false для работы с серверами с подобными недочетами

POP3 параметры

Обратите внимание, что при использовании "pop3s" протокола для доступа POP3 через SSL, все свойства будут носить имя "mail.pop3s.*".

Имя	Тип	Описание
mail.pop3.port	Integer	Порт сервера POP3 для подключения. По умолчанию 110.
mail.pop3.connectiontimeout	Integer	Значение таймаута сокета соединения в миллисекундах. Бесконечный таймаут по умолчанию.
mail.pop3.timeout	Integer	Значение таймаута сокета ввода/вывода в миллисекундах. Бесконечный таймаут по умолчанию.
mail.pop3.rsetbeforequit	Boolean	Посылает POP3 команду RSET при закрытии папки до отправки команды QUIT. Полезно для POP3 серверов, которые отмечают все прочитанные сообщения как удаленные. Предотвращает такие сообщения от удаления и стирания без запроса клиента. По умолчанию false.
mail.pop3.localaddress	String	Локальный адрес (имя хоста) для привязки при создании POP3 сокета. Обычно не требует настройки, но бывает полезным в случае многосетевых хостов, когда важно выбрать определенный локальный адрес для привязки.
mail.pop3.localport	Integer	Номер локального порта для привязки при создании POP3 сокета.
mail.pop3.apop.enable	Boolean	Если установлено в true, использует APOP вместо USER/PASS для авторизации на POP3 сервере, если POP3 сервер поддерживает APOP. APOP посылает хэш пароля, а не открытый текст пароля. По умолчанию false.
mail.pop3.ssl.enable	Boolean	Если установлено в true, использует SSL для соединения и SSL порт по умолчанию. По умолчанию false для "pop3" протокола и true для "pop3s" протокола.
mail.pop3.ssl.checkserveridentity	Boolean	Если установлено в true, проверяет идентификацию сервера как указано RFC 2595. Эти дополнительные проверки на основе содержания сертификата сервера нацелены предотвратить атаки "перехватчика". По умолчанию false.
mail.pop3.ssl.trust	String	Если установлено в "*", доверяет всем хостам. Если настроен на разделенный пробелами список хостов, доверяет всем этим хостам. В противном случае, доверие хостам зависит от сертификата сервера.
mail.pop3.ssl.socketFactory.port	Integer	Определяет порт для подключения при с использованием SSL. Если не задано, используется порт по умолчанию.
mail.pop3.ssl.protocols	String	Определяет SSL протоколы, которые будут активированы для SSL подключения. Значение свойства - разделенный пробелами список токенов.
mail.pop3.ssl.ciphersuites	String	Определяет наборы шифров SSL, которые будут активированы для SSL соединения. Значение свойства - разделенный пробелами список токенов.
mail.pop3.starttls.enable	Boolean	Если true, активирует использование команды STLS (если поддерживается сервером) для переключения соединения на TLS защищенное соединение до выдачи каких-либо команд авторизации. Обратите внимание, что соответсвующее хранилище сертификатов должно быть сконфигурировано, чтобы клиент доверял сертификату сервера. По умолчанию false.
mail.pop3.starttls.required	Boolean	Если true, требует использовать команду STLS. Если сервер не поддерживает команду STLS, или если команда выдает сбой, метод подключения также выдаст сбой. По умолчанию false.
mail.pop3.disabletop	Boolean	Если установлено в true, POP3 команда TOP не будет использоваться для извлечения заголовков сообщений. Используется для POP3 серверов, которые не осуществляют должным образом команду TOP, либо предоставляют неверную информацию в результатах команды TOP. По умолчанию false.
mail.pop3.disablecapa	Boolean	Если установлено в true, POP3 команда CAPA не будет использоваться для получения возможностей сервера. Полезно для POP3 серверов, которые не реализуют должным образом команду CAPA, или предоставляют неверную информацию в результатах команды CAPA. По умолчанию false.
mail.pop3.forgettopheaders	Boolean	Если установлено в true, заголовки, которые могли быть извлечены с использованием POP3 команды TOP, будут забыты и заменены заголовками, извлеченными как часть POP3 команды RETR. Некоторые серверы, например, некоторые версии Microsft Exchange и IBM Lotus Notes, будут возвращать слегка измененные заголовки каждый раз при использовании команд TOP или RETR. Чтобы POP3 провайдер мог корректно осуществлять разбор содержания сообщения, возвращенного командой RETR, необходимо использовать заголовки, также возвращенные командой RETR. Установка этого свойства в true приведет к использованию таких заголовков, даже если они

		отличаются от заголовков, возвращенных ранее в результате использования команды TOP. По умолчанию false.
mail.pop3.filecache.enable	Boolean	Если установлено в true, POP3 провайдер будет кэшировать данные сообщений во временный файл, а не в память. Сообщения добавляются в кэш только при доступе к содержанию сообщения. Заголовки сообщений всегда кэшируются в память (по требованию). Файловый кэш удаляется при закрытии папки или остановке сервера. По умолчанию false.
mail.pop3.filecache.dir	String	При активации файлового кэша, данное свойство можно использовать для переопределения директории, используемой по умолчанию для временных файлов.
mail.pop3.keepmessagecontent	Boolean	Содержание сообщения кэшируется при первом извлечении. Обычно закэшированное содержимое стирается при недостатке памяти и при необходимости извлекается повторно. Если данное свойство установлено в true, все кэшированное содержание останется нетронутым, и память не будет использоваться повторно до закрытия папки. По умолчанию false.

16.4.3 Справочник по контекстам E-mail

Модуль E-mail добавляет следующие сущности в дерево контекстов 1450 AtomMind Server .

Корневой контекст

Следующие действия 87 добавляются в корневой 1559 контекст:

ОТПРАВИТЬ EMAIL

Эти инструкции Вызов функции 103 AtomMind Server необходимы для отправления сообщений e-mail.

Тип действия: Вызов функции 103

Имя действия: sendMail

Группа действий: Дополнительные действия

Права доступа: Доступно на уровне 486 прав доступа *Администратор*

ПРОВЕРИТЬ ВХОДЯЩУЮ ПОЧТУ

Этот Вызов функции 103 заставляет AtomMind Server получать почту от входящего почтового сервера.

Тип действия: Вызов функции 103

Имя действия: checkMail

Права доступа: Доступно на уровне 486 прав доступа *Инженер*

Группа действий: Дополнительные действия

Контекст утилитов

Следующие функции 70 добавляются в контекст утилитов 1613:

ПРОВЕРИТЬ MAIL

Обучает AtomMind Server получать и обрабатывать входящие сообщения e-mail.

Имя функции: checkMail

Права доступа: Доступно на уровне 486 прав доступа *Инженер*

Записи входа: 0

Формат входа: нет

Записи выхода: 0

Формат выхода: нет

ПОЛУЧИТЬ ВХОДЯЩУЮ ПОЧТУ

Получить список входящих сообщений e-mail. Работает только с серверами почты POP3.

Имя функции: getMail

Права доступа: Доступно на уровне 486 прав доступа *Инженер*

Записи входа: 1

Формат входа:

Имя	Тип	Описание
address	String	Адрес сервера входящих e-mail (IP или имя хоста сервера POP3).
username	String	Имя пользователя (логин) сервера e-mail.
password	String	Пароль сервера входящих.
properties	DataTable	Таблица дополнительных настроек почты 1430.

Записи выхода: 0...без ограничений

Формат выхода:

Имя	Тип	Описание
sender	String	E-mail и имя автора сообщения (опционально).
subject	String	Тема сообщения.
text	String	Текст сообщения.
headers	DataTable	Список заголовков e-mail.
attachments	DataTable	Список прикреплений e-mail.
to	DataTable	Список получателей.
cc	DataTable	Список получателей в копии.
bcc	DataTable	Список получателей в слепой копии.

ОТПРАВИТЬ EMAIL

Отправляет email. Эта функция будет качественно работать, если верно установлена отправка сообщений 1428 в глобальной конфигурации сервера.

Имя функции: sendMail

Права доступа: Доступно на уровне 486 прав доступа *Оператор*

Записи входа: 1

Формат 50 входа:

Имя	Тип	Описание
recipients	String	Список получателей email, отделенных запятыми.
subject	String	Тема сообщения.
message	String	Текст сообщения.
html	Boolean	Показывает, что этот текст сообщения в формате HTML.
attachments	DataTable	Таблица данных с двумя полями: имя (Строка) и данные (Данные), содержит список прикреплений.

Записи выхода: 0

Формат 50 выхода: нет

16.5 SMS сообщения

Этот раздел описывает, как отправлять SMS сообщения, используя AtomMind Server. Например, SMS могут передаваться при возникновении тревог 779.

Возможность управления SMS предоставляется в отдельном плагине 207 SMS. Конфигурация плагина описана здесь 1437.

Сообщения отправляются через SMS шлюз Clickatell. Clickatell не является бесплатным, но имеет ряд преимуществ:

- Высоконадежная передача SMS сообщений в большинство стран мира
- Отсутствие необходимости подключать сотовый телефон к вашему серверу
- Отсутствие необходимости использовать внешнее ПО
- Простая конфигурация

ТВЭЛ никак не связан с Clickatell. Clickatell не перечисляет ТВЭЛ никакие денежные средства или вознаграждение и не является ни в коей мере клиентом или партнером ТВЭЛ. Основания рекомендации использования данного шлюза носят чисто технический характер.

Если вы используете язык, отличный от английского, необходимо писать сообщения с использованием Юникод UTF-16 для обеспечения правильного отображения символов.

Чтобы начать использовать сервис по отправке SMS Clickatell, необходимо:

1. Зарегистрироваться на http://www.clickatell.com.
2. Авторизоваться в Clickatell Central.
3. Перейти на **Manage My Products**.
4. Добавить **HTTP/S** подключение.
5. Ввести только имя подключения и оставить все настройки по умолчанию (вы можете захотеть изменить некоторые настройки под свои нужды, например, зафиксировать IP, с которого принимаются SMS сообщения).
6. Введите ваши Clickatell настройки в конфигурацию отправки SMS AtomMind Server и активируйте отправку SMS.

16.5.1 Отправка SMS

Опции в данной группе используются для определения, как AtomMind Server отправляет SMS сообщения.

Разрешить отправку SMS

Имя ключа в файле конфигурации: smsSendingEnabled

Тип значения: Boolean

Возможные значения: true или false

Значение по умолчанию: false

Позволяет AtomMind Server отправлять SMS сообщения.

Имя пользователя аккаунта Clickatell

Имя ключа в файле конфигурации: smsClickatellUsername

Тип значения: String

Возможные значения: Любая строка

Значение по умолчанию: "" (пустое)

Определяет имя пользователя Clickatell. Должно совпадать с Вашим логином на сервере Clickatell.

Пароль аккаунта Clickatell

Имя ключа в файле конфигурации: smsClickatellPassword

Тип значения: String

Возможные значения: Любая строка

Значение по умолчанию: "" (пустое)

Определяет пользовательский пароль на сервере Clickatell. Должно соответствовать вашему паролю на сервере Clickatell.

API ID аккаунта Clickatell

Имя ключа в файле конфигурации: smsClickatellApiId

Тип значения: String

Возможные значения: Любая строка

Значение по умолчанию: "" (пустое)

Указывает идентификатор API Вашего соединения, созданного на сервере Clickatell. Соединение должно использовать формат **HTTP/S**.

Убедитесь, что указываете идентификатор соединения (числовое), а не имя соединения.

Телефон отправителя (В международном формате)

Имя ключа в файле конфигурации: smsSender

Тип значения: String

Возможные значения: Любой корректный номер сотового телефона

Значение по умолчанию: "" (пустое)

Номер телефона-отправителя SMS сообщений. Номер должен быть зарегистрирован на Clickatell.

Номер должен начинаться с "+".

16.5.2 Дополнительные настройки SMS

Таблица расширенных настроек SMS позволяет точно настроить параметры отправки/получения SMS сообщений. Каждый параметр определяется через **Имя** и **Значение**.

16.5.3 Ссылка на контексты SMS

Модуль SMS добавляет следующие сущности к дереву контекстов 1450 AtomMind Server .

Корневой контекст

Следующие действия 87 добавляются к корневому 1559 контексту:

ОТПРАВИТЬ SMS

Вызов функции 103 дает команду AtomMind Server на отправку SMS сообщения.

Тип действия: Вызов функции 103

Имя действия: sendSms

Группа действия: Дополнительные действия

Права доступа: Доступно при уровне 486 прав доступа *Администратор*

Контекст утилит

Следующие функции 70 добавляются к контексту утилит 1613:

ОТПРАВИТЬ SMS

Отправляет SMS сообщение. Эта функция будет работать правильно только если отправка SMS 1437 должным образом установлена в настройках сервера.

Имя функции: sendSms

Права доступа: Доступ но при уровне 486 прав доступа *Оператор*

Входные записи: 1

Входной формат 50:

Имя	Тип	Описание
recipient	String	Номер телефона получателя в международном формате.
message	String	Текст сообщения.

Выходные записи: 0

Выходной формат 50: None

16.6 Геозоны

Данная функция позволяет определять и работать с геозонами, которые, по сути, являются секциями или наборами точек географической карты.

Функционал Геозон обеспечивается специальной моделью 810. Модель доступна для установки из ресурсов 208. Модель Геозона является экземплярной, каждый экземпляр может быть одного из следующих типов:

- Точки
- Путь
- Полигон

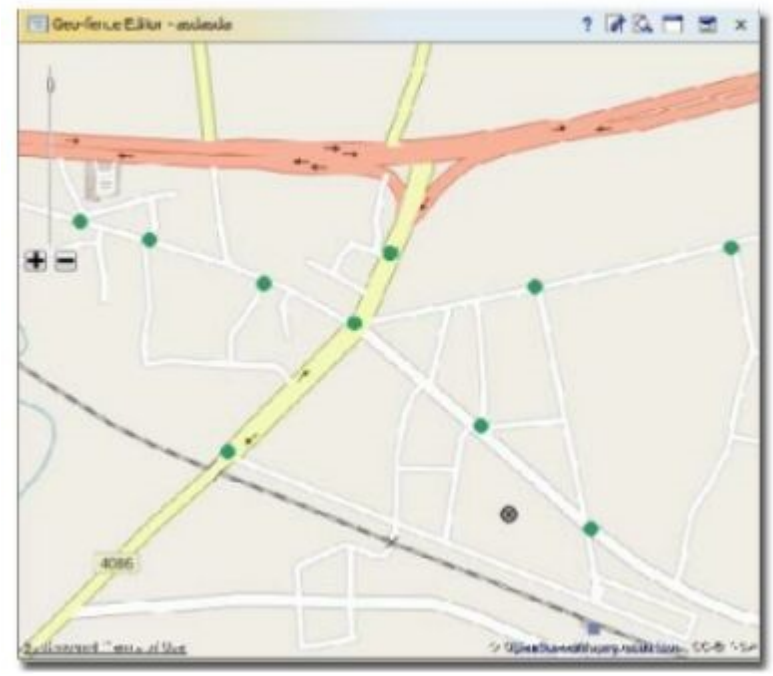
Каждый экземпляр Геозоны состоит из нескольких геоточек, заданных широтой и долготой.

Модуль также предоставляет несколько функций для работы с геозонами, например, для проверки, входит ли определенная точка в геозону и т.д. Эти функции описаны в разделе Справочник по контексту геозоны[1440].

Создавать и управлять геозонами можно в Редакторе геозон[1440].

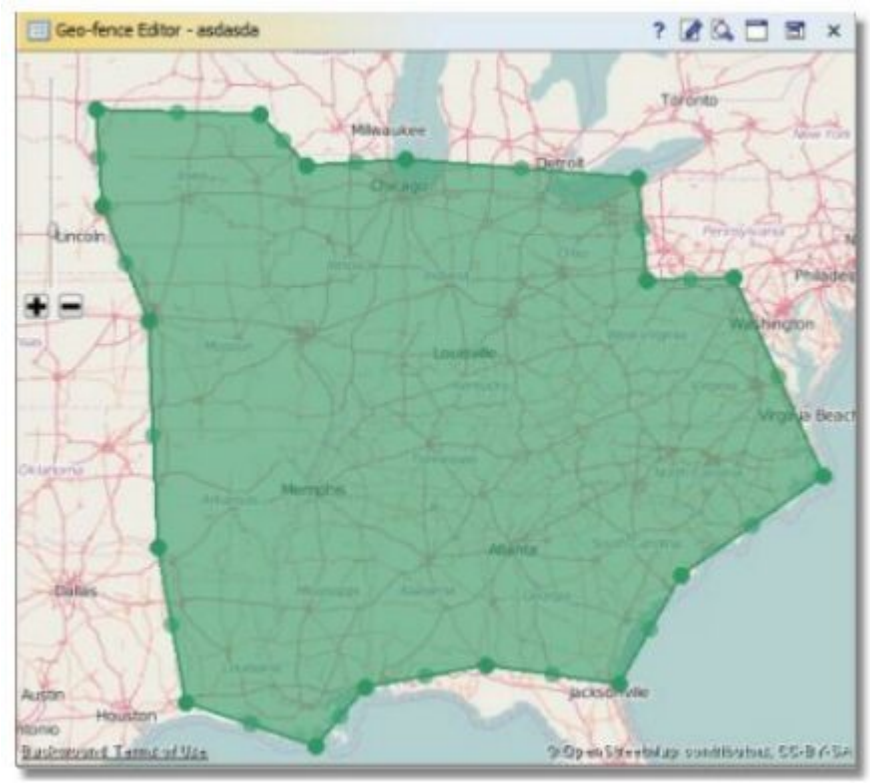
ПРИМЕРЫ ГЕОЗОН:

ТОЧКИ:



ПУТЬ:

ПОЛИГОН:



16.6.1 Редактор геозон

Редактор Геозон - это преднастроенный виджет[943], который помогает создавать геозоны[1438], определяя из точки на карте. Данный виджет позволяет добавлять новые точки Геозоны, перемещать и удалять их.

Для редактирования Геозоны, нажмите правой кнопкой мыши по контексту Геозона и выберите Редактор Геозон в меню.

Добавление и удаление точек

Редактор Геозон очень прост в использовании:

- Кликните левой кнопкой мыши по карте, чтобы **добавить** новую геоточку
- Перетащите и отпустите точку, чтобы **переместить** ее
- Для **удаления** геоточки, кликните по ней правой кнопкой мыши выберите **Удалить точку** в контекстном меню.

16.6.2 Справочник по контексту геозоны

Данный контекст[41] имеет такие же действия, как и контекст модели[1539], за исключением дополнительных функций:

Общие переменные (свойства) [?[61]]

СВОЙСТВА

Настройка определяет базовые свойства геозоны.

Поле	Тип	Описание
Name	String	Имя геозоны.
Description	String	Описание геозоны.

ПАРАМЕТРЫ

Настройка определяет параметры геозоны.

Поле	Тип	Описание
Type	String	Список доступных типов: Полигон, Точки, Путь
Editable	Boolean	Определяет, редактируется ли геозона

ТОЧКИ

Таблица определяет координаты и параметры точек геозоны.

Поле	Тип	Описание
Latitude	Double	Широта - это географическая координата, определяющая положение точки на поверхности Земли по оси север-юг.
Longitude	Double	Долгота - это географическая координата, определяющая положение точки на поверхности Земли по оси восток-запад.
Description	String	Описание геоточки.

Общие функции

ПОВЕРНУТЬ

Поворачивает выбранные геозоны.

Имя функции: rotate

Права доступа: Доступно на уровне 486 прав доступа *Оператор*

Записи ввода: 1

Формат 50 ввода:

Имя	Тип	Описание
Theta	Double	Угол поворота.

Записи вывода: 1

Формат 50 вывода:

Имя	Тип	Описание
success	Boolean	True, если поворот завершен удачно. Иначе false.

ДВИГАТЬ

Двигает выбранные геозоны.

Имя функции: move

Права доступа: Доступно на уровне 486 прав доступа *Оператор*

Записи ввода: 1

Формат 50 ввода:

Имя	Тип	Описание
T X	Double	Значение оси X.
T Y	Double	Значение оси Y.

Записи вывода: 1

Формат 50 вывода:

Имя	Тип	Описание

success	Boolean	True, если передвижение выполнено удачно. Иначе false.

РАССТОЯНИЕ

Возвращает расстояние от точки до текущей геозоны.

Имя функции: distance

Права доступа: Доступно на уровне 486 прав доступа *Оператор*

Записи ввода: 1

Формат 50 ввода:

Имя	Тип	Описание
Latitude	Double	Широта точки, от которой вычисляется дистанция.
Longitude	Double	Долгота точки, от которой вычисляется дистанция.

Записи вывода: 1

Формат 50 вывода:

Имя	Тип	Описание
Distance	Double	Расстояние до геозоны.

ПЕРЕСЕКАЕТ

Возвращает true, если выбранная геозона пересекается с текущей геозоной.

Имя функции: intersect

Права доступа: Доступно на уровне 486 прав доступа *Оператор*

Записи ввода: 1

Формат 50 ввода:

Имя	Тип	Описание
Geo-fence Context	String	Контекст геозоны, который будет проверен на пересечения.
Longitude	Double	Долгота точки проверяется относительно включенного в геозону.

Записи вывода: 1

Формат 50 вывода:

Имя	Тип	Описание
Intersect	Boolean	Пересекающиеся геозоны.

МАСШТАБИРОВАТЬ

Масштабирует выбранную геозону.

Имя функции: scale

Права доступа: Доступно на уровне 486 прав доступа *Оператор*

Записи ввода: 1

Формат 50 ввода:

Имя	Тип	Описание
Scale	Double	Фактор масштабирования.

Записи вывода: 1

Формат 50 вывода:

Имя	Тип	Описание
success	Boolean	True, если масштабирование закончилось успешно. Иначе false.

ВКЛЮЧАЕТ В СЕБЯ

Возвращает true, если точка включена в текущую геозону.

Имя функции: includes

Права доступа: Доступно на уровне 486 прав доступа *Оператор*

Записи ввода: 1

Формат 50 ввода:

Имя	Тип	Описание
Latitude	Double	Широта точки проверяется на нахождение в геозоне.
Longitude	Double	Долгота точки проверяется на нахождение в геозоне.

Записи вывода: 1

Формат 50 вывода:

Имя	Тип	Описание
Includes	Boolean	Входит ли точка в геозону.

ОТРАЗИТЬ

Отразить выбранную геозону.

Имя функции: flip

Права доступа: Доступно на уровне 486 прав доступа *Оператор*

Записи ввода: 1

Формат 50 ввода:

Имя	Тип	Описание
Flip X	Boolean	Отразить геозону горизонтально.
Flip Y	Boolean	Отразить геозону вертикально.

Записи вывода: 1

Формат 50 вывода:

Имя	Тип	Описание

success	Boolean	True, если отражение завершилось успешно. Иначе false.

16.7 Механизм параметризации

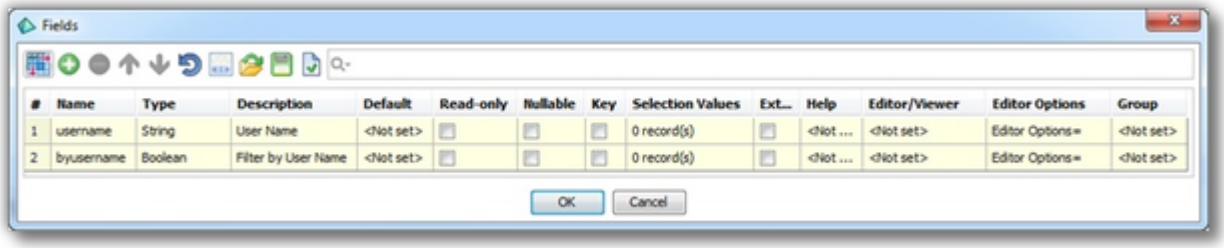
Параметризация представляет собой процесс построения запроса[829], выражения фильтра событий[762] или выражения исходных данных отчета[929], который предложит пользователю различные параметры при его выполнении. Как только вы завершили процесс, вы получаете простую строку - часть текста вы можете использовать в качестве текста для запроса или выражения фильтра событий.

Исходные данные для процессора параметризации заданы в формате XML. Он определяет количество параметров которые должны быть введены. Обычно это происходит следующим образом: администраторы создают исходные данные параметризации (XML) и вставляют их в параметризированный запрос или фильтр событий. Пользователи могут уточнять параметры на лету при выполнении запроса или фильтра.

Формат параметров

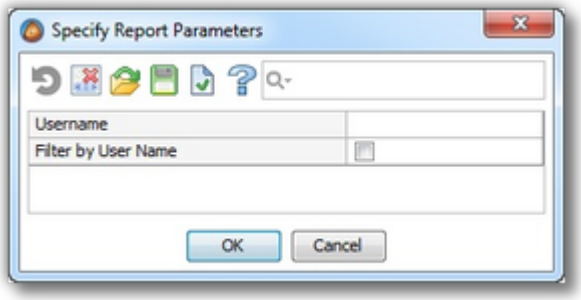
Формат является форматом[49] Таблицы данных[49], которая создается и отображается в режиме редактирования, чтобы пользователь мог изменять поля. Значения этих полей будут конвертированы в строки и вставлены в **Выражение для параметризации**. Таблица данных, построенная из **Формата**, всегда содержит только одну запись. Данная таблица позволяет пользователю интерактивно задаввать значения параметров, которые будут в дальнейшем использованы в **Выражении для параметризации**.

Далее приведен пример таблицы полей **Формат**:

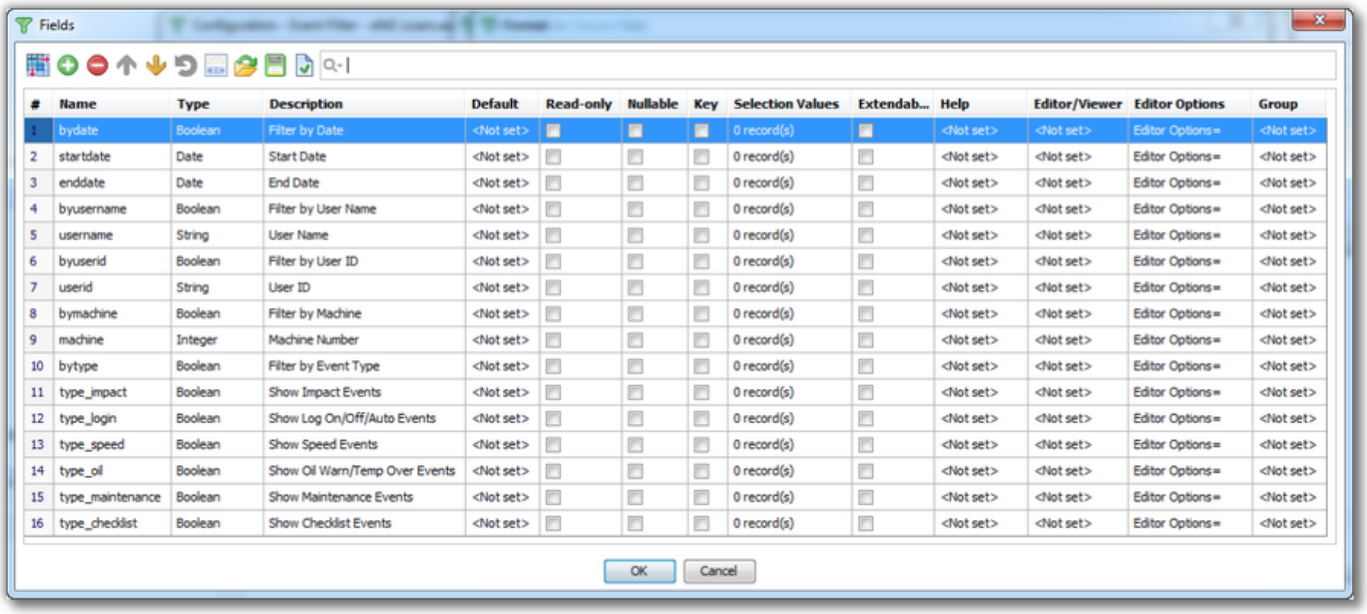


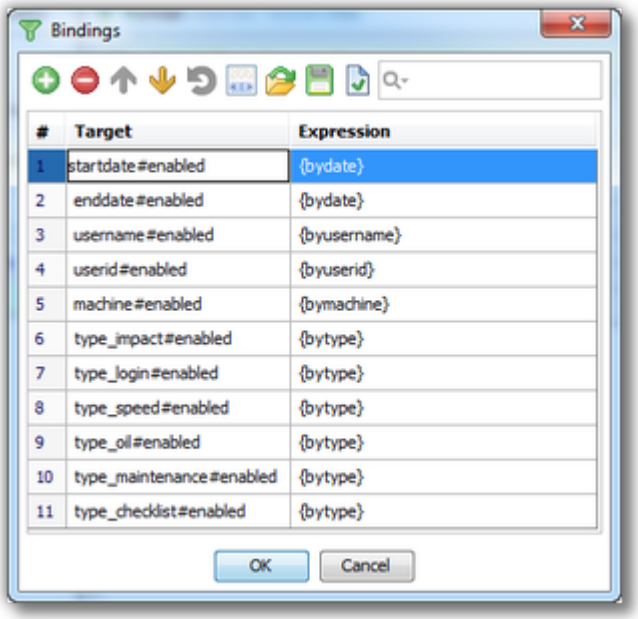
Данный пример определяет два поля: поле типа Boolean под названием "byusername" (с удобным для чтения описанием, "Фильтрация по имени пользователя") и строку "username" ("Имя пользователя"). Первый параметр может использоваться, к примеру, для включения фильтрации определенным пользователем в отчете, второй параметр – это имя пользователя.

Пользователю дается подсказка для ввода данных параметра (скриншот, сделанный в AtomMind Client):



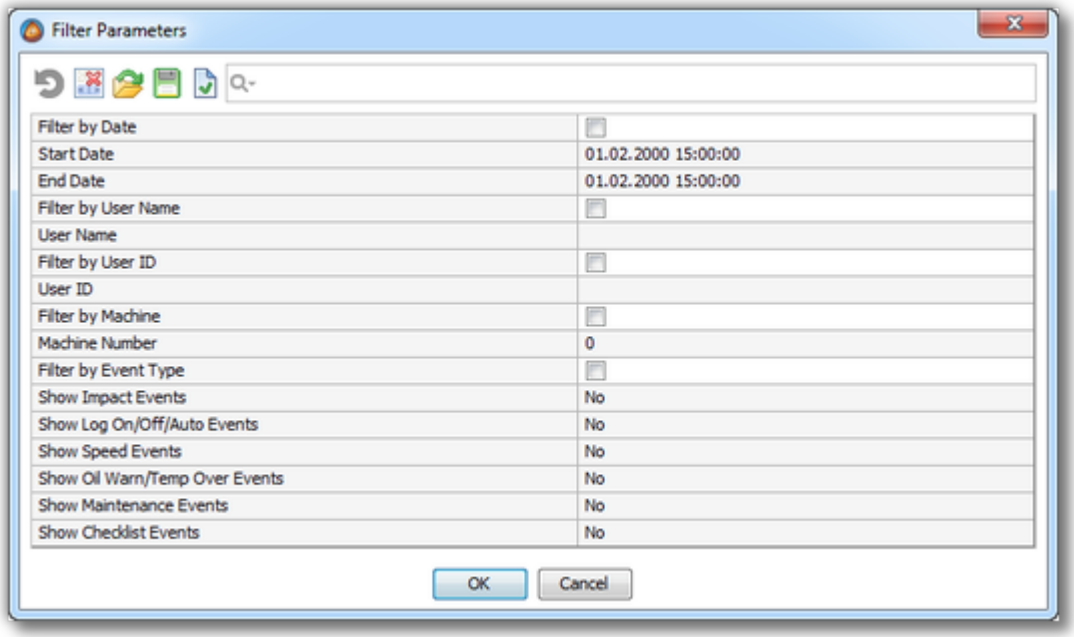
Здесь намного более сложный формат, который включает в себя даже привязки[741].





Этот формат определяет множество полей и некоторые отношения между ними, контролируемые привязками. Например, привязка "`username#enabled={byusername}`" делает так, чтобы пользователь мог заполнить лишь параметр "Имя пользователя" ("username"), если включена фильтрация по имени пользователя (т.е., поле "byusername" выставлено как TRUE).

При запуске фильтра, использующего этот формат, пользователю дается подсказка для ввода данных параметра, используя следующий диалог (скриншот, сделанный в AtomMind Client):



Выражение для параметризации

Выражение для параметризации представляет собой всего лишь текст, необходимый для формирования запроса 829 или Выражения AtomMind, используемых фильтром событий 762 или отчетом 928. Текст может содержать две специальные конструкции, которые зависят от значений параметров, определенных полями **Формата**.

Так как **Выражение для параметризации** содержит теги XML (см. далее), оно не может быть записано в блоке CDATA. Из этого следует, что символы, используемые в синтаксисе XML должны быть экранированы (escaped). Например, если в итоговом выражении должен появиться символ &, он должен быть записан как `&amp` в **Выражении для параметризации**.

УСЛОВНЫЙ АБЗАЦ

Условный абзац определяет блок текста, который вставляется в **параметризованное выражение**, если выражение в его заголовке имеет результат TRUE. Данное выражение может содержать ссылки на различные поля в **Формате**.

Синтаксис:

```
<p enabled="ENABLING_CONDITION_EXPRESSION">TEXT_OF_CONDITIONAL_PARAGRAPH</p>
```

ENABLING_CONDITION_EXPRESSION является выражением AtomMind 112, которое может содержать ссылки на ячейки Таблиц данных, построенных из PARAMETERS_FORMAT. Форматом ссылок является `{FIELD_NAME}`, где FIELD_NAME - имя поля, определенного через PARAMETERS_FORMAT. Более подробную информацию о ссылках см. здесь 117.

Пример ENABLING_CONDITION_EXPRESSION:

```
{filterByName} && {filterByValue}
```

Пример of EXPRESSION_TO_PARAMETERIZE с условным абзацем:

```
SELECT * FROM users.*:childInfo <p enabled="{showOnlyUsersWithPhones}">WHERE
childInfo$phone IS NOT NULL</p>
```

В данном примере, `childInfo$phone` ссылается на поле "phone number" в записи информации о пользователе (см. Запросы 829). `ENABLING_CONDITION_EXPRESSION` ссылается на поле типа Boolean `showOnlyUsersWithPhones`, определенное в PARAMETERS_FORMAT (например, так: "`<<showOnlyUsersWithPhones><B><D=Show only users that have phone numbers>>`"). Если это поле установлено пользователем на FALSE во время процесса параметризации, полученным выражением будет "`SELECT * FROM users.*:childInfo`". Если это поле установлено на TRUE, итоговым выражением будет "`SELECT * FROM users.*:childInfo WHERE childInfo$phone IS NOT NULL`", т.е. TEXT_OF_CONDITIONAL_PARAGRAPH включен в результат.

Условный абзац не должен обязательно быть в конце строки запроса/фильтра. В следующем примере, он расположен в середине выражения (прямо перед оператором ORDER BY): `SELECT * FROM users.*:childInfo <p enabled="{showOnlyUsersWithPhones}">WHERE childInfo$phone IS NOT NULL</p> ORDER BY childInfo$name DESC`

ВЫРАЖЕНИЯ

Как было сказано выше, **Выражение для параметризации** является просто строкой. Само по себе оно не является Выражением AtomMind. Однако, оно может *включать* Выражения AtomMind. Такие выражения AtomMind могут содержать в себе ссылки на ячейки Таблиц данных, простроенных из **Формата**. Выражение оценивается, результат оценки конвертируется в строку и вставляется в конечный результат параметризации (выражение запроса или фильтра AtomMind, которое образуется в ходе процесса параметризации).

Синтаксис:

```
<e>EXPRESSION</e>
```

Данное выражение может содержать ссылки на ячейки Таблицы данных, построенной из **Формата** так же, как и ENABLING_CONDITION_EXPRESSION условного параграфа (см. выше).

Пример:

```
SELECT * FROM users.*:childInfo WHERE childInfo$phone = '<e>{phone}</e>'
```

Метки ' сверху необходимы, т.к. они используются для экранирования строкового литерала SQL.

The ' marks above are necessary - they're used to escape the SQL string literal.

This example assumes a "phone" field exists in PARAMETERS_FORMAT. For example, if a user has entered phone "123-45-67" during the parameterization process, the result of parameterization will be "`SELECT * FROM users.*:childInfo WHERE childInfo$phone = '123-45-67'`".

РАЗЛИЧИЕ

1) **Условные абзацы** позволяют вам добавлять/удалять различные части итогового выражения, основанные на результате да/нет (Boolean) какого-либо выражения.

2) **Выражения** позволяют вам вставлять специальные значения, полученные в результате вычислений, в ваш итоговый резльтат параметризации.

Поток процесса параметризации

1. Выполняется анализ и проверка пригодности данных для параметризации.
2. Пользователю предлагается ввести параметры, заданные через **Формат**.
3. Полученная Таблица данных с параметрами, введенными оператором вручную, сохранятся и доступна для использования в качестве, например, таблицы по умолчанию для расчета значений дополнительных параметров отчета 936.
4. Теперь изменяется **Выражение для параметризации** с применением параметров из шага (2): текст условных абзацев добавляется или даляется, выражения оцениваются, а результат оценки вставляется в конечную строку.
5. Полученная строка используется в качестве выражения запроса или фильтра.

16.8 Контекст хранилища

Контекст хранилища – это контекст 41 AtomMind Server, который предоставляет методы просмотра, выбора, создания, обновления, удаления, сортировки, поиска и фильтрации больших групп сходных объектов, таких как записи в таблицах реляционных баз данных, экземпляры класса 885 или объекты сервера, индексируемые поисковой машиной AtomMind Server.

Далее приведено несколько примеров контекстов, которые могут быть выбраны в качестве Контекста Хранилища:

- Контекст устройства 1494, управляемый Драйвером устройства базы данных 647. Когда он действует как контекст хранилища, он позволяет управлять записями в любой таблице во внешних БД.
- Контекст класса 1473. Позволяет управлять экземплярами определенного класса 885.
- Контекст утилит 1613. Позволяет управлять записями в любой таблице базы данных 692 AtomMind Server.

16.9 Подключение коммуникаций через последовательн

Виртуальная машина Java, которая запускает AtomMind Server, требует специальный драйвер **Java Communication API** для взаимодействия с последовательными устройствами (RS-232, RS-482 и т.д.). Данный драйвер устанавливается автоматически при помощи Установщика AtomMind Server для ОС Microsoft Windows. Для других ОС, драйвер Java Communication API нужно устанавливать вручную, чтобы получить доступ к последовательным устройствам

Если драйвер Java Communication API установлен неправильно, Настройки подключения последовательного устройства не будут включать список последовательных портов системы.

Чтобы установить Java Communication API на вашу ОС, следуйте приведенным далее инструкциям:

1. Скачайте Java Communication API для вашей ОС с сайта Sun Java. На момент создания данного руководства, он был доступен по адресу: http://java.sun.com/products/javacomm/.

2. Следуйте инструкциям пакета дистрибутива драйвера.

По умолчанию Виртуальная машина Java, которая запускает AtomMind Server, расположена в субдиректории `/jre` установки AtomMind Server.

Заметка по Linux

Если ваш сервер установлен на машине Linux, в AtomMind будут видны только порты, которые следуют стандартным конвенциям о наименовании (`/dev/ttyS*`).

Если у вашего порта такое имя, которое не соответствует конвенциям, появится ошибка `No such port` или `gnu.io.NoSuchPortException`.

Чтобы справиться с этим, создайте символическую ссылку на него, например:

```
# ln -s /dev/tty200 /dev/ttyS200
# updatedb
```

Необходимо перезапустить AtomMind Server после создания символической ссылки, чтобы новый порт появился в списке последовательных портов. В ином случае имя нового порта можно вручную внести в настройки устройства.

16.10 Net Admin

Администратор сети *(Net Admin)* - это функция, используемая для осуществления контроля за сервером с локального хоста без авторизации. Функцию *Net Admin* можно использовать даже тогда, когда невозможно установить соединение с сервером через AtomMind Client, даже когда нет доступа к его Web UI 220 и Меню в системном трее 173 (т.е. на системе без графического пользовательского интерфейса). Ее обычно используют для отладки AtomMind Server, если другие методы (AtomMind Client 359, Web UI 220) не работают по какой-либо причине.

Доступ к функции *Net Admin* осуществляется при помощи Telnet или иного аналогичного программного обеспечения. Доступ должен производится с того PC, на котором запущен AtomMind Server. Невозможно получить доступ к Net Admin с других хостов.

Чтобы подключиться к *Net Admin*:

- убедитесь, что Вы используете сервер, на котором запущен AtomMind Server.
- Зайдите по `telnet` на `127.0.0.1` (`localhost`), порт `6440`. Должно установиться соединение без приглашения для ввода команд.
- Введите команду Net Admin и нажмите Enter. Команды Админа Сети описаны в главе Список команд Net Admin и скрипты запуска 1449.
- Если команда выполнена *успешно*, Вы получите подтверждение `A` (Ack) и, возможно, текстовое сообщение-ответ.
- Если команда выполнена *неуспешно*, Вы получите сообщение об ошибке `E` (Error) и, возможно, описание ошибки.

Интерфейс *Net Admin активируется* по умолчанию. Он используется во время атоматического обновления или деинсталляции AtomMind Server. Если нежелательные пользователи имеют локальный доступ

(физический или через оболочку Unix) к хосту с AtomMind Server, эту функцию следует отключить из соображений безопасности. Ее можно отключить, используя интерфейс *Web Admin*.

16.10.1 Список команд и скрипты запуска

Этот раздел содержит информацию о командах Net Admin 1448 и обработчика Скриптов запуска 1651.

Аргументы команд разделяются символом "**/**". Каждая строка в скрипте запуска может содержать лишь одну команду.

Выполнение команд Net Admin и Startup Script, которые изменяют у сервера статус контекстного дерева, может привести его к нестабильному состоянию или повредить БД. Перед выполнением команд следует провести резервное копирование базы данных.

ПОДДЕРЖИВАЕМЫЕ КОМАНДЫ

E	Остановить сервер
R	Перезапустить сервер
W	Перезапустить веб-сервер внутри AtomMind Server
S/*context_mask* /*variable* /*value_for_row1_column1/value_for_row1_column2* /...	Установить значение для переменной контекста 41. Устанавливает значение переменной '*variable*' в каждом контексте, относящемуся к 44 '*context_mask*', созданной таблицы данных 49 из следующих параметров команды.
C/context_mask/function/*value_for_row1_column1/* *value_for_row1_column2/...*	Вызывает функцию '*function*' из каждого контекста, соответствующего 44 '*context_mask*', передающей таблицу данных, созданную 1449 из следующих параметров, как входное значение.
F/context_mask/variable/setField/setValue[/searchField/searchValue]	Установить поле '*setField*' первой записи значения переменной '*variable*' в каждом контексте, относящемся 44 к '*context_mask*' для '*setValue*'. Если параметры '*searchField*' и '*searchValue*' установлены, значение в записях изменяется там, где значение поля '*searchField*' равно '*searchValue*'.

Создание таблицы данных при помощи аргументов команды

Во время обработки некоторых команд (Set Variable, Call Function и пр.) Таблица данных 49 создаётся из списка арументов команд, как описано здесь.

Если значение аргумента окружено символами "**$**", AtomMind Server возьмет данные для этого параметра из указанного имени файла. Файл должен находиться в директории, где установлен AtomMind Server. Например, если значение параметра "`$parameterizer.xml$`", AtomMind Server получит данные для этого параметра из "`parameterizer.xml`".

17 Справочник по контекстам

Эта глава содержит подробное описание всех контекстов[41], которые доступны в AtomMind Server.

Описание каждого контекста включает следующие части:

Обзор	Общая информация о контексте
Уникальные действия	Это действия[87], которые применимы лишь к данному контексту.
Общие действия	Это действия[87], которые применимы ко многим различным контекстам в AtomMind Server.
Информация о контексте	Дополнительная информация о контексте.
Состояния и иконки контекста	Доступные состояния[44] контекста и их визуальная репрезентация.
Общие переменные (свойства)	Переменные[61] контекста для непосредственного применения пользователями.
Общие функции	Функции[70] контекста для непосредственного применения пользователями.
Общие события	События[73] контекста для непосредственного применения пользователями.

17.1 Администрирование

Это системный контекст, который предоставляет доступ к различным данным, связанным со всей работой AtomMind Server. Он не отображается в видимой части контекстного дерева.

Дополнительная информация

Информация о контексте

Тип контекста[43]: администрирование

Имя контекста[42]: administration

Описание контекста[43]: Administration

Путь контекста[42]: administration

Контекстная маска[44]: administration

Права доступа к контексту [?[44]]

Уровень	Описание
Не определен	Нет доступа.
Наблюдатель	Нет доступа.
Оператор	Нет доступа.
Менеджер	Нет доступа.
Инженер	Нет доступа.
Администратор	Все операции.

Общие переменные (свойства) [?[61]]

У этого контекста нет открытых переменных (свойств).

Общие функции [?70]

У этого контекста нет открытых функций.

Общие события [?73]

Общие события: info (Information)

ЖУРНАЛ

Появляется, когда к журналу событий166 сервера добавляются сообщения на уровне171 *инфо* или более высоком уровне.

Имя события: log

Права доступа: Доступны на уровне486 прав доступа *Администратора*

Период действия: непостоянный

Записи: 1

Формат50 записи:

Имя поля	Тип поля	Комментарии
level	String	Уровень сообщения.
message	String	Текст сообщения.
cause	String	Причина сообщения (трассировка стека исключений).
thread	String	Server thread that produced the message.
location	String	Номер файла и строки линии источника сообщения.

ДЕЙСТВИЕ

Появляется, когда системный оператор478 выполняет действие87 или действие автоматически выполняется одним из серверных компонентов (например, по расписанию823 или при появлении тревоги779).

Имя события: action

Права доступа: Доступны на уровне486 прав доступа *Администратора*

Период действия: 100 дней

Записи: 1

Формат50 записи:

Имя поля	Тип поля	Комментарии
user	String	Имя пользователя системного оператора.
login	String	Имя логина системного оператора.
mode	Integer	Режим выполнения. Различаются следующие режимы: • **Нормальный**. Выполнение инициируется оператором. • **Переадресация**. Это действие инициируется другим действием. • **Пакет**. Несколько действий одновременно начинают выполняться.

		• **Автоматический**. Общий автоматический режим выполнения, т.е. когда действие выполняется под контролем серверного плагина[207]. • **Тревога**. Режим выполнения корректирующих действий при появлении тревоги. • **Планировщик**. Режим выполнения по расписанию.
context	String	Путь контекста[41], в котором выполнялось действие.
name	String	Название действия.
description	String	Описание действия.

EMAIL

Появляется, когда AtomMind Server получает входящее email-сообщение.

Имя события: email

Права доступа: Доступны на уровне[486] прав доступа *Администратора*

Период действия: 100 дней

Записи: 1

Формат[50] записи:

Название поля	Тип поля	Комментарии
sender	String	Адрес отправителя.
subject	String	Тема сообщения.
text	String	Текст сообщения.

ОТКАЗОУСТОЙЧИВОСТЬ

Это событие появляется, когда начинает работать подчиненный узел отказоустойчивый кластер[1326] в случае отказа главного узла.

Имя события: failover

Права доступа: Доступен на уровне[486] прав доступа *Администратора*

Период действия: 100 дней

Записи: 1

Формат[50] записи: отсутствует.

17.2 Тревоги

Этот контекст[41] является контейнером, которые содержит все тревоги[779] для отдельного пользователя.

Уникальные действия [?[1450]]

СОЗДАТЬ НОВУЮ ТРЕВОГУ (действие по умолчанию[88])

Это действие используется для создания новой тревоги. Оно позволяет пользователю задать основные свойства[795] для новой тревоги и настроить ее сразу же после создания.

Тип действия: создать 105

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*.

Общие действия [? 1450]

Создать на основе шаблона 105, копировать в дочерние контексты 111, импорт 108, экспорт 108, редактировать права доступа 106, просмотр событий 109, поиск/фильтрация 110, различные групповые действия 101 согласно контекстам потомков.

Действия над переменными [? 102]

СОЗДАТЬ ТРЕВОГУ

Это действие создает новую тревогу для переменной. Во-первых, оно позволяет пользователю выбрать тип тревоги:

- **Отслеживание значения переменной**. Тревога появляется, если значение переменной не соответствует определенному критерию. Пользователь может выбрать, какое поле с переменной будет проверяться выражением триггера, и указать для них операции сравнения.
- **Предстоящее нарушение договора об уровне обслуживания**. Тревога возникает заранее, если значение статистической переменной может вскоре приблизиться к числовому порогу в договоре об уровне обслуживания.

Имя действия: createForVariable

Не интерактивный режим 99: не поддерживается

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*.

Действия, связанные с событием [? 102]

СОЗДАТЬ ТРЕВОГУ

Это действие создает новую тревогу с одним триггером события 787, который инициирует тревогу, если происходит событие данного типа. Система предлагает пользователю выбрать, какие поля события будут проверяться выражением триггера, и задать для них операторов сравнения. Если поля не выбраны, любое событие будет вызывать тревогу.

Имя действия: createForEvent

Не интерактивный режим 99: не поддерживается

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*.

Состояния и иконки контекста

У данного контекста нет состояний 44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста 43: тревоги

Имя контекста 42: alerts

Описание контекста 43: тревоги

Путь контекста 42: users.USER_NAME.alerts

Контекстная маска 44: users.*.alerts

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий.
Оператор	То же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт тревог.
Инженер	То же, что у Менеджера.
Администратор	То же, что у Менеджера.

Общие переменные (свойства) [?61]

У данного контекста нет общих переменных (свойств).

Общие функции [?70]

Общие функции: makeCopy (скопировать)71, delete (удалить)72

СОЗДАТЬ

Создает новую тревогу.

Имя функции: create

Права доступа: Доступно на уровне486 с правами доступа для *Менеджера*

Записи ввода: 1

Формат49 ввода: Аналогичный формату переменной childInfo1455 в контексте Тревога1454.

Записи вывода: 0

Формат49 вывода: нет

Общие события [?73]

Общие события: info (информация)77

17.3 Тревога

Данный контекст41 предоставляет доступ к одной тревоге779 и позволяет ею управлять.

Уникальные действия [?1450]

НАСТРОИТЬ ТРЕВОГУ (действие по умолчанию88)

Это действие используется для редактирования свойств795 тревоги.

> Изменение поля **Имя** во время этой операции меняет название данного контекста. Это может привести к сбою в работе других компонентов системы, использующих имя/путь контекста в качестве основного идентификатора.

Тип действия: настроить105

УПРАВЛЕНИЕ НЕПОДТВЕРЖДЕННЫМИ ТРЕВОГАМИ

Отображает список ожидающих804 тревог при помощи GUI процедуры Управление неподтвержденными тревогами96.

Имя действия: pendingAlerts

Не интерактивный режим: не поддерживается

Права доступа: Доступно на уровне с правами доступа для *Наблюдателя*.

Общие действия [?]

Удалить, создать копию, реплицировать, редактировать права доступа, просмотр событий, показать статус

Состояния и иконки контекста

Иконка	Код	Состояние
	0	Тревога активирована
	1	Тревога деактивирована
	2	Тревога активна
	3	Тревога эскалирована

Дополнительная информация

Информация о контексте

Тип контекста: тревога

Имя контекста: предоставляется пользователем

Описание контекста: предоставляется пользователем

Путь контекста: users.USER_NAME.common.ALERT_NAME

Контекстная маска: users.*.alerts.*

Права доступа к контексту [?]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Получение тревоги. Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Конфигурация и удаление тревоги.
Инженер	То же, что у Менеджера.
Администратор	То же, что у Менеджера.

Общие переменные (свойства) [?]

Общие переменные: groupMembership (членство в группе), activeAlerts (активные тревоги)

СВОЙСТВА ТРЕВОГИ

См. описание переменной и ее полей здесь.

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи для *Менеджера*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
name	Строка	1-50 знаков
description	Строка	1-50 знаков
enabled	Булевое	
message	Строка	Поле может не иметь значения (определено как "nullable")
permissions	Строка	
persistState	Булевое	

ТРИГГЕРЫ, АКТИВИРУЕМЫЕ СОБЫТИЯМИ

См описание переменной и ее полей здесь 796.

Имя переменной: eventTriggers

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи для *Менеджера*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
mask	Строка	
event	Строка	
filter	Строка	
level	Целое	
correlated	Строка	
correlator	Строка	
message	Строка	

ТРИГГЕРЫ, АКТИВИРОВАННЫЕ СОСТОЯНИЕМ

См. описание переменной и ее полей здесь 797.

Имя переменной: variableTriggers

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи для *Менеджера.*

Формат 49 записи:

Имя поля	Тип поля	Примечания
mask	Строка	
variable	Строка	
filter	Строка	
monitorStateChange	Булевое	
period	Длинное	Измеряется в миллисекундах
delay	Длинное	Измеряется в миллисекундах
level	Целое	
deactivator	Строка	
deactivationDelay	Длинное	Измеряется в миллисекундах
message	Строка	

ОПОВЕЩЕНИЕ О ТРЕВОГЕ

См. описание переменной и ее полей здесь 799.

Имя переменной: notifications

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи для *Менеджера.*

Формат 49 записи:

Имя поля	Тип поля	Примечания
notifyOwner	Булевое	
ackRequired	Булевое	
sound	Блок данных	Редактируется звуковым редактором.
mailToOwner	Булевое	
mailRecipients	Таблица данных	
additionalMailRecipients	Строка	
smsRecipients	Таблица данных	

ЭСКАЛАЦИЯ ТРЕВОГИ

См. описание переменной и ее полей здесь 799.

Имя переменной: escalation

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи для *Менеджера.*

Формат 49 записи:

Имя поля	Тип поля	Примечания
pending	Булевое	
numberEscalation	Булевое	
numberThreshold	Целое	
timeEscalation	Булевое	
timeThreshold	Целое	

АВТОМАТИЧЕСКИЕ КОРРЕКТИРУЮЩИЕ ДЕЙСТВИЯ

См. описание переменной и ее полей здесь 799.

Имя переменной: alertActions

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи для *Менеджера.*

Формат 49 записи:

Имя поля	Тип поля	Комментарии
executionType	Целое	
mask	Строка	
action	Строка	
input	Таблица данных	
condition	Строка	
runFromSource	Булевое	

ИНТЕРАКТИВНЫЕ КОРРЕКТИРУЮЩИЕ ДЕЙСТВИЯ

См. описание переменной и ее полей здесь 799.

Имя переменной: interactiveActions

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи для *Менеджера.*

Формат 49 записи:

Имя поля	Типя поля	Комментарии
executionType	Целое	
mask	Строка	
action	Строка	
parameters	Таблица данных	
runFromSource	Булевое	

СТАТУС ТРЕВОГИ

Переменная содержит различные параметры тревоги, такие как количество шаблонов ожидающих тревог. Переменная отображена действием Просмотреть статус 111.

Имя переменной: status

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя.*

Формат 49 записи:

Имя поля	Тип поля	Примечания
enabled	Булевое	Указывает, активирована ли тревога.
pendingInstanceCount	Целое	Количество ожидающих (не подтвержденных) экземпляров тревог.
maxPendingTime	Длинное	Время жизни самого старого ожидающего экземпляра тревоги.
escalated	Булевое	Указывает, эскалирована ли тревога.
escalationReason	Строка	Причина (-ы) эскалации.

АКТИВНЫЕ ТРЕВОГИ

Это список активных 804 и ожидающих 804 экземпляров тревог.

Имя переменной: activeInstances

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*.

Формат 49 записи:

Имя поля	Тип поля	Комментарии
event	Длинное	Идентификатор события тревоги 790 (скрытое поле).
type	Целое	Тип экземпляра: активный 804 и неподтвержденный 804.
time	Дата	Время появления тревоги.

level	Целое	Уровень тревоги.
source	Строка	Контекст источника тревоги, т.е. контекст, чьи события/состояния вызвали тревогу.
message	Строка	Сообщение тревоги.
trigger	Строка	Сообщение триггера тревоги.
cause	Строка	Причина тревоги.
data	Таблица данных	Дата тревоги.

СТАТУС ТРИГГЕРОВ, АКТИВИРУЕМЫХ СОБЫТИЯМИ

Таблица, указывающая статусы отдельных триггеров событий 787. Переменную отображает действие Показать статус 111.

Имя переменной: eventTriggerStatus

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя.*

Формат 49 записи:

Имя поля	Тип поля	Примечания
trigger	Целое	Нулевой индекс триггера в таблице **Триггеры, активированные событиями**.
active	Булевое	Указывает, активны ли триггер и тревога.
details	Таблица данных	Содержит информацию о подстатусе триггера для каждого контекста, соответствующего маске триггера: • путь и описание контекста. • флажок, указывающий, активен ли триггер для данного контекста. • количество событий, зарегистрированных за последний **период** триггера (если триггер активирован более чем одним событием).

СТАТУС ТРИГГЕРОВ, АКТИВИРУЕМЫХ СОСТОЯНИЕМ

Таблица, указывающая на статусы отдельных триггеров переменных 782. Переменную отображает действие показать статус 111.

Имя переменной: variableTriggerStatus

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
trigger	Целое	Нулевой индекс триггера в таблице **Триггеры, активированные состоянием переменной**.

active	Булевое	Указывает, активен ли триггер и тревога.
details	Таблица данных	Содержит информацию о подстатусе триггера для каждого контекста, соответствующего маске триггера: • путь и описание контекста. • флажок, указывающий, активен ли триггер для данного контекста. • Время, истекшее с момента регистрации активирования состояния (если не активно) или деактивирования состояния (если активно). Если это время превышает **Период** активации (или **Период деактивации**), триггер будет активироваться (деактивироваться). • Флажок, указывающий, обнаружено ли биение для данного контекста.

Общие функции [? 70]

У этого контекста нет общих функций.

Общие события [? 73]

Общие события: info (информация) 77, contextStatusChanged (статус изменен) 83

ТРЕВОГА

См. описание события и его полей здесь 790.

Имя события alert

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*.

Период действия: 10 лет

Записи: 1

Формат 49 записи:

Имя поля	Тип поля	Примечания
description	Строка	Описание тревоги.
context	Строка	Контекст, определяющий событие/переменную, которые инициировали тревогу.
entity	Строка	Имя переменной, если тревога была инициирована состоянием переменной/изменением состояния или именем события, если тревога была инициирована событием.
cause	Строка	Причина тревоги.
message	Строка	Сообщение тревоги.
trigger	Строка	Сообщение триггера тревоги.
data	Таблица данных	Данные тревоги.

ДЕАКТИВАЦИЯ

This event is fired when the alert is deactivated for a certain context.

Имя события: deactivation

Права доступа: Доступно на уровне 486 прав доступа *Наблюдатель*

Период действия: 10 лет

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Комментарии
context	Строка	Контекст, для которого была деактивирована тревога.
duration	Длинное	Продолжительность тревоги.

17.4 Приложения

Этот контекст 41 является контейнером, который содержит все приложения 1318 для конкретного пользователя.

Уникальные действия [? 1450]

СОЗДАТЬ ПРИЛОЖЕНИЕ (действие по умолчанию 88)

Это действие используется для создания новых приложений. Позволяет пользователю указывать основные свойства 1319 нового приложения и настраивать его сразу же после создания.

Имя действия: create

Иконка действия: 

Тип действия: Создать 105

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*.

ИМПОРТ

Действие используется для импорта настроек и ресурсов Приложения из дисковой папки, которая была ранее создана действием Экспорт 1464 Приложения.

Во время импорта необходимо определить ряд опций:

- **Путь к папке приложения**. Путь к папке на диске AtomMind Server, которая содержит предварительно экспортированное Приложение. Обычно к этой папке переходят из репозитория Системы контроля версий.
- **Обработка существующих контекстов**. **Пропустить**, **Заменить** или **Обновить** существующие ресурсы приложения (если они уже были ранее импортированы).
- **Обработка ошибок**. Если во время импортирования произошла обработка ошибок, процесс можно либо продолжить (опция **Игнорировать**) или отменить (опция **Отменить**).
- **Путь к папке зависимостей**. Папка содержит подпапки с приложениями, от которых зависит текущее приложение. Эти папки обычно также синхронизируются Системой контроля версий.

Имя действия: import

Иконка действия: 

Неинтерактивный Режим 99**:** Не поддерживается

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*.

Общие действия [? 1450]

Создать на основе шаблона 105, Копировать в дочерние контексты 111, Импорт 108, Экспорт 108, Редактировать права доступа 106, Просмотр событий 109, Поиск/фильтрация 110, различные Групповые действия 101 согласно контекстам потомков.

Состояния и иконки контекста

У этого контекста нет состояний 44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста 43: applications

Имя контекста 42: applications

Описание контекста 43: Приложения

Путь контекста 42: users.USER_NAME.applications

Маска контекста 44: users.*.applications

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание приложения, экспорт и импорт.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [? 61]

У этого контекста нет общих переменных (свойств).

Общие функции [? 70]

Общие функции: makeCopy (сделать копию) 71, delete (удалить) 72

17.5 Приложение

Этот контекст 41 позволяет получить доступ и управлять отдельным приложением 1318.

Уникальные действия [? 1450]

НАСТРОИТЬ (действие по умолчанию 88)

Это действие используется для редактирования свойств 1319 Приложения.

Изменение поля **Имя** во время этой операции меняет название текущего контекста. Это может привести к сбою в работе других компонентов системы, использующих имя/путь контекста в качестве основного идентификатора.

Тип действия: Настроить 105

УПАКОВАТЬ

Это действие упаковывает 1318 ресурсы Приложения как плагины 207 AtomMind Server (*.jar файл). Действие открывает окно, позволяющее сохранить файл плагина в любую папку на диске клиента.

Имя действия: pack

Иконка действия: 

Неинтерактивный режим 99: Не поддерживается

Права доступа: Доступно на уровне 486 прав доступа *Менеджер*.

ЭКСПОРТ

Это действие сохраняет 1318 настройки и ресурсы Приложения в папку на диске машины, где установлен AtomMind Server.

- **Путь к папке приложения**. Путь к папке AtomMind Server, в которую будут экспортированы Приложение и его ресурсы.
- **Обезличить экспортируемые данные**. Удаляет информацию о владельцах экспортируемых ресурсов.
- **Добавить файлы IDE проекта**. Добавляет файлы, необходимые для создания плагина 207 AtomMind Server из Приложения с использованием IDE (Eclipse or Intellij Idea).
- **ID приложения/плагина**. Идентификатор Приложения. Используется для управления зависимостями 1321. Плагин 207 AtomMind Server, созданный из Приложения при помощи IDE, будет иметь такой же идентификатор.
- **Пакет**. Пакет, в который будут помещены классы и ресурсы Приложения в проекте IDE.

Имя действия: export

Иконка действия: 

Неинтерактивный режим 99: Не поддерживается

Права доступа: Доступно на уровне 486 прав доступа *Менеджер*.

ОБНОВИТЬ

Действие обновляет Приложение из папки на диске, которая обычно синхронизируется с Системой контроля версий.

Во время процесса обновления нужно определить несколько опций:

- **Путь к папке приложения**. Путь к папке на диске AtomMind Server, которая содержит предварительно экспортированное Приложение. Обычно к этой папке переходят из репозитория Системы контроля версий.
- **Обработка существующих контекстов**. **Пропустить**, **Заменить** или **Обновить** существующие ресурсы приложения (если они уже были ранее импортированы).
- **Обработка ошибок**. Если во время импортирования произошла обработка ошибок, процесс можно либо продолжить (опция **Игнорировать**) или отменить (опция **Отменить**).
- **Удалить не перечисленные в приложении контексты**. Опция приведет к удалению ресурсов, которые были включены в текущую версию Приложения, но не существуют в импортируемой версии.

Имя действия: update

Иконка действия: 

Неинтерактивный режим 99: Не поддерживается

Права доступа: Доступно на уровне 486 прав доступа *Менеджер*.

ДОБАВЛЕНИЕ НЕСКОЛЬКИХ РЕСУРСОВ

Действие позволяет добавлять многочисленные ресурсы к Приложению, выбрав их из дерева контекста.

Имя действия: multipleResourceAddingAction

Иконка действия: 

Неинтерактивный режим: Не поддерживается

Права доступа: Доступно на уровне прав доступа *Менеджер*.

ПЕРЕТАСКИВАНИЕ РЕСУРСОВ

Действие позволяет добавлять многочисленные ресурсы к Приложению путем перетаскивания.

Имя действия: dndResourceAddingAction

Иконка действия: 

Неинтерактивный режим: Не поддерживается

Права доступа: Доступно на уровне прав доступа *Менеджер*.

Общие действия [?]

Удалить, Сделать копию, Реплицировать, Редактировать права доступа контекста, Просмотреть события, Просмотреть статус

Состояния и иконки контекста

Контекст не имеет состояний. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста: application

Имя контекста: предоставляется пользователем

Описание контекста: предоставляется пользователем

Путь контекста: users.USER_NAME.applications.APPLICATION_NAME

Маска контекста: users.*.applications.*

Права доступа к контексту [?]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Упаковка ресурсов, настройка и удаление.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [?]

Общие переменные: groupMembership (членство группы), validity (пригодность), activeAlerts (активные тревоги)

СВОЙСТВА

См. описание переменной и полей здесь1319.

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне486 с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*.

Формат49 записи:

Имя поля	Тип поля	Примечания
name	String	1 - 50 символов (Должно начинаться с буквы)
description	String	1 - 50 символов
applicationFolderPath	String	
comment	String	

РЕСУРСЫ

См. описание переменной и полей здесь1319.

Имя переменной: resources

Записи: 0...нет ограничений

Права доступа: Доступно для чтения на уровне486 с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*.

Формат49 записи:

Имя поля	Тип поля	Примечания
resource	String	
description	String	
contextDescription	String	
category	String	
subcategory	String	
groupName	String	
groupDescription	String	
version	Integer	
dependencies	Data Table	
createOnFirstServerLaunch	Boolean	

protected	Boolean	

ЛОКАЛИЗАЦИЯ

См. описание переменной и полей здесь1320.

Имя переменной: textResources

Записи: 0...нет ограничений

Права доступа: Доступно для чтения на уровне486 с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*.

Формат49 записи:

Имя поля	Тип поля	Примечания
languages	Data Table	
textResourceBundle	Data Table	

ПЕРЕМЕННЫЕ КОНТЕКСТОВ

См. описание переменной и полей здесь1321.

Имя переменной: contextVariables

Записи: 0...нет ограничений

Права доступа: Доступно для чтения на уровне486 с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*.

Формат49 записи:

Имя поля	Тип поля	Примечания
context	String	
variable	String	

ЗАВИСИМОСТИ

См. описание переменной и полей здесь1321.

Имя переменной: applicationDependencies

Записи: 0...нет ограничений

Права доступа: Доступно для чтения на уровне486 с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*.

Формат49 записи:

Имя поля	Тип поля	Примечания
id	String	

Общие функции [?70]

У этого контекста нет общих функций.

Общие события [?73]

Общие события: info (информация)77

17.6 Автозапуск

Этот контекст41 является контейнером, который содержит все действия Автозапуска941 для определенного пользователя.

Уникальные действия [?1450]

- Добавить действие для автозапуска1469 (действие по умолчанию88)

Другие уникальные действия:

ИНИЦИИРОВАТЬ АВТОЗАПУСК

Запускает все действия Автозапуска, определенные потомками данного контекста, аналогично тому, как они запускаются при загрузке. Удобно для тестирования.

Имя действия:	autorun
Не интерактивный режим99:	не поддерживается
Права доступа:	доступно на уровне486 с правами доступа для *Наблюдателя*.

Общие действия [?1450]

Создать на основе шаблона105, копировать в дочерние контексты111, импорт108, экспорт108, редактировать права доступа106, просмотр событий109, поиск/фильтрация110, различные групповые действия101 согласно контекстам потомков.

Состояния и иконки контекста

У данного контекста нет состояний44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста43: autorunActions

Имя контекста42: autorun

Описание контекста43: Auto Run

Путь контекста42: users.USER_NAME.autorun

Контекстная маска44: users.*.autorun

Права доступа к контексту [?44]

Уровень	Описание
Отсутствует	Нет допуска.

Наблюдатель	Мониторинг основных событий.
Оператор	То же, что у Наблюдателя.
Менеджер	Автозапуск создания, экспорта и импорта действий.
Инженер	То же, что у Менеджера.
Администратор	То же, что у Менеджера.

Общие переменные (свойства) [?61]

У данного контекста нет общих переменных (свойств).

Общие функции [?70]

Общие функции: makeCopy (скопировать)71, delete (удалить)72

СОЗДАТЬ

Создает новое действие Автозапуска.

Имя функции: create

Права доступа: доступно на уровне486 с правами доступа для *Менеджера*.

Записи ввода: 1

Формат49 ввода: аналогичный формату переменной childInfo1471 в контексте Действие автозапуска1470.

Записи вывода: 0

Формат49 вывода: нет

Общие события [?73]

Общие события: info (информация)77

17.6.1 Действие: добавить действие в Автозапуск

ДОБАВИТЬ ДЕЙСТВИЕ ДЛЯ АВТОЗАПУСКА (действие по умолчанию88)

Это действие добавляет новое действие, которое будет автоматически запускаться при загрузке. Это действие по перетаскиванию мышью101 допускает контексты любого типа.

Поток действия:

1) Выберите контекст, как описано в Действиях по перетаскиванию мышью101.

2) Определите90, какое действие будет запущено.

3) Создается новое действие Автозапуска с параметрами, заданными в пунктах (1) и (2).

Тип действия: Перетаскивание мышью101

Иконка действия: 

Имя действия: add

Не интерактивный режим99: не поддерживается

Права доступа: Доступно на уровне486 с правами доступа для *Менеджера*.

17.7 Действие автозапуск

Этот контекст[41] предоставляет доступ к одному действию автозапуск[941] и позволяет им управлять.

Уникальные действия [?[1450]]

НАСТРОИТЬ (действие по умолчанию[88])

Это действие используется для редактирования свойств[942] действия Автозапуск.

Изменение поля **Имя** во время этой операции меняет название данного контекста. Это может привести к сбою в работе других компонентов системы, использующих имя/путь контекста в качестве основного идентификатора.

Тип действия: настроить[105]

ЗАПУСТИТЬ ДЕЙСТВИЕ

Это действие немедленно запускает действие Автозапуска для тестирования.

Имя действия:	launch
Не интерактивный режим[99]:	не поддерживается
Права доступа:	Доступно на уровне[486] в правами доступа для *Наблюдателя*.

Общие действия [?[1450]]

Удалить[106], копировать[109], реплицировать[110], редактировать права доступа[106], просмотр событий[109], показать статус[111]

Состояния и иконки контекста

Иконка	Код	Состояние
	0	Действие Автозапуск активировано
	1	Действие Автозапуск деактивировано

Дополнительная информация

Информация о контексте

Тип контекста[43]: autorunAction

Имя контекста[42]: предоставляется пользователем

Описание контекста[43]: предоставляется пользователем

Путь контекста[42]: users.USER_NAME.autorun.AUTORUN_ACTION_NAME

Контекстная маска[44]: users.*.autorun.*

Права доступа к контексту [?[44]]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Выполнение действия автозапуска.  Вам также нужно иметь права доступа к действию, на которое указывает автозапуск.

	Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Конфигурация и удаление действия автозапуска.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [?61]

Общие переменные: groupMembership (членство группы)67, activeAlerts (Active Alerts)

СВОЙСТВА АВТОМАТИЧЕСКИ ЗАПУСКАЕМОГО ДЕЙСТВИЯ

См. описание переменной и ее полей здесь942.

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне486 в правами доступа для *Наблюдателя*, доступно для записи для *Менеджера*.

Формат49 записи:

Имя поля	Тип поля	Примечания
name	Строка	1 - 50 знаков
description	Строка	1 - 100 знаков
mask	Строка	
action	Строка	
enabled	Булевое	
executionParameters	Таблица данных	

Общие функции [?70]

У данного контекста нет общих функций.

Общие события [?73]

Общие события: info (информация)77, contextStatusChanged (статус изменен)83

17.8 Классы

Этот контекст41 является контейнером, содержащим все классы885 для определенного пользователя.

Уникальные действия [?1450]

СОЗДАТЬ (действие по умолчанию88)

Это действие используется для создания нового класса. Это позволяет пользователю задавать основные опции 890 для нового класса и редактировать его сразу же после создания.

Тип действия: создать 105

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*

Общие действия [? 1450]

Создать на основе шаблона 105, Копировать в дочерние контексты 111, Импортировать 108, Экспортировать 108, Редактировать права доступа к контексту 106, Просмотр событий 109, Поиск/Фильтр 110, различные Группировки действий 101 согласно дочерним контекстам.

Состояния и иконки контекста

Данный контекст не имеет состояний 44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста 43: classes

Имя контекста 42: classes

Описание контекста 43: Classes

Путь контекста 42: users.USER_NAME.classes

Контекстная маска 44: users.*.classes

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание новых классов, экспорт, импорт.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [? 61]

Этот контекст не имеет общих переменных (свойств).

Общие функции [? 70]

Общие функции: копировать (Make Copy) 71, удалить (Delete) 72

CREATE

Создает новый класс.

Имя функции: create

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*

Записи ввода: 1

Формат 50 ввода: Такой же как у переменной childInfo 1474 в контексте Класс 1473.

Записи вывода: 0

Формат 50 вывода: отсутствует

Общие события [? 73]

Общие события: info (Информация) 77

17.9 Класс

Этот контекст 41 предоставляет вам доступ и управление одним классом 885 и его экземплярами.

Уникальные действия [? 1450]

НАСТРОИТЬ

Данное действие Настроить 105 используется для редактирования свойств 890 класса.

Изменение имени класса не разрешено.

Тип действия: настроить 105

Общие действия [? 1450]

Удалить 106, Копировать 109, Реплицировать 110, Редактировать права доступа к контенту 106, Просмотр событий 109, Показать статус 111

Состояния и иконки контекста

Контекст не имеет состояний 44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста 43: class

Имя контекста 42: предоставляется пользователем

Описание контекста 43: предоставляется пользователем

Путь контекста 42: users.USER_NAME.classes.CLASS_NAME

Маска контекста 44: users.*.classes.*

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Управление экземпляром класса. Просмотр конфигурации. Мониторинг основных событий. Просмотр статуса.
Оператор	Те же, что у Наблюдателя.
Менеджер	Удаление класса.
Инженер	Те же, что у Менеджера.

Администратор	Настройка класса.

Общие переменные (свойства) [?61]

Общие переменные: groupMembership (Group Membership)67, validity (Validity)68, activeAlerts (Active Alerts)

СВОЙСТВА

Смотрите описание переменной и ее полей здесь890.

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне486 прав доступа *Наблюдатель, д*оступно для записи на уровне прав доступа *Менеджер*

Формат50 записи:

Имя поля	Тип поля	Примечания
name	String	1 - 50 символов
description	String	1 - 50 символов
namingExpression	String	
storageContext	String	
normalConcurrentBindings	Integer	
maximumConcurrentBindings	Integer	
maximumBindingQueueLength	Integer	
normalConcurrentInstanceBindings	Integer	
maximumConcurrentInstanceBindings	Integer	
maximumInstanceBindingQueueLength	Integer	

ПОЛЯ

Смотрите описание переменной и ее полей здесь891.

Имя переменной: fields

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 прав доступа *Наблюдатель*, доступно для записи на уровне прав доступа *Администратор*

Формат 50 записи:

Имя поля	Тип поля	Примечания
name	String	
type	String	
description	String	
defaultValue	String	
readonly	Boolean	
nullable	Boolean	
key	Boolean	
selvals	Data Table	
extselvals	Boolean	
hidden	Boolean	
inline	Boolean	
encrypted	Boolean	
help	String	
editor	String	
editorOptions	Data Table	
group	String	
primaryKey	Boolean	
length	Integer	

ОТНОШЕНИЯ МНОГИЕ КО МНОГИМ

Смотрите описание переменной и ее полей здесь 891.

Имя переменной: manyToManyRelations

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 прав доступа *Наблюдатель*, доступно для записи на уровне прав доступа *Администратор*

Формат 50 записи:

Имя поля	Тип поля	Примечания
name	String	
description	String	

relatedClass	String	
cascadeDelete	Boolean	

ЖИЗНЕННЫЕ ЦИКЛЫ

Смотрите описание переменной и ее полей здесь 891.

Имя переменной: lifecycles

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 прав доступа *Наблюдатель*, доступно для записи на уровне прав доступа *Администратор*

Формат 50 записи:

Имя поля	Тип поля	Примечания
name	String	
description	String	
states	Data Table	
stateTransitions	Data Table	

ПРОСМОТРЫ

Смотрите описание переменной и ее полей здесь 891.

Имя переменной: views

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 прав доступа *Наблюдатель*, доступно для записи на уровне прав доступа *Администратор*

Формат 50 записи:

Имя поля	Тип поля	Примечания
name	String	
description	String	
columns	Data Table	
filter	Data Table	
sorting	Data Table	

ПРИВЯЗКИ

Переменная содержит привязки класса 894.

Имя переменной: bindings

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 прав доступа *Наблюдатель*, доступно для записи на уровне прав доступа *Менеджер*

Формат 50 записи:

Имя поля	Тип поля	Примечания
target	String	
expression	String	
activator	String	
condition	String	
onstartup	Boolean	
onevent	Boolean	
periodically	Boolean	
period	Long	

Общие функции [? 70]

ОТКРЫТЬ ХРАНИЛИЩЕ

Открыть сессию хранилища.

Имя функции: storageOpen

Права доступа: Доступно на уровне 486 прав доступа *Наблюдатель*

Записи ввода: 1

Формат 50 ввода:

Имя	Тип	Описание
id	Long	Идентификатор сессии хранилища
view	String	Имя отображения класса
query	String	Строка запроса
table	String	Имя таблицы
columns	DataTable	Таблица с настройками колонок
filter	DataTable	Таблица с натройками фильтра
sorting	DataTable	Таблица с настройками сортировки
getData	Boolean	Определяет следует ли открывать сессию или просто вернуть данные
limit	Integer	Максимальное количество возвращаемых экземпляров
offSet	Integer	Количество экземпляров, которое необходимо пропустить с начала результующего списка

Записи вывода: 1

Формат вывода:

Имя	Тип	Описание
id	Long	Идентификатор сессии хранилища
count	Integer	Количество полученных экземпляров
data	DataTable	Содержит таблицу с полученными экземплярами, если параметр *getData* равен true

ЗАКРЫТЬ ХРАНИЛИЩЕ

Закрыть сессию хранилища.

Имя функции: storageClose

Права доступа: Доступно на уровне прав доступа *Наблюдатель*

Записи ввода: 1

Формат ввода:

Имя	Тип	Описание
id	Long	Идентификатор сессии хранилища

Записи вывода: 0

Формат вывода: Нет

ПОЛУЧИТЬ ДАННЫЕ

Возвращает таблицу с экземплярами полученными в сессии.

Имя функции: storageGet

Права доступа: Доступно на уровне прав доступа *Наблюдатель*

Записи ввода: 1

Формат ввода:

Имя	Тип	Описание
id	Long	Идентификатор сессии хранилища
first	Integer	Порядковый номер первого из получаемых экземпляров
count	Integer	Количество получаемых экземпляров

Записи вывода: 1

Формат вывода:

Имя	Тип	Описание
size	Integer	Количество полученных экземпляров
data	DataTable	Таблица с полученными экземплярами

ОБНОВИТЬ ДАННЫЕ

Обновить экземпляры в сессии или полученные с помощью фильтра.

Имя функции: storageUpdate

Права доступа: Доступно на уровне прав доступа *Наблюдатель*

Записи ввода: 0...unlimited

Формат 50 ввода:

Имя	Тип	Описание
id	Long	Идентификатор сессии хранилища
row	Integer	Номер строки, которую необходимо изменить. Используется, если параметр *id* не равен null
column	String	Имя колонки, которую необходимо изменить.
value	DataTable	Новое выставляемое значение, помещенное в таблицу
table	String	Имя таблицы. Используется, если параметр *id* равен null
filter	DataTable	Настройки фильтра. Используется, если параметр id равен null

Записи вывода: 1

Формат 50 вывода:

Имя	Тип	Описание
count	Integer	Количество затрагиваемых записей.

УДАЛИТЬ ДАННЫЕ

Удалить экземпляры в сессии или полученные с помощью фильтра.

Имя функции: storageDelete

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*

Записи ввода: 1

Формат 50 ввода:

Имя	Тип	Описание
id	Long	Идентификатор сессии хранилища
rowList	DataTable	Список строчек, которые необходимо удалить. Используется, если параметр **id** не равен null
table	String	Имя таблицы. Используется, если параметр **id** равен null
filter	DataTable	Настройки фильтра. Используется, если параметр **id** равен null

Записи вывода: 1

Формат 50 вывода:

Имя	Тип	Описание
count	Integer	Количество затрагиваемых записей.

ДОБАВИТЬ ДАННЫЕ

Добавить новые экземпляры в сессию или по имени таблицы.

Имя функции: storageInsert

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*

Записи ввода: 1

Формат ввода:

Имя	Тип	Описание
id	Long	Идентификатор сессии хранилища
values	DataTable	Таблица, содержащая добавляемые экземпляры
table	String	Имя таблицы. Используется, если параметр **id** равен null

Записи вывода: 0...не ограничено

Формат вывода:

Имя	Тип	Описание
instanceId	String	Идентификатор добавленного экземпляра

СВЯЗАТЬ ЭКЗЕМПЛЯРЫ

Связать экземпляры отношением многие-ко-многим.

Имя функции: storageLinkInstance

Права доступа: Доступно на уровне прав доступа *Наблюдатель*

Записи ввода: 1

Формат ввода:

Имя	Тип	Описание
table	String	Имя таблицы
relatedTable	String	Имя связанной таблицы
relationName	String	Имя отношения
relatedIds	DataTable	Таблица с парами идентификаторов экземпляров, которые необходимо связать.

Записи вывода: 0

Формат вывода: Нет

РАЗЪЕДИНИТЬ ЭКЗЕМПЛЯРЫ

Разъединить связанные экземпляры отношением многие-ко-многим.

Имя функции: storageUnlinkInstance

Права доступа: Доступно на уровне прав доступа *Наблюдатель*

Записи ввода: 1

Формат ввода:

Имя	Тип	Описание
table	String	Имя таблицы
relatedTable	String	Имя связанной таблицы
relationName	String	Имя отношения
relatedIds	DataTable	Таблица с парами идентификаторов экземпляров, которые необходимо разъединить.

Записи вывода: 0

Формат 50 вывода: Нет

ПОЛУЧИТЬ ФОРМАТ ДАННЫХ

Возвращает пустую таблицу с форматом, который соответствует формату класса и представления (если оно задано).

Имя функции: getFormat

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*

Записи ввода: 1

Формат 50 ввода:

Имя	Тип	Описание
table	String	Не используется в контексте класса
view	String	Имя представления класса

Записи вывода: 0...не ограничено

Формат 50 вывода: Динамический

Общие события [? 73]

Общие события: info (Информация) 77

СОЗДАН ЭКЗЕМПЛЯР КЛАССА

Это событие формируется каждый раз, когда создается экземпляр класса.

Имя события: classInstanceCreated

Права доступа: Доступно на уровне 486 прав доступа *Наблюдатель*

Период действия: Не сохраняется

Записи: 0...не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
instanceId	String	ID нового экземпляра.
instanceDescription	String	Удобное для чтения описание нового экземпляра.
modificationAuthor	String	Имя пользователя 478, который создал экземпляр.

ИЗМЕНЕН ЭКЗЕМПЛЯР КЛАССА

Это событие формируется, когда редактируется одно или более поле экземпляра класса.

Имя события: classInstanceChanged

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*

Период действия: Не сохраняется

Записи: 0...не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
instanceId	String	ID измененного экземпляра.
instanceDescription	String	Удобное для чтения описание измененного экземпляра.
modificationAuthor	String	Имя пользователя 478, который инициировал изменение.
fieldName	String	Имя измененного поля.
fieldDescription	String	Описание измененного поля.
oldValue	String	Значене старого поля, преобразованного в строку.
newValue	String	Значене нового поля, преобразованного в строку.

УДАЛЕН ЭКЗЕМПЛЯР КЛАССА

Это событие формируется каждый раз, когда удален экземпляр класса.

Имя события: classInstanceDeleted

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*

Период действия: Не сохраняется

Записи: 0...не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
instanceId	String	ID удаленного экземпляра.
instanceDescription	String	Удобное для чтения описание удаленного экземпляра.
modificationAuthor	String	Имя пользователя 478, который удалил экземпляр.
instance	DataTable	Таблица, содержащая значения полей удаленного экземпляра класса.

ОТКОММЕНТИРОВАН ЭКЗЕМПЛЯР КЛАССА

Это событие формируется, когда к экземпляру класса добавляется комментарий.

Имя события: classInstanceCommented

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*

Период действия: Не сохраняется

Записи: 0...не ограничено

Формат 50 записи:

Имя поля	Тип поля	Примечания
instanceId	String	ID комментируемого экземпляра.

instanceDescription	String	Удобное для чтения описание комментируемого экземпляра
author	String	Имя пользователя[478], который добавил комментарий.
comment	String	Текст комментария.

17.10 Общие таблицы

Этот контекст[41] является контейнером, который содержит все общие таблицы[1485] для отдельного пользователя.

Уникальные действия [?[1450]]

- Создать общую таблицу из значения переменной[1484]

Другие уникальные действия:

СОЗДАТЬ ТАБЛИЦУ (действие по умолчанию[88])

Это действие используется для создания новой Общей таблицы. Оно позволяет пользователю задать основные свойства[2179] новой таблицы и настроить их сразу же после создания.

Тип действия: создать[105]

Права доступа: Доступно на уровне[486] с правами доступа для *Менеджера*.

Общие действия [?[1450]]

Создать на основе шаблона[105], копировать в дочерние контексты[111], импорт[108], экспорт[108], редактировать права доступа[106], просмотр событий[109], поиск/фильтрация[110], различные групповые действия[101] согласно контекстам потомков.

Состояния и иконки контекста

У этого контекста нет состояний[44]. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста[43]: commonData

Имя контекста[42]: common

Описание контекста[43]: общие данные

Путь контекста[42]: users.USER_NAME.common

Контекстная маска[44]: users.*.common

Права доступа к контексту [?[44]]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт общих таблиц.

Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [?61]

У данного контекста нет общих переменных (свойств).

Общие функции [?70]

Общие функции: makeCopy (скопировать)71, delete (удалить)72

СОЗДАТЬ

Создает новую общую таблицу.

Имя функции: create

Права доступа: Доступно на уровне486 с правами доступа для *Менеджера*.

Записи ввода: 1

Формат49 ввода: Аналогичные формату переменной childInfo1486 в контексте Общая таблица данных1485.

Записи вывода: 0

Формат49 вывода: нет

СОЗДАТЬ ИЗ ПЕРЕМЕННОЙ

Создает новую общую таблицу из значения переменной контекста.

Имя функции: createFromVariable

Права доступа: Доступно на уровне486 с правами доступа для *Менеджера*.

Записи ввода: 1

Формат49 ввода:

Имя	Тип	Описание
context	Строка	Имя контекста, из которого будет получена переменная.
variable	Строка	Имя переменной.

Записи вывода: 0

Формат49 вывода: нет

Общие события [?73]

Общие события: info (информация)77

17.10.1 Действие: создать общую таблицу из переменной

Это действие позволяет создавать новую общую таблицу, поля и содержимое которой (данные) будут взяты из значения переменной61 контекста.

Поток действия:

1) Выберите источник контекст, как описано в разделе Действия по перетаскиванию мышью101.

2) Задайте 90 переменную (для данного контекста), которая будет использоваться для создания общей таблицы.

3) Новая Общая таблица создана.

Тип действия:	Перетаскивание мышью 101
Имя действия:	createFromVariable
Не интерактивный режим 99:	не поддерживается
Права доступа:	Доступно на уровне 486 с правами доступа для *Менеджера.*

17.11 Общая таблица данных

Этот контекст 41 предоставляет Вам доступ к одной общей таблице 2176 и позволяет ею управлять.

Уникальные действия [? 1450]

НАСТРОИТЬ

Это действие используется для редактирования конфигурации 2179 общей таблицы.

Изменение поля **Имя** во время этой операции меняет название данного контекста. Это может привести к сбою в работе других компонентов системы, использующих имя/путь контекста в качестве основного идентификатора.

Тип действия:	настроить 105

РЕДАКТИРОВАТЬ ДАННЫЕ (действие по умолчанию 88)

Это действие используется для доступа к содержимому 2178 общей таблицы.

Тип действия:	настроить 105
Имя действия:	editData

Общие действия [? 1450]

Удалить 106, создать копию 109, реплицировать 110, редактировать права доступа 106, просмотр событий 109, показать статус 111

Состояния и иконки контекста

У данного контекста нет состояний 44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста 43: commonDataTable

Имя контекста 42: предоставляется пользователем

Описание контекста 43: предоставляется пользователем

Путь контекста 42: common.COMMON_TABLE_NAME

Контекстная маска 44: users.*.common.*

Права доступа к контексту [? 44]

Уровень	Описание

Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий. Просмотр статуса. Просмотр содержания общих таблиц.
Оператор	Просмотр конфигурации. Редактирование содержания общих таблиц.
Менеджер	Конфигурация полей и свойств общей таблицы. Удаление общей таблицы.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [?61]

Общие переменные: groupMembership (членство группы)67, validity (Validity)68, activeAlerts (Active Alerts)

СВОЙСТВА ТАБЛИЦЫ

См. описание переменной и ее полей здесь2179.

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне486 с правами доступа для *Наблюдателя*, доступно для записи для *Менеджера*.

Формат49 записи:

Имя поля	Тип поля	Примечания
name	Строка	1-50 знаков
description	Строка	1-50 знаков
minЗаписи	Целое	0...не ограничено
maxЗаписи	Целое	0...не ограничено, должно быть больше или равно minRecords
recordПрава доступа	Булевое	

ПОЛЯ

См. описание переменной и ее полей здесь2179.

Имя переменной: fields

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне486 с правами доступа для *Наблюдателя*, доступно для записи для *Менеджера*.

Формат49 записи:

Имя поля	Тип поля	Примечания
name	Строка	Ключевое поле, значение может содержать только буквы, числа и символ подчеркивания.
type	Строка	
description	Строка	
default	Строка	
readonly	Булевое	
nullable	Булевое	
key	Булевое	
selvals	Таблица данных	
extselvals	Булевое	
help	Строка	
editor	Строка	

ДОПОЛНИТЕЛЬНЫЕ СВОЙСТВА

См. описание переменной и ее полей здесь 2180.

Имя переменной: customProperties

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи для *Менеджера*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
enabled	Булевое	
validityExpression	Строка	
validityListeners	Таблица данных	

ОБЩАЯ ТАБЛИЦА

Эта переменная используется для доступа и изменения данных общей таблицы.

Имя переменной: value

Записи: Зависит от полей **minRecords** и **maxRecords** переменной Информация о таблице 1486.

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи для *Оператора*.

Формат 49 записи: динамический, зависит от значения переменной Поля 1486.

СТАТУС ТАБЛИЦЫ

У этой переменной метрические значения статуса общей таблицы.

Имя переменной: status

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи для *Менеджера*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
modified	Дата	Дата/время последнего изменения формата и/или данных общей таблицы.

Общие функции [? 70]

У этого контекста нет общих функций.

Общие события [? 73]

Общие события: info (информация) 77

17.12 Конфигурация

Это системный контекст, который предоставляет доступ к глобальной конфигурации 179 AtomMind Server. Он не отображается в видимой части контекстного дерева.

Дополнительная информация

Контекстная информация

Тип контекста 43: config

Имя контекста 42: config

Описание контекста 43: Config

Путь контекста 42: config

Контекстная маска 44: config

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Нет доступа.
Оператор	Нет доступа.
Менеджер	Нет доступа.
Инженер	Нет доступа.
Администратор	Все операции.

Общие переменные (свойства) [?61]

Этот контекст определяет одну переменную для каждой группы настроек, описанных в Параметрах глобальной конфигурации179. Пожалуйста, ознакомьтесь с этим разделом, чтобы иметь представление о формате данных переменных и описании их полей.

Общие функции [?70]

У данного контекста нет общих функций.

Общие события [?73]

Общие события: info (информация)77

17.13 Инструментальные панели

Этот контекст41 является контейнером, который содержит все инструментальные панели912 для определенного пользователя.

Уникальные действия [?1450]

СОЗДАТЬ ИНСТРУМЕНТАЛЬНУЮ ПАНЕЛЬ (действие по умолчанию88)

Это действие используется для создания новых инструментальных панелей. Оно позволяет пользователю задать основные свойства914 для новой инструментальной панели и настроить ее сразу же после создания.

Тип действия:	Create105
Права доступа:	Доступно на уровне486 с правами доступа для *Менеджера*.

Общие действия [?1450]

Создать на основе шаблона105, копировать в дочерние контексты111, импорт108, экспорт108, редактировать права доступа106, просмотр событий109, поиск/фильтрация110, различные групповые действия101 согласно контекстам потомков.

Состояния и иконки контекста

У этого контекста нет состояний44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста43: dashboards

Имя контекста42: dashboards

Описание контекста43: Dashboards

Путь контекста42: users.USER_NAME.dashboards

Контекстная маска44: users.*.dashboards

Права доступа к контексту [?44]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий.

Оператор	Те же, что у Наблюдателя.
Менеджер	Содание, экспорт и импорт инструментальной панели.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [?61]

У этого контекста нет общих переменных (свойств).

Общие функции [?70]

Общие функции: makeCopy (скопировать)71, delete (удалить)72

СОЗДАТЬ

Создает новую инструментальную панель.

Имя функции: create

Права доступа: Доступно на уровне486 с правами доступа для *Менеджера*.

Записи ввода: 1

Формат49 ввода: Аналогичный формату переменной childInfo1491 в контексте Инструментальная панель1490.

Записи вывода: 0

Формат49 вывода: нет

Общие события [?73]

Общие события: info (информация)77

17.14 Инструментальная панель

Этот контекст41 предоставляет доступ к отдельной инструментальной панели912 и позволяет ею управлять.

Уникальные действия [?1450]

ОТКРЫТЬ ПАНЕЛЬ (действие по умолчанию88)

Это действие открывает913 инструментальную панель.

Имя действия: open

Права доступа: Доступно на уровне486 с правами доступа для *Наблюдателя.*

НАСТРОИТЬ

Это действие используется для редактирования свойств914 инструментальной панели.

Изменение поля **Имя** во время этой операции меняет название данного контекста. Это может привести к сбою в работе других компонентов системы, использующих имя/путь контекста в качестве основного идентификатора.

Тип действия: Configure105

Общие действия [?1450]

Удалить 106, создать копию 109, реплицировать 110, редактировать права доступа 106, просмотр событий 109, показать статус 111

Состояния и иконки контекста

Иконка	Код	Состояние
	0	Эта инструментальняа панель относительная.
	1	Эта инструментальная панель абсолютная.

Дополнительная информация

Информация о контексте

Тип контекста 43: dashboard

Имя контекста 42: предоставляется пользователем

Описание контекста 43: предоставляется пользователем

Путь контекста 42: users.USER_NAME.dashboards.DASHBOARD_NAME

Контекстная маска 44: users.*.dashboards.*

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Открытие инструментальной панели. Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Конфиграция и удаление инструментальной панели.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [? 61]

Общие переменные: groupMembership (членство группы) 67, validity (Validity) 68, activeAlerts (Active Alerts)

СВОЙСТВА

См. описание переменной и ее полей здесь 914.

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи для *Менеджера*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
name	Строка	1 - 50 знаков

description	Строка	1 - 50 знаков
title	Строка	Поле может не иметь значения (определено как "nullable")
type	Целое	
validityExpression	Строка	
validityListeners	Таблица данных	

ЭЛЕМЕНТЫ

См. описание переменной и её полей здесь 917.

Имя переменной: elements

Records: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи для *Менеджера*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
title	Строка	
type	Целое	
location	Таблица данных	
parameters	Таблица данных	
validityExpression	Строка	

Общие функции [? 70]

У этого контекста нет общих функций.

Общие события [? 73]

Общие события: info (информация) 77

17.15 Устройства

Этот контекст 41 является контейнером для всех устройств определенного пользователя 478.

Уникальные действия [? 1450]

ДОБАВИТЬ УСТРОЙСТВО (действие по умолчанию 88)

Это действие используется для добавления нового устройства в систему. Оно состоит из нескольких этапов:

1. Система предлагает пользователю задать 90 имя устройства (т.е. имя контекста устройства 1494) и тип (тип драйвера устройства 518, который будет связан с устройством).

2. Новый контекст устройства 1494 создается на сервере.

2. Пользователь настраивает свойства подключения для нового устройства, используя UI процедуру редактировать свойства[91].

Имя действия: create

Иконка действия: 

Права доступа: Доступно на уровне[486] с правами доступа для *Менеджера*.

Общие действия [?[1450]]

Копировать в дочерние контексты[111], импорт[108], экспорт[108], редактировать права доступа[106], просмотр событий[109], поиск/фильтрация[110], различные групповые действия[101] согласно контекстам потомков.

Состояния и иконки контекста

У этого контекста нет состояний[44]. Оно всегда представлено иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста[43]:devices

Имя контекста[42]:devices

Описание контекста[43]: devices

Путь контекста[42]: **users.USER_NAME.devices**

Контекстная маска[44]: **users.*.devices**

Права доступа к контексту [?[44]]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт устройств.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [?[61]]

У этого контекста нет общих переменных (свойств).

Общие функции [?[70]]

Общие функции: makeCopy (скопировать)[71], delete (удалить)[72]

ДОБАВИТЬ УСТРОЙСТВО

Создает новую учетную запись устройства.

Имя функции: add

Права доступа: Доступно на уровне[486] с правами доступа для *Менеджера*.

Записи ввода: 1

Формат 49 ввода:

Имя	Тип	Описание
driver	Строка	ID плагина драйвера устройства 518.
name	Строка	Имя учетной записи.
description	Строка	Описание учетной записи.

Записи вывода: 0

Формат 49 вывода: нет

Общие события [? 73]

Общие события: info (информация) 77

17.16 Устройство

Данный контекст 41 предоставляет Вам доступ к одному устройству и позволяет им управлять.

Уникальные действия [? 1450]

Некоторые действия данного контекста зависят от типа представляющего его устройства. AtomMind Server создает в этом контексте одно действие вызвать функцию 103 на одну операцию устройства. Действия, относящиеся к операциям устройства доступны на уровне 486 с правами доступа для Оператора.

РЕДАКТИРОВАТЬ СВОЙСТВА АККАУНТА УСТРОЙСТВА

Это действие помогает установить параметры связи между AtomMind Server и Device.

Это действие предоставляет доступ к нескольким группам свойств:

- Свойства связи, специфичные для определенного типа устройства (например, имя хоста и номер порта для сетевых устройств)
- Типичные свойства устройства 510
- Опции синхронизации параметров устройства 502
- Группы параметров, операций и событий, который должны быть синхронизированы и доступны в рамках AtomMind

Изменение поля **Имя** во время этой операции меняет название данного контекста. Это может привести к сбою в работе других компонентов системы, использующих имя/путь контекста в качестве основного идентификатора.

Тип действия: настроить 105

Имя действия: setup

Иконка действия: 

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*.

УПРАВЛЕНИЕ УСТРОЙСТВОМ (действие по умолчанию 88)

Открывает все инструментальные панели 912, которые действительны 913 для устройства и имеют набор флажков **загрузка с помощью действия управления**. Если релевантные инструментальные панели не обнаружены, просто запустите действие Настроить устройство 1495.

Имя действия: manage

Иконка действия: 

Не интерактивный режим: не поддерживается

Права доступа: Доступно на уровне с правами доступа для *Наблюдателя*.

КОНФИГУРИРОВАТЬ УСТРОЙСТВО

Это действие используется для редактирования параметров аппаратного устройства. Новые значения параметров сохраняются в кэше сервера во время редактирования и записываются в действительное устройство при следующей синхронизации. См. описание Драйвера устройства, которое управляет устройством.

Данный контекст может содержать некоторые другие уникальные действия; это зависит от управляющего им Драйвера устройства. Например, если им управляет драйвер AtomMind, он будет содержать специфичные для устройства действия.

Тип действия: настроить

СИНХРОНИЗИРОВАТЬ

Это действие сразу же запускает синхронизацию между AtomMind Server и устройством.

Тип действия: вызвать функцию

Имя действия: synchronize

Иконка действия: 

Права доступа: Доступно на уровне с правами доступа для *Оператора*.

ПЕРЕИНИЦИАЛИЗИРОВАТЬ ДРАЙВЕР УСТРОЙСТВА

Это действие очищает кэш настроек устройства, удаляет всю другую информацию об устройстве, о которой помнит AtomMind Server. Затем запускает синхронизацию так, как если бы устройство было подключено к AtomMind Server в первый раз.

Тип действия: вызвать функцию

Имя действия: reset

Иконка действия: 

Права доступа Доступно на уровне с правами доступа для *Менеджера.*

Общие действия [?]

Удалить, реплицировать, редактировать права доступа, просмотр событий, показать статус

Действия, связанные с переменной [?]

ПАРАМЕТРЫ СИНХРОНИЗАЦИИ НАСТРОЕК УСТРОЙСТВА

Это действие позволяет редактировать опции синхронизации для определенной, предоставленной устройством настройки. Оно доступно лишь для входящих от устройства переменных.

Имя действия: editVariableSyncOptions

Не интерактивный режим: не поддерживается

Права доступа: Доступно на уровне с правами доступа для *Менеджера.*

Состояния и иконки контекста

Информация о статусах устройства доступна здесь.

Иконка	Код	Состояние
	20	режим Offline, синхронизировано

	21	режим Online, синхронизировано
	22	приостановлено, синхронизировано
	23	статус подключения не известен, синхронизировано
	30	режим Offline, ожидает синхронизации
	31	режим Online, ожидает синхронизации
	32	приостановлено, ожидает синхронизации
	33	статус подключения не известен, ожидает синхронизации
	40	режим Offline, ошибка синхронизации
	41	режим Online, ошибка синхронизации
	42	приостановлен, ошибка синхронизации
	43	статус подключения не известен, ошибка синхронизации
	50	режим Offline, не синхронизировано, или синхронизация выполняется.
	51	режим Online, не синхронизировано, или синхронизация выполняется
	52	приостановлено, не синхронизировано, или синхронизация выполняется
	53	статус подключения не известен, не синхронизировано, или синхронизация выполняется
	70	режим Offline, подключается (только статус с расширенными полномочиями (extended status))
	71	режим Online, подключается (только статус с расширенными полномочиями, AtomMind Server запросил повторное подключение)
	72	приостановлено, подключается (только статус с расширенными полномочиями, означает, что устройство было приостановлено во время попытки подключения)
	73	статус подключения не известен, подключается (только статус с расширенными полномочиями)
	80	режим Offline, читает метаданные (только статус с расширенными полномочиями, означает, что соединение было разорвано во время чтения метаданных, а синхронизация прервана)
	81	режим Online, читает метаданные (только статус с расширенными полномочиями)
	82	приостановлено, читает метаданные (только статус с расширенными полномочиями, означает, что устройство было приостановлено во время чтения устройством метаданных)
	83	статус подключения не известен, читает метаданные (никогда не возникает на практике)
	90	режим Offline, выполняется синхронизации параметров (только статус с расширенными полномочиями, означает, что соединение было разорвано во время синхронизации параметров, а синхронизация прервана)
	91	режим Online, выполняется синхронизации параметров (только статус с расширенными полномочиями)
	92	приостановлено, выполняется синхронизации параметров (только статус с расширенными полномочиями, означает, что устройство было приостановлено во время синхронизации параметров)
	93	статус подключения не известен, выполняется синхронизации параметров (никогда не возникает на практике)

Дополнительная информация

Информация о контексте

Тип контекста[43]: **device.DEVICE_TYPE**

Имя контекста[42]: предоставляется Драйвером устройства[518]

Описание контекста[43]: предоставляется Драйвером устройства[518]

Путь контекста[42]: **users.USER_NAME.devices.NAME_OF_THIS_CONTEXT**

Контекстная маска[44]: **users.*.devices.***

Права доступа к контексту [?[44]]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Просмотр настроек устройства. Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр свойств учетной записи устройства. Редактирование настроек устройства. Выполнение операций устройства. Мониторинг событий устройства. Запрос синхронизации устройства.
Менеджер	Конфигурация свойств учетной записи устройства. Удаление устройства. Переустановка драйвера устройства. Просмотр статистических данных.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [?[61]]

Некоторые общие переменные данного контекста зависят от типа представленного им устройства. AtomMind Server создает одну общую переменную в данном контексте для каждой настройки устройства. Переменные, относящиеся к параметрам устройства, доступны для чтения на уровне с правами доступа для Наблюдателя и доступны для записи для Оператора.

Общие переменные: groupMembership (членство группы)[67], activeAlerts (Active Alerts)

ОБЩИЕ СВОЙСТВА УСТРОЙСТВА

Содержит типовые свойства[510] устройств.

Имя переменной: genericProperties

Записи: 1

Права доступа: Доступно для чтения уровне[486] с правами доступа для *Наблюдателя*, доступно для записи для *Менеджера*.

Формат[49] записи:

Имя поля	Тип поля	Примечания
name	Строка	
description	Строка	
type	Строка	

syncPeriod	Длинное	Измеряется в миллисекундах
startSyncOnSettingChange	Булевое	
interruptOnError	Булевое	
suspend	Булевое	
disableSynchronousSettingValueRW	Булевое	
extendedStatus	Булевое	
syncQueueLength	Целое	
timeZone	Строка	
metadata	Целое	
activeEntities	Целое	
cache	Целое	
settingsDefaultQuality	Целое	
eventStoragePeriod	Длинное	
dependency	Строка	
status	Строка	
color	Строка	
latitude	Строка	
longitude	Строка	
locationStoragePeriod	Длинное	
offlineAlert	Булевое	
virtualNetwork	Строка	

КАНАЛЫ СТАТИСТИКИ

Эта переменная позволяет управлять статистическими каналами[507] устройства.

Имя переменной: statisticsProperties

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне[486] прав доступа *Наблюдатель*, доступно для записи для *Менеджера*.

Формат[49] записи:

Имя поля	Тип поля	Примечания
name	Строка	Имя канала.

variable	Строка	Имя переменной, на которой базируется канал.
properties	Таблица данных	Свойства 719 канала.

ПАРАМЕТРЫ СИНХРОНИЗАЦИИ НАСТРОЕК УСТРОЙСТВА

Содержит опции синхронизации для каждой из настроек 502 устройства.

Имя переменной: settingSyncOptions

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 прав доступа *Наблюдатель*, доступно для записи для *Менеджера*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
name	Строка	
description	Строка	
mode	Целое	
updateHistoryStorageTime	Длинное	измеряется в миллисекундах
syncPeriod	Длинное	измеряется в миллисекундах
filter	Строка	поле может не иметь значения (определено как "nullable")
master	Строка	
addPreviousValueToVariableUpdateEvent	Булевое	

АКТИВЫ

Эта переменная позволяет управлять активами 501 устройства.

Имя переменной: assets

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 прав доступа *Наблюдатель*, доступно для записи для *Менеджера*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
id	Строка	Уникальный ID актива.
description	Строка	Описание актива в удобной для пользователя форме.
enabled	Булевое	Флажок, указывающий, активирован ли актив, и доступны ли его члены (параметры, операции и события) в AtomMind.

children	Таблица данных	Список вложенных активов. Его формат аналогичен формату данной переменной.

ПЕРЕМЕННЫЕ

Эта таблица позволяет выбрать, какие настройки устройства будут использоваться в контексте Device. Активирован, если опция активные сущности[511] установлена на **выбранные сущности**.

Имя переменной: managedVariables

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне[486] прав доступа *Наблюдатель*, доступно для записи для *Менеджера*.

Формат[49] записи:

Имя поля	Тип поля	Примечания
name	Строка	Имя переменной.
description	Строка	Описание переменной.
group	Строка	Группа переменной.
active	Булевое	Флажок, определяющий, будет ли переменная использоваться контекстом устройства.

ФУНКЦИИ

Эта таблица позволяет выбрать, какие операции устройства будут использоваться в контексте Device. Активирован, если опция активные сущности[511] установлена на **выбранные сущности**.

Имя переменной: managedFunctions

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне[486] прав доступа *Наблюдатель*, доступно для записи для *Менеджера*.

Формат[49] записи:

Имя поля	Тип поля	Примечания
name	Строка	Имя функции.
description	Строка	Описание функции.
group	Строка	Группа функции.
active	Булевое	Флажок, определяющий, будет ли функция использоваться контекстом устройства.

СОБЫТИЯ

Эта таблица позволяет выбрать, какие события устройства будут использоваться в контексте Device. Активирован, если опция активные сущности[511] установлена на **выбранные сущности**.

Имя переменной: managedEvents

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 прав доступа *Наблюдатель*, доступно для записи для *Менеджера*.

Record Format 50:

Имя поля	Тип поля	Примечания
name	Строка	Имя события.
description	Строка	Описание события.
group	Строка	Группа события.
active	Булевое	Флажок, определяющий, будет ли событие использоваться контекстом устройства.

СТАТУС

Возвращает статус устройства.

Имя переменной: status

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя.*

Формат 49 записи:

Имя поля	Тип поля	Примечания
status	Строка	Пользовательский текстуальный статус устройства. Рассчитывается выражением статуса 512, которое является настройкой учетной записи устройства.
color	Цвет	Пользовательский цвет статуса устройства. Рассчитывается выражением цвета 513, которое является настройкой учетной записи устройства.
driver	Строка	драйвер устройства
syncTime	Дата	Дата/время последней синхронизации между AtomMind Server и устройством.
connectionStatus	Целое	статус подключения устройства
syncStatus	Целое	статус синхронизации устройства
syncDetails	Строка	прогресс текущей синхронизации

СТАТУС СИНХРОНИЗАЦИИ НАСТРОЕК

Возвращает информацию о статусе синхронизации 516 для параметров устройства.

Имя переменной: settingsStatus

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя.*

Формат 49 записи:

Имя поля	Тип поля	Примечания
name	Строка	Имя переменной 61 параметра. Это поле скрыто.
setting	Строка	Описание параметра, т.е. описание переменной параметра.
serverTime	Дата	Дата/время последней синхронизации.
duration	Длинное	Продолжительность последней синхронизации, т.е. время, которое потребовалось драйверу устройства для чтения/записи значения параметров из аппаратного оборудования.
updated	Булевое	Флажок "Обновленный на сервере" является верным (true), если значение параметра было обновлено в кэше сервера, и новое значение еще не записано в аппаратное оборудование.
syncStatus	Строка	Текстовое описание текущего статуса синхронизации параметров.

СТАТИСТИКА

Эта переменная позволяет просматривать статистику устройства, т.е. агрегированные данные, собранные статистическими каналами 507 устройства.

Имя переменной: statistics

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя.*

Формат 49 записи:

Имя поля	Тип поля	Примечания
name	Строка	Имя канала.
variable	Строка	Имя переменной, на которой базируется канал.
statistics	Таблица данных	Краткие статистические данные.

МЕСТОПОЛОЖЕНИЕ

Возвращает текущее местоположение устройства. См. Отслеживание местоположения устройства 516 для получения более подробной информации.

Имя переменной: location

Записи: 1

Права доступа: Читаются на уровне 486 с правами доступа для *Наблюдателя*

Формат 50 записи:

Имя поля	Тип поля	Примечания
latitude	Плавающее	Текущая широта устройства в формате числа с плавающей точкой.
longitude	Плавающее	Текущая долгота устройства в формате числа с плавающей точкой.

СТАТУСЫ ПЕРЕМЕННОЙ

Возвращает дополнительную информацию о статусе параметров устройства.

Имя переменной: variableStatuses

Записи: 0...не ограничено

Права доступа: Доступно для чтения уровне 486 с правами доступа для *Наблюдателя.*

Формат 49 записи:

Имя поля	Тип поля	Примечания
name	Строка	Имя переменной
status	Строка	Уникальный ID строки статуса
comment	Строка	Описание статуса в форме, удобной пользователю для чтения

Общие функции [? 70]

Некоторые общие функции данного контекста зависят от типа представляющего его устройства. AtomMind Server создает в данном контексте одну общую функцию для каждой операции устройства. Функции, относящиеся к операциям устройства, доступны на уровне 486 с правами доступа для Оператора.

СИНХРОНИЗИРОВАТЬ

Запускает синхронизацию между Device и AtomMind Server. Можно также синхронизировать только одну переменную, а не целое Device. Передайте имя вашей переменной в параметры функции синхронизации. См. действие Синхронизировать 1495.

Имя функции: synchronize

Права доступа: Доступно на уровне 486 с правами доступа для *Оператора*.

Записи ввода: 1

Формат 49 ввода:

Имя	Тип	Описание
variable	String	Синхронизирует с AtomMind Server

Записи вывода: 0

Формат 49 вывода: нет

ПЕРЕИНИЦИАЛИЗИРОВАТЬ ДРАЙВЕР УСТРОЙСТВА

Заставляет AtomMind Server очистить всю информацию об устройстве и запустить синхронизацию. См. действие Перезапустить драйвер устройства 1495.

Имя функции: reset

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*.

Записи ввода: 0

Формат 49 ввода: нет

Записи вывода: 0

Формат 49 вывода: нет

Общие события [? 73]

Общие события данного контекста зависят от типа представляющего его устройства. AtomMind Server создает в данном контексте одно общее событие для каждого типа события, которое может сгенерировать устройство.

Общие события: info (информация) 77, contextStatusChanged (статус изменен) 83

СИНХРОНИЗИРОВАНО

Это событие появляется в конце каждого цикла синхронизации 514. Оно не принадлежит к какой-либо группе событий и поэтому не появляется в журнале событий во время действия Отследить связанные события 109.

Имя события: synchronized

Права доступа: Доступно на уровене 486 прав доступа *Наблюдатель*

Период действия: Непостоянный

Записи: 0

ФУНКЦИЯ ВЫЗВАНА

Это событие появляется каждый раз при выполнении операции устройства 509.

Имя события: functionCalled

Права доступа: Доступно на уровене 486 с правами доступа для *Наблюдателя*

Период действия: Непостоянный

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
function	Строка	Имя вызванной функции.
input	Таблица данных	Таблица входных данных функции.
output	Таблица данных	Таблица выходных данных функции.
user	Строка	Имя пользователя, который вызвал нулевую функцию, если вызов был произведен системным компонентом.

ХРОНОЛОГИЧЕСКОЕ СОБЫТИЕ ИЗМЕНЕНИЯ

Событие инициировано в течение фазы соединения каждого цикла синхронизации 514, предоставляющего хронологию переменных. Содержит переменную 61 (опредленную в контексте) и значение данных (изменение, которое произошло в определенное время в прошлом).

Имя события: backdatedChange

Права доступа: Доступно на уровене 486 прав доступа *Наблюдатель*

Период действия: Непостоянный (см. событие изменений[84] для постоянного обновления переменных)

Записи: 1

Формат[50] записи:

Имя поля	Тип поля	Примечания
variable	Строка	Имя измененной переменной.
value	Таблица данных	Хронологическое значение переменной.

17.17 Настройка драйверов/плагинов

Данный контекст[41] является контейнером, который содержит все контексты драйверов/расширений[1506], которые отвечают за Драйвер устройства[518] на определенном уровне[521].

Уникальные действия [?[1450]]

АКТИВИРОВАТЬ/ДЕАКТИВИРОВАТЬ РАСШИРЕНИЯ (действие по умолчанию[88])

Это действие используется для редактирования свойства глобальной конфигурации Активные плагины[205]. Оно доступно только в контексте глобальные настройки[521].

Тип действия: настроить[105]

Общие действия [?[1450]]

Редактировать права доступа[106], просмотр событий[109], поиск/фильтрация[110], различные групповые действия[101] согласно контекстам потомков.

Состояния и иконки контекста

У этого контекста нет состояний[44]. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста[43]: "pluginsGlobalConfig" для глобальных настроек, "pluginsUserConfig" для пользовательских настроек.

Имя контекста[42]: plugins

Описание контекста[43]: "Drivers/Plugins"для глобальных настроек, "Drivers/Plugins Per-account Configuration"для пользовательских настроек

Путь контекста[42]: "plugins" для глобальных настроек, "users.USER_NAME.plugins" для пользовательских настроек

Контекстная маска[44]: "plugins" для глобальных настроек, "users.*.plugins" для пользовательских настроек

Права доступа к контексту [?[44]]

Уровень	Описание
Отсутствует	Нет доступа.

Наблюдатель	Нет доступа.
Оператор	Нет доступа.
Менеджер	Нет доступа.
Инженер	Нет доступа к глобальным настройкам. Все операции для пользовательских настроек.
Администратор	Все операции.

Общие переменные (свойства) [?61]

У данного контекста нет общих переменных (свойств).

Общие функции [?70]

У этого контекста нет общих функций.

Общие события [?73]

Общие события: info (информация)77

17.18 Настройка драйвера/плагина

Данный контекст41 предоставляет Вам на определенном уровне521 доступ к драйверу устройства518 и возможность его настраивать.

Уникальные действия [?1450]

РЕДАКТИРОВАТЬ СВОЙСТВА ДРАЙВЕРА/РАСШИРЕНИЯ (действие по умолчанию88)

Это действие используется для редактирования свойств плагина. См. описание плагина для получения информации о его свойствах. В отличие от действия Конфигурировать стандартно, действие Редактировать свойства плагина подсказывает перезапустить сервер после окончания редактирования. Перезагрузка необходима для активирования изменений настройки плагина.

Тип действия: настроить105

Общие действия [?1450]

Редактировать права доступа к контексту106, отслеживать связанные действия109

Состояния и иконки контекста

У этого контекста нет состояний44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста43: pluginConfig

Имя контекста42: plugin-dependent

Описание контекста43: аналогично описанию плагина

Путь контекста42: "plugins.NAME_OF_THIS_CONTEXT" для глобальных настроек, "users.USER_NAME.plugins.NAME_OF_THIS_CONTEXT" для пользовательских настроек

Контекстная маска 44: "plugins.NAME_OF_THIS_CONTEXT" для глобальных настроек, "users.*.plugins.NAME_OF_THIS_CONTEXT" для пользовательских настроек

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Нет доступа.
Оператор	Нет доступа.
Менеджер	Нет доступа.
Инженер	Нет доступа к глобальным настройкам. Все операции для пользовательских настроек.
Администратор	Все операции.

Общие переменные (свойства) [? 61]

Каждый драйвер устройства определяет свои общие переменные для данного контекста.

Общие функции [? 70]

У данного контекста нет общих функций.

Общие события [? 73]

Общие события: info (информация) 77

17.19 Фильтры событий

Этот контекст 41 является контейнером, который содержит все фильтры событий 762 для определенного пользователя.

Уникальные действия [? 1450]

СОЗДАТЬ ФИЛЬТР СОБЫТИЙ (действие по умолчанию 88)

Действие создать 105 используется для создания нового фильтра. Оно позволяет пользователю задать основные свойства 769 для нового фильтра и настроить его сразу же после создания.

Тип действия: создать 105

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*.

Общие действия [? 1450]

Создать из шаблона 105, реплицировать в дочерний узел 111, импортировать 108, экспортировать 108, редактировать права доступа к контексту 106, отслеживать связанные действия 109, поиск/фильтр 110, различные действия, сгруппированные 101 согласно контекстам потомков.

Действия, связанные с переменной. [? 102]

СОЗДАТЬ ФИЛЬТР ДЛЯ ПРОСМОТРА ИСТОРИИ ИЗМЕНЕНИЙ

Это действие создает фильтр событий 762 для просмотра истории изменения переменной.

Имя действия: filterForVariable

Неинтерактивный режим: не поддерживается

Права доступа: доступно на уровне с правами доступа для *Менеджера*.

Действия, связанные с событием [?]

СОЗДАТЬ ФИЛЬТР СОБЫТИЙ

Это действие создает новый фильтр событий с одним правилом для просмотра событий выбранного типа.

Система подсказывает пользователю выбрать, какие поля события будут проверятся выражением фильтра, и задать для них операции сравнения. Если поля не были выбраны, **любое** возникшее событие будет проходить через фильтр.

Имя действия: filterForEvent

Не интерактивный режим: не поддерживается

Права доступа: доступно на уровне с правами доступа для *Менеджера*.

Состояния и иконки контекста

У этого контекста нет состояний. Оно всегда представлено иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста: eventFilters

Имя контекста: filters

Описание контекста: фильтры событий

Путь контекста: users.USER_NAME.filters

Контекстная маска: users.*.filters

Права доступа к контексту [?]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт фильтра.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [?]

У этого контекста нет общих переменных (свойств).

Общие функции [?]

Общие функции: makeCopy (скопировать), delete (удалить)

СОЗДАТЬ

Создает новый фильтр событий.

Имя функции:	create
Права доступа:	Доступно на уровне 486 с правами доступа для *Менеджера*.
Записи ввода:	1
Формат 49 ввода:	аналогичный формату переменной childInfo 1510 в контексте фильтр событий 1509.
Записи вывода:	0
Формат 49 вывода:	нет

Общие события [? 73]

Общие события: info (информация) 77

17.20 Фильтр событий

Этот контекст 41 предоставляет Вам доступ к одному фильтру событий 762 и позволяет им управлять.

Уникальные действия [? 1450]

- параметризовать 1513

Другие уникальные действия:

НАСТРОИТЬ ФИЛЬТР

Это действие используется для редактирования свойств 768 фильтра события.

Изменение поля **Имя** во время этой операции меняет название данного контекста. Это может привести к сбою в работе других компонентов системы, использующих имя/путь контекста в качестве основного идентификатора.

Тип действия: настроить 105

Общие действия [? 1450]

Удалить 106, создать копию 109, реплицировать 110, редактировать права доступа 106, просмотр событий 109, показать статус 111

Состояния и иконки контекста

Иконка	Код	Состояние
	0	Фильтр является нормальным фильтром
	1	Это фильтр по умолчанию (автоматического запуска) владеющего им пользователя

Дополнительная информация

Информация о контексте

Тип контекста 43: eventFilter

Имя контекста 42: предоставляется пользователем

Описание контекста 43: предоставляется пользователем

Путь контекста 42: users.USER_NAME.filters.FILTER_NAME

Маска контекста 44: users.*.filters.*

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Активация фильтра. Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Конфигурация и удаление фильтра.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [? 61]

Общие переменные: groupMembership (членство группы) 67, activeAlerts (Active Alerts)

СВОЙСТВА ФИЛЬТРА

См. описание переменной и ее полей здесь 769.

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для редактирования для *Менеджера*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
name	Строка	1-50 знаков
description	Строка	1-50 знаков
showDataFieldNames	Булевое	
showServerContextNames	Булевое	
showServerEventNames	Булевое	

ПРАВИЛА ФИЛЬТРАЦИИ

См. описание переменной и ее полей здесь 769.

Имя переменной: rules

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для редактирования для *Менеджера*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
description	Строка	
mask	Строка	
event	Строка	
enabled	Булевое	
level	Целое	
expression	Строка	
parameterized	Булевое	
color	Цвет	

ОСНОВНЫЕ ОТОБРАЖАЕМЫЕ ПОЛЯ

См. описание переменной и ее полей здесь 769.

Имя переменной: shownFields

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для редактирования для *Менеджера*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
context	Булевое	
event	Булевое	
level	Булевое	
data	Булевое	
ack	Булевое	

ДОПОЛНИТЕЛЬНЫЕ ОТОБРАЖАЕМЫЕ ПОЛЯ

См. описание переменной и ее полей здесь 769.

Имя переменной: additionalFields

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для редактирования для *Менеджера*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
name	Строка	Только для чтения
description	Строка	Только для чтения
edescs	Строка	Только для чтения
shown	Булевое	

ПРАВИЛА ЦВЕТОВОГО ВЫДЕЛЕНИЯ

См. описание переменной и ее полей здесь 769.

Имя переменной: customHighlighting

Записи: 0...не ограниченно

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для редактирования для *Менеджера*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
mask	Строка	
event	Строка	
level	Целое	
expression	Строка	
color	Цвет	

НАСТРОЙКА ПРОСМОТРА ИСТОРИИ

См. описание переменной и ее полей здесь 769.

Имя переменной: historySettings

Записи: 0...не органичено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для редактирования для *Менеджера*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
limitTimeFrame	Булевое	
timeFrame	Целое	
timeUnit	Целое	
useCustomEndTimePoint	Булевое	
customEndTimePoint	Дата	

Общие функции [?70]

ПОЛУЧИТЬ ПАРАМЕТРЫ

Возвращает первоначальную (незаполненную) таблицу параметров, которая может заполняться и использоваться для активации параметризованного фильтра[772].

Имя функции: getParameters

Права доступа: Доступно на уровне[486] с правами доступа для *Наблюдателя*

Записи ввода: 1

Формат[50] ввода:

Имя	Тип	Описание
realtime	Булевое	Определяет, подходит ли таблица параметров для реального времени (если true), или должен возвращаться исторический раздел Журнала Событий (если false).

Записи вывода: 0...не ограничено

Формат[50] вывода: Динамический

Общие события [?73]

Общие события: info (информация)[77]

17.20.1 Действие: параметризовать

Это действие используется для параметризации[772] фильтра событий.

Поток действий:

1. Сервер подготавливает список категорий событий, которые подходят для параметризации.
2. Система подсказывает пользователю выбрать[90] категории событий для параметризации.
3. Сервер генерирует данные параметризации и сохраняет их в свойствах фильтра.
4. Параметризованный фильтр открывается для редактирования при помощи GUI процедуры редактировать свойства[91].

Имя действия: parameterize

Неинтерактивный режим[99]: не поддерживается

Права доступа: Доступно на уровне[486] с правами доступа для *Наблюдателя*.

17.21 События

Это системный контекст, который предоставляет доступ к различным данным и операциям, связанным с управлением событиями[73]. Он не отображается в видимой части контекстного дерева.

Уникальные действия [?1450]

УДАЛИТЬ ИСТЕКШИЕ СОБЫТИЯ

Удаляет истекшие события из истории событий.

Тип действия:	вызвать функцию 103
Имя действия:	removeExpired
Права доступа:	Доступно на уровне 486 с правами доступа для *Инженеров*.

Действия, относящиеся к переменным [? 102]

ПОДТВЕРДИТЬ ТРЕВОГУ

Это действие доступно для:

- Переменной активного экземпляра 1454 любого контекста тревоги 1454
- Переменной активных тревог 67 любого контекста сервера

Это действие позволяет подтвердить событие тревоги 806, которое было выбрано в таблице активных экземпляров или активных тревог. Оно предлагает оператору сначала ввести текст подтверждения.

Имя действия:	acknowledgeAlert
Неинтерактивный режим 99:	не поддерживается
Права доступа:	Доступны на уровне 486 прав доступа *Отсутствует*

Дополнительная информация

Информация о контексте

Тип контекста 43: события

Имя контекста 42: события

Описание контекста 43: события

Путь контекста 42: события

Контекстная маска 44: события

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Доступ к истории событий.
Оператор	Подтверждение событий.
Менеджер	Удаление событий.
Инженер	Массовое удаление событий. Удаление просроченных событий.
Администратор	Все операции.

Общие переменные (свойства) [? 61]

У данного контекста нет общих переменных (свойств).

Общие функции [? 70]

ПОДТВЕРДИТЬ СОБЫТИЕ

Добавляет новое подтверждение 73 в событие.

Имя функции: acknowledge

Права доступа: Доступно на уровне 486 с правами доступа для *Оператора*.

Записи ввода: 1

Формат 49 ввода:

Имя	Тип	Описание
context	Строка	Контекст события для подтверждения.
name	Строка	Имя события для подтверждения.
id	Длинное	ID события для подтверждения.
ack	Строка	Текст подтверждения.
author	Строка	Автор подтверждения (факультативное поле, используется username активного пользователя, если автор не был задан). Это простое текстовое поле, поэтому его не следует считать проверяемым (auditable) (должно быть всегда точным).

Записи вывода: 0

Формат 49 вывода: нет

ОБОГАТИТЬ СОБЫТИЕ

Добавляет новое обогащение 76 к событию.

Имя функции: enrich

Права доступа: Accessible at *Engineer* permission level 486

Записи ввода: 1

Формат 50 ввода:

Имя	Тип	Описание
context	Строка	Контекст события для подтверждения.
name	Строка	Имя события для подтверждения.
id	Длинное	Идентификатор события для подтверждения.
enrichmentName	Строка	Имя обогащения. Может содержать только английские буквы, цифры и подчеркивания.
enrichmentValue	Строка	Строка значения обогащения. Цифры, даты и другие типы обогащений должны быть закодированы в строки с использованием метода, предпочтительного для инструментов, которые должны будут позже анализировать это обогащение.

Записи вывода: 0

Формат вывода: Отсутствует

УДАЛИТЬ СОБЫТИЕ

Навсегда удаляет событие из истории событий.

Имя функции: delete

Права доступа: Доступно на уровне с правами доступа для *Менеджера*.

Записи ввода: 0...не ограничено

Формат ввода:

Имя	Тип	Описание
context	Строка	Путь контекста события, в котором произвошло событие.
name	Строка	Имя события.
id	Длинное	Идентификатор события.

Записи вывода: 0

Формат вывода: нет

УДАЛИТЬ ИСТЕКШИЕ СОБЫТИЯ

Удаляет все истекшие события из истории событий. Обратите внимание, что обычно все истекшие события автоматически удаляетAtomMind Server при периодическом выполнении этой функции.

Имя функции: removeExpired

Права доступа: Доступно на уровне с правами доступа для *Инженера*.

Записи ввода: 0

Формат ввода: нет

Записи вывода: 0

Формат вывода: нет

ПОЛУЧИТЬ ИСТОРИЮ СОБЫТИЯ

Возвращает историю определенного события. См. действие Просмотреть историю событий.

Имя функции: get

Права доступа: Доступно на уровне с правами доступа для *Наблюдателя*.

Записи ввода: 0

Формат ввода:

Имя	Тип	Описание
mask	Строка	Контекстная маска, определяющая ряд контекстов, из которых необходимо выбрать события.
event	Строка	Имя события.
filter	Строка	Текст выражения для фильтрации событий или NULL для отключения фильтрации.

<table>
<tr><td></td><td></td><td>
<table>
<tr><td colspan="2">Среда вычисления 114 выражения фильтра:</td></tr>
<tr><td>Контекст по умолчанию 119</td><td>Контекст, в котором произошло событие.</td></tr>
<tr><td>Таблица данных по умолчанию 120</td><td>Таблица данных события.</td></tr>
<tr><td>Ряд по умолчанию 119</td><td>0</td></tr>
<tr><td>Переменные среды 123</td><td>
<table>
<tr><th>Имя переменной</th><th>Тип значения</th><th>Описание</th></tr>
<tr><td>id</td><td>Длинное</td><td>Уникальный идентификатор события.</td></tr>
<tr><td>context</td><td>Строка</td><td>Полный путь к контексту события.</td></tr>
<tr><td>event</td><td>Строка</td><td>Имя события.</td></tr>
<tr><td>level</td><td>Целое</td><td>Уровень 75 события.</td></tr>
<tr><td>time</td><td>Дата</td><td>Временная метка события.</td></tr>
<tr><td>acknowledgements</td><td>Таблица данных</td><td>Таблица подтверждений 76 события.</td></tr>
<tr><td>enrichments</td><td>Таблица данных</td><td>Таблица обогащений 76 события.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>fromDate</td><td>Дата</td><td>Временная метка, используемая для выбора только тех событий, которые произошли после определенной даты. Если значение NULL, будут выбраны все события.</td></tr>
</table>

<table>
<tr><td>toDate</td><td>Дата</td><td>Временная метка, используемая для выбора только тех событий, которые произошли до определенной даты. Если значение NULL, будут выбраны все события.</td></tr>
<tr><td>dataAsTable</td><td>Булевое</td><td>Это скрытое поле.

Если оно равно false (по умолчанию), поле таблицы данных каждого события помещается в отдельное поле вывода функции.

Если оно равно true, таблица данных каждого события помещается в отдельное поле (тип - Таблица данных) вывода функции.</td></tr>
<tr><td>sortField</td><td>Строка</td><td>Это скрытое поле. Оно определяет порядок сортировки событий:

• context - сортировать по пути источника контекста
• name - сортировать по названию типа события
• creationtime - сортировать по времени создания
• expirationtime - сортировать по времени истечения
• level - сортировать по уровню
• count - сортировать по количеству дублированных событий</td></tr>
<tr><td>sortAscending</td><td>Булевое</td><td>Это скрытое поле. Оно определяет, должен ли использоваться восходящий порядок сортировки.</td></tr>
<tr><td>limit</td><td>Целое</td><td>Это скрытое поле. Оно определяет максимальное количество событий для загрузки или NULL для загрузки всех доступных событий. Эта опция не совместима с фильтрацией.</td></tr>
<tr><td>additionalCriteria</td><td>Таблица данных</td><td>Это скрытое поле. Определяет дополнительные критерии фильтрации уровня хранения. Подходит только для реляционных БД.

<table>
<tr><th>Иля поля</th><th>Тип значения</th><th>Описание</th></tr>
<tr><td>logical</td><td>Integer</td><td>Логическая операция:

• 0 - НЕТ
• 1 - И
• 2 - ИЛИ</td></tr>
<tr><td>type</td><td>Integer</td><td>Тип фильтра:

• 0 - Условие
• 1 - Вложенное условие</td></tr>
<tr><td>column</td><td>String</td><td>Имя столбца.</td></tr>
<tr><td>operation</td><td>String</td><td>Операция фильтрации. Доступны следующие операции:

• EQUALS</td></tr>
</table>
</td></tr>
</table>

		<table><tr><td></td><td></td><td>• **IS_GREATER_THAN** • **IS_GREATER_OR_EQUAL_THAN** • **IS_LESS_THAN** • **IS_LESS_OR_EQUAL_THAN**</td></tr><tr><td>value</td><td>String</td><td>Значение столбца.</td></tr><tr><td>nested</td><td>Data Table</td><td>Определяет вложенное условие.</td></tr></table>

Записи вывода: 0...не ограничено

Формат 49 **вывода:** динамический

Имя	Тип	Описание
eId	Длинное	Идентификатор события (скрытое поле).
eCreationtime	Дата	Время создания события.
eExpirationtime	Дата	Время истечения события.
eContext	Строка	Путь контекста источника события.
eName	Строка	Имя типа события.
eLevel	Целое	Уровень события.
eCount	Целое	Количество событий (если подобные события были блокированы дедупликацией).
eAcknowledgements	Таблица данных	Подтверждения события.
eData	Таблица данных	Данные, специфичные для события. Это поле будет включено в вывод, только если включены настройки ввода dataAsTable.
*	*	Оставшиеся поля будут добавляться, только если выключены настройки ввода dataAsTable. Количество, имена, типы и другие параметры полей будут соответствовать тем, которые заявлены в определении событий.

ПОЛУЧИТЬ СОБЫТИЯ ПО ИДЕНТИФИКАТОРУ

Возвращает одно событие с определенным идентификатором.

Имя функции: getById

Права доступа: Доступны на уровне 486 с правами доступа для *Наблюдателя*

Записи ввода: 1

Формат 50 ввода:

Имя	Тип	Описание
context	Строка	Контекст события.
event	Строка	Имя типа события.
id	Длинное	Идентификатор события.
dataAsTable	Булевое	Это скрытое поле. Если оно равно false (по умолчанию), каждое поле таблицы данных события помещается в отдельное поле вывода функции. Если равно true, каждая таблица данных события помещается в одно поле (тип - Таблица данных) вывода функции.

Записи вывода: 1

Формат 50 вывода: Динамический:

Имя	Тип	Описание
eId	Длинное	Идентификатор события (скрытое поле).
eCreationtime	Дата	Время создания события.
eExpirationtime	Дата	Время истечения события.
eContext	Строка	Путь контекста источника события.
eName	Строка	Имя типа события.
eLevel	Целое	Уровень события.
eCount	Целое	Количество событий (если подобные события блокировались дедупликацией).
eAcknowledgements	Таблица данных	Подтверждения события.
eData	Таблица данных	Данные, специфичные для события. Это поле будет включено в вывод, только если включены настройки ввода dataAsTable.
*	*	Оставшиеся поля будут добавляться, только если выключены настройки ввода dataAsTable. Количество, имена, типы и другие параметры полей будут соответствовать тем, которые заявлены в определении событий.

УДАЛИТЬ СОБЫТИЯ

Удаляет события из истории событий.

Имя функции: massDelete

Права доступа: Доступно на уровне 486 с правами доступа для *Инженера*.

Записи ввода: 1

Формат 49 ввода:

Имя	Тип	Описание
mask	Строка	Контекстная маска, определяющая ряд контекстов, из которых необходимо удалить события.
event	Строка	Имя события.
startDate	Дата	Nullable (Нет данных). Если задано, удаляются только те события, которые возникают после startDate.
endDate	Дата	Nullable (Нет данных). Если задано, удаляются только те события, которые возникли до endDate.

Записи вывода: 0

Формат 49 вывода: нет

Общие события [? 73]

Общие события: info (информация) 77

17.22 Коррелляторы событий

Данный контекст 41 является контейнером для всех корреляторов событий 774 определенного пользователя.

уникальные действия [? 1450]

СОЗДАТЬ (Действие по умолчанию 88)

Действие используется для создания нового коррелятора событий. Оно позволяет пользователю указать основные свойства 890 для нового коррелятора событий.

Тип действия: Создать 105

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*.

общие действия [? 1450]

Создать на основе шаблона 105, Копировать в дочерние контексты 111, Импорт 108, Экспорт 108, Редактировать права доступа 106, Просмотр событий 109, Поиск/фильтрация 110, различные групповые действия 101 согласно контекстам потомков.

Состояния и иконки контекста

У этого контекста нет состояний 44. Оно всегда представлено иконкой .

Дополнительная информация

Информация о контексте

Тип контекста[43]: eventCorrelators

Имя контекста[42]: eventCorrelators

Описание контекста[43]: Event Correlators

Путь контекста[42]: users.USER_NAME.eventCorrelators

Контекстная маска[44]: users.*.eventCorrelators

Права доступа к контексту [?[44]]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий.
Наблюдатель	Те же, что у Наблюдателя.
Менеджер	Создание нового коррелятора событий, экспорт и импорт.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [?[61]]

У этого контекста нет общих переменных (свойств).

общие функции [?[70]]

Общие функции: makeCopy (Скопировать)[71], delete (Удалить)[72]

СОЗДАТЬ

Создает новый коррелятор событий.

Имя функции: create

Права доступа: Доступно на уровне[486] с правами доступа для *Менеджера*.

Записи ввода: 1

Формат[50] ввода: Такой же, как у переменной childInfo[1522] контекста Коррелятор событий[1522].

Записи вывода: 0

Формат[50] вывода: нет

Общие события [?[73]]

Общие события: info (Информация)[77]

17.23 Коррелятор событий

Контексты Коррелятора событий хранятся в контейнере Коррелляторы событий[152] Системного дерева. Каждый контекст дает доступ к одному коррелятору событий[774] и позволяет им управлять.

уникальные действия [?[1450]]

КОНФИГУРИРОВАТЬ

Действие Конфигурировать[105] используется для редактирования свойств коррелятора событий.

Изменение поля **Имя** во время данной операции приведет к переименованию текущего контекста. Это может вызвать сбой в работе других компонентов системы, которые используют имя/путь контекста в качестве первичного идентификатора.

Тип действия: Конфигурировать[105]

общие действия [?[1450]]

Удалить[106], Создать копию[109], Реплицировать[110], Редактировать права доступа[106], Просмотр событий[109], Показать статус[111]

Состояния и иконки контекста

У этого контекста нет состояний[44]. Оно всегда

Иконка	Код	Состояние
	0	Данный коррелятор событий включен.
	1	Данный коррелятор событий отключен.

Дополнительная информация

Информация о контексте

Тип контекста[43]: eventCorrelator

Имя контекста[42]: задается пользователем

Описание контекста[43]: задается пользователем

Путь контекста[42]: users.USER_NAME.eventCorrelators.CORRELATOR_NAME

Контекстная маска[44]: users.*.eventCorrelators.*

Права доступа к контексту [?[44]]

Уровень	Описание
Отсутствует	Нет доступа
Наблюдатель	Мониторинг основных событий. Просмотр состояний.
Оператор	Просмотр настроек Коррелятора событий.
Менеджер	Удаление Коррелятора событий. Настрой Коррелятора событий.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

общие переменные (свойства) [?[61]]

Общие переменные: groupMembership (Членство в группах)[67], activeAlerts (Активные тревоги)[67]

СВОЙСТВА

Определяет основные свойства коррелятора событий.

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа *Наблюдатель*, возможность записи на уровне 486 с правами доступа *Менеджер*

Формат 50 записи:

Имя поля	Тип поля	Описание поля	Примечания
name	String	Имя контекста	Имя контекста коррелятора событий используется для ссылки на этот контекст из других частей системы.
description	String	Описание контекста	Отображаемое имя контекста коррелятора событий.
enabled	Boolean	Активный	Включена ли корреляция событий. Если это свойство `true`, данный контекст будет обрабатывать потоки событий.
correlatorScript	String	Скрипт коррелятора	Скрипт языка Streaming SQL, который определяет логику корреляции событий.
outputEventFormat	Data Table	Формат исходящих событий	Определяет формат событий 819 для исходящих событий, генерируемых коррелятором. Данное свойство используется только если имя контекста не указано в определении потока. В противном случае, используется событие указанного контекста. Вы можете просмотреть исходящие события контекста коррелятора, выполнив на нем действие Просмотр событий 109.

общие функции [? 70]

У этого контекста нет общих функций.

Общие события [? 73]

Этот контекст имеет следующие общие события.

Общие события: Добавлен дочерний контекст 78, Удален дочерний контекст 78, Добавлен дескриптор переменной 80, Удален дескриптор переменной 80, Добавлен дескриптор функции 80, Удален дескриптор функции 81, Добавлен дескриптор события 81, Удален дескриптор события 81, Базовая информация контекста изменена 83, Переменная контекста изменена 84, Изменение 84, Изменение статуса 83.

17.24 Избранные

Этот контекст 41 является контейнером, который содержит все избранное 2186 для определенного пользователя.

Уникальные действия [? 1450]

- Добавить в избранное 1526 (действие по умолчанию 88)

Общие действия [? 1450]

Создать из шаблона 105, реплицировать в дочерний узел 111, импортировать 108, экспортировать 108, редактировать права доступа к контексту 106, отслеживать связанные события 109, поиск/фильтр 110, различные действия, сгруппированные 101 согласно контекстам потомков.

Состояния и иконки контекста

У этого контекста нет состояний[44]. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста[43]: избранное

Имя контекста[42]: избранное

Описание контекста[43]: избранное

Путь контекста[42]: users.USER_NAME.favourites

Контекстная маска[44]: users.*.favourites

Права доступа к контексту [?[44]]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт избранного.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [?[61]]

У этого контекста нет общих переменных (свойств).

Общие функции [?[70]]

Общие функции: makeCopy (скопировать)[71], delete (удалить)[72]

СОЗДАТЬ

Создает новое избранное.

Имя функции: create

Права доступа: Доступно на уровне[486] с правами доступа для *Менеджера.*

Записи ввода: 1

Формат[49] ввода: Аналогичный формату переменной childInfo[1527] в контексте Избранное[1526].

Записи вывода: 0

Формат[49] вывода: нет

Общие события [?[73]]

Общие события: info (информация)[77]

17.24.1 Действие: добавить в избранное

Это действие добавляет контексты в таблицу Избранное. Оно реализуется как действие по перетаскиванию мышью[101], которое принимает контексты любого типа. Когда Вы перетаскиваете узел Системного дерева в узел Избранные, создается новое Избранное независимо от того, что Вы перетащили мышью.

Только что созданное Избранное можно запустить при помощи действия по умолчанию "Запустить действие" в новом контексте Избранное (например, дважды кликнув по нему в Клиенте[359]).

Порядок действий:

1. Выберите контекст, как описано в разделе Действия по перетаскиванию мышью[101]. Например, в методе выбора Перетаскивание мышью, кликните по узлу Системное дерево, для которого необходимо применить это действие. Удерживайте левую кнопку мыши нажатой и перетащите ее на узел Избранное.
2. На этом этапе Вам будет необходимо определить[90], какое Действие (для данного контекста) будет выполнено при активировании Избранного.
3. После того, как Вы определили действие, будет создано новое Избранное.

Тип действия:	Перетаскивание мышью[101]
Иконка действия:	
Имя действия:	add
Не интерактивный режим[99]:	не поддерживается
Права доступа:	Доступно на уровне[486] с правами доступа для *Менеджера*.

17.25 Избранное

Этот контекст[41] предоставляет Вам доступ к одному действию избранное[2186].

Уникальные действия [?[1450]]

ЗАПУСТИТЬ ДЕЙСТВИЕ (действие по умолчанию[88])

Это действие запускает цель избранного.

Имя действия:	launch
Неинтерактивный режим[99]:	Не поддерживается
Права доступа:	Доступно на уровне[486] с правами доступа для *Наблюдателя*.

НАСТРОИТЬ

Это действие используется для редактирования свойств[2187] Избранного.

Изменение поля **Имя** во время этой операции меняет название данного контекста. Это может привести к сбою в работе других компонентов системы, использующих имя/путь контекста в качестве основного идентификатора.

Тип действия:	настроить[105]

Общие действия [?[1450]]

Удалить 106, создать копию 109, реплицировать 110, редактировать права доступа 106, просмотр событий 109, показать статус 111

Состояния и иконки контекста

Иконка	Код	Состояние
	0	Избранное активировано
	1	Избранное деактивировано

Дополнительная информация

Информация о контексте

Тип контекста 43: избранное

Имя контекста 42: предоставляется пользователем

Описание контекста 43: предоставляется пользователем

Путь контекста 42: users.USER_NAME.favourites.FAVOURITE_NAME

Контекстная маска 44: users.*.favourites.*

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Доступ к избранному. Вам необходимо иметь достаточно прав для доступа к действию, на которое указывает избранное. Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Конфигурация и удаление избранного.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [? 61]

Общие переменные: groupMembership (Членство группы) 67, activeAlerts (Активные тревоги)

СВОЙСТВА

См. описание переменной и ее полей здесь 2187.

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи для *Менеджера*.

Формат 49 записи:

Имя поля	Тип поля	Примечания

name	Строка	1 - 50 знаков
description	Строка	1 - 100 знаков
mask	Строка	
action	Строка	
enabled	Булевое	
executionParameters	Таблица данных	

Общие функции [?70]

У этого контекста нет общих функций.

Общие события [?73]

Общие события: info (информация)77

17.26 Группы устройств

Этот контекст41 является контейнером, который содержит все группы751 определенного типа753.

Уникальные действия [?1450]

СОЗДАТЬ ГРУППУ (действие по умолчанию88)

Это действие используется для создания новых групп. Оно позволяет пользователю задавать основные свойства756 для новой группы и настраивать ее сразу же после создания.

Тип действия: создать105

Права доступа: Доступно на уровне486 с правами доступа для *Менеджера*.

Общие действия [?1450]

Создать на основе шаблона105, копировать в дочерние контексты111, импорт108, экспорт108, редактировать права доступа106, просмотр событий109, поиск/фильтрация110, различные групповые действия101 согласно контекстам потомков.

Состояния и иконки контекста

У данного контекста нет состояний44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста43: группы

Имя контекста42: "usergroups" для групп пользователя478 контейнера, "devgroups" для групп устройства контейнера.

Описание контекста43: "User Groups" для групп пользователя478 контейнера, "Device Groups" для групп устройства контейнера.

Путь контекста42: "usergroups" для групп пользователя478 контейнера, "users.USER_NAME.devgroups" для групп устройства контейнера.

Маска контекста[44]: "usergroups" для групп пользователя[478] контейнера, "users.*.devgroups" для групп устройства контейнера.

Права доступа к контексту [?[44]]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт групп.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [?[61]]

У этого контекста нет общих переменных (свойств).

Общие функции [?[70]]

Общие функции: delete (удалить)[72]

СОЗДАТЬ

Создать новую группу.

Имя функции: create

Права доступа: Доступно на уровне[486] с правами доступа для *Менеджера*.

Записи ввода: 1

Формат[49] ввода: Аналогичный формату переменной childInfo[1532] в контексте Группа[1529].

Записи вывода: 0

Формат[49] вывода: нет

Общие события [?[73]]

Общие события: info (информация)[77]

17.27 Группа устройств

Этот контекст[41] предоставляет Вам доступ к одной группе[751] и позволяет ею управлять.

Уникальные действия [?[1450]]

- Создать датчик статуса группы[1535]

Другие уникальные действия:

НАСТРОИТЬ ГРУППУ (действие по умолчанию[88])

Это действие используется для редактирования свойств[756] Группы.

Изменение поля **Имя** во время этой операции меняет название данного контекста. Это может привести к сбою в работе других компонентов системы, использующих имя/путь контекста в качестве основного идентификатора.

Тип действия: настроить 105

СОЗДАТЬ В ГРУППЕ

Это действие используется для создания нового объекта и добавления его в группу сразу же после создания. Тип объекта совпадает с типом группы, например, **Тревога** для **Группы тревог** и пр.

Поток действия:

1. Запустите действие создать 105 из контекста "контейнера", соответствующего группе, например, контекст **тревог администратора** 479 для **Группы тревог Администратора**.

2. Добавить только что созданный объект в группу.

Имя действия: create

Иконка действия: 

Не интерактивный режим 99**:** Не поддерживается

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*.

СОЗДАТЬ ВЛОЖЕННУЮ ГРУППУ

Это действие используется для создания вложенной группы внутри данной группы.

Имя действия: createNestedGroup

Иконка действия: 

Права доступа: Доступно на уровне 486 прав доступа для *Менеджера*

РЕПЛИЦИРОВАТЬ ИЛИ ДОБАВИТЬ В ГРУППУ

Это действие по перетаскиванию мышью 101, которое принимает контексты того же типа, что и члены группы, т.е. контексты пользователя 1608 для групп Пользователя и пр. Логика данного действия совершенно отличается в двух случаях:

- Когда принятый контекст не является членом данной группы.
- Когда принятый контекст уже является членом группы.

В первом случае принятый контекст просто добавляется в список членов группы. В этом случае не выполняется взаимодействие с используемым.

Во втором случае, это действие ведет себя как действие реплицировать 110, т.е. выполняет реплицирование конфигурации членов группы во все остальные члены группы.

Это действие очень удобно для добавления новых членов группы. Новые члены добавляются путем перетаскивания мышью на контекст Группы.

Тип действия: перетаскивание мышью 101

Имя действия: replicateOrAdd

Не интерактивный режим 99**:** Не поддерживается

Права доступа: Доступно на уровне 486 с правами доступа для *Оператора*.

КОНВЕРТИРОВАТЬ ДИНАМИЧЕСКУЮ ГРУППУ В СТАТИЧЕСКУЮ

Это действие добавляет каждый дочерний узел динамической 753 группы как статический дочерний узел в ту же группу так, чтобы позднее его можно было индивидуально удалить. Когда действие выполнено, выражение действительности группы опустошается (emptied) так, чтобы у него больше не было данамических дочерних узлов.

Достаточно просто создать группу объектов, соответствующих определенным критериям:

- создайте новую группу и задайте ее выражение достоверности для автоматического добавления соответствующих ему объектов;

- преобразуйте группу в статическую (static);
- добавьте или удалите определенные объекты так, чтобы они точно соответствовали Вашей группе.

Имя действия: convertToStatic

Не интерактивный режим: Не поддерживается

Права доступа: Доступно на уровне с правами доступа для *Инженера*.

Общие действия [?]

Удалить, редактировать права доступа, просмотр событий, поиск/фильтрация, показать статус

Состояния и иконки контекста

Иконка	Код	Состояние
	0	Обычная группа
	1	Группа, которая выполняет автоматическую репликацию

Дополнительная информация

Информация о контексте

Тип контекста: группа

Имя контекста: предоставляется пользователем

Описание контекста: предоставляется пользователем

Путь контекста: "usergroups.GROUP_NAME" для группы пользователя, "users.USER_NAME.devgroups.GROUP_NAME" для группы устройства

Контекстные маски: "usergroups.*" для групп пользователя, "users.*.devgroups.*" для групп устройства

Права доступа к контексту [?]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Просмотр списка членов группы. Доступ к списку членов группы не означает доступ к самим членам группы. См. подробности в Права доступа к членам группы. Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации. Управление членством группы.
Менеджер	Конфигурация и удаление группы.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [?]

Общие переменные: groupMembership (членство группы)67, validity (пригодность)68, activeAlerts (активные тревоги)

СВОЙСТВА

См описание переменной и ее полей здесь756.

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне486 с правами доступа для *Наблюдателя*, доступно для записи для *Менеджера*.

Формат49 записи:

Имя поля	Тип поля	Примечания
name	Строка	1 - 50 знаков
description	Строка	1 - 50 знаков
autoReplication	Булевое	
hideMembers	Булевое	
validityExpression	Строка	
validityListeners	Таблица данных	

СТАТУС ГРУППЫ

См. описание переменной и ее полей здесь757.

Имя переменной: groupStatus

Записи: 1

Права доступа: Доступно для чтения на уровне486 с правами доступа для *Наблюдателя*, доступно для записи для *Менеджера*.

Формат49 записи:

Имя поля	Тип поля	Примечание
enabled	Булевое	
variable	Строка	
expression	Строка	
statuses	Таблица данных	

ОПЦИИ КОПИРОВАНИЯ

См. описание переменной и ее полей здесь758.

Имя переменной: replication

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне486 с правами доступа для *Наблюдателя*, доступно для записи для *Менеджера*.

Формат49 записи:

Имя поля	Тип поля	Примечания
variable	Строка	
description	Строка	
replicate	Булевое	
useMaster	Булевое	
master	Строка	

СТАТИЧЕСКИЕ ЧЛЕНЫ ГРУППЫ

Эта переменная показывает членов группы, которые были вручную добавлены операторами устройств. Список включает только тех членов, которые доступны читающему эту переменную.

Отметим, что определенные статические члены группы могут также появляться в списке Пригодности (например, соответствовать выражению пригодности динамической группы[753]).

Имя переменной: staticMembers

Записи: 0...не ограничено.

Права доступа: Доступно для чтения на уровне[486] прав доступа *Отсутствует*

Формат[50] записи:

Имя поля	Тип поля	Примечания
context	Строка	Путь контекста статического члена группы.

СТАТУС ЧЛЕНОВ ГРУППЫ

Эта переменная показывает отдельные статусы всех членов группы. Эти отдельные статусы используются для расчета собранных статусов группы[754].

Имя переменной: memberStatus

Записи: 0... не ограничено.

Права доступа: Доступно для чтения на уровне[486] с правами доступа для *Наблюдателя.*

Формат[50] записи:

Имя поля	Тип поля	Примечания
context	Строка	Путь контекста члена группы.
status	Строка	Строка статуса члена группы.

Общие функции [?[70]]

ДОБАВИТЬ ЧЛЕН

Добавляет новый контекст члена в группу.

Имя функции: add

Права доступа: Доступно на уровне[486] с правами доступа для *Оператора*.

Записи ввода: 1

Формат 49 ввода:

Имя	Тип	Описание
context	Строка	Путь контекста для добавления.

Записи вывода: 0

Формат 49 вывода: нет

УДАЛИТЬ ЧЛЕН

Удаляет контекст члена из группы.

Имя функции: remove

Права доступа: Доступно на уровне 486 с правами доступа для *Оператора*.

Записи ввода: 1

Формат 49 ввода:

Имя	Тип	Описание
context	Строка	Путь контекста для удаления.

Записи вывода: 0

Формат 49 вывода: нет

ВЫЗВАТЬ ФУНКЦИЮ ДЛЯ ЧЛЕНОВ ГРУППЫ

Вызывает одну и ту же функцию контекста из каждого контекста члена.

Имя функции: call

Права доступа: Доступно на уровне 486 с правами доступа для *Оператора*.

Записи ввода: 1

Формат 49 ввода:

Имя	Тип	Описание
function	Строка	Имя функции для вызова.
parameters	Строка	Параметры ввода функции.

Записи вывода: 0...не ограничено

Формат 49 вывода:

Имя	Тип	Описание
context	Строка	Путь контекста.
successful	Boolean	Указывает, прошло ли выполнение успешно.
error	Строка	Текст сообщения об ошибке, если выполнение неуспешно.
return	Таблица данных	Вывод функции, если выполнение успешно.

Общие события [? 73]

Общие события: info (информация)[77], contextStatusChanged (статус изменен)[83]

17.27.1 Действие: создать датчик статуса группы

Это действие позволяет создавать датчик[2181], который будет указывать на статус группы, т.е. агрегированный статус всех членов группы. Статус группы определяется посредством двух параметров: **Выражения** и **Начального значения**.

Выражение статуса группы вычисляется для каждого контекста члена группы. Оно должно включать в себя две ссылки:

- `{env/previous}` - ссылка, относящаяся к результату предыдущего подсчета
- ссылка на данные текущего контекста, такого как `{.:status$connectionStatus}`

Во время первой оценки, {env/previous} превратится в **Начальное значение**.

Порядок действий:

1. Система подсказывает[90] пользователю ввести выражение для подсчета для каждого члена группа, и начальное значение для статуса группы.
2. Выражение статуса группы составляется на основании выше приведенных данных, после чего создается новый датчик.
3. Отображается уведомление[97], указывающее на имя нового датчика.

Имя действия:	createStatusTracker
Иконка действия:	
Не интерактивный режим[99]:	не поддерживается
Права доступа:	Доступно на уровне[486] с правами доступа для *Менеджера*.

17.28 Машинное обучение

Этот конеткст[41] является контейнером, содержащим все обучаемые модули[1599] для выбранного пользователя.

Уникальные действия [?[1450]]

СОЗДАТЬ (действие по умолчанию[88])

Это действие используется для создания нового обучаемого модуля. Позволяет пользователю указать основные свойства[870] для нового обучаемого модуля и настроить его сразу же после создания.

Тип действия:	Создать[105]
Права доступа:	Доступно на уровне[486] прав доступа *Менеджер*

Общие действия [?[1450]]

Создать на основе шаблона[105], копировать в дочерние контексты[111], импорт[108], экспорт[108], редактировать права доступа[106], просмотр событий[109], поиск/фильтрация[110], различные групповые действия[101] согласно контекстам потомков.

Состояния и иконки контекста

У этого контекста нет состояний[44]. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста 43: machineLearning

Имя контекста 42: machineLearning

Описание контекста 43: Машинное обучение

Путь контекста 42: users.USER_NAME.machineLearning

Маска контекста 44: users.*.machineLearning

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт обучаемого модуля.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [? 61]

У этого контекста нет общих переменных (свойств).

Общие функции [? 70]

Общие функции: makeCopy (сделать копию) 71, delete (удалить) 72

СОЗДАТЬ

Создает новый обучаемый модуль 1599

Имя функции: create

Права доступа: Доступно на уровне 486 прав доступа для *Менеджера*

Записи ввода: 1

Формат 50 ввода: Тот же, что формат переменной childInfo 1599 в контексте Обучаемый модуль 1599.

Записи вывода: 0

Формат 50 вывода: нет

ЭКСПОНЕНЦИАЛЬНОЕ СГЛАЖИВАНИЕ

Выполняет тройное экспоненциальное сглаживание по модели Хольта-Уинтерса.

Имя функции: exponentialSmoothing

Права доступа: Доступно на уровне 486 прав доступа для *Наблюдателя*

Записи ввода: 1

Формат 50 ввода:

Имя	Тип	Описание
data	Data Table	Таблица, содержащая, как минимум, одно числовое поле (т.е. зависимую переменную) с

		записями данных временных рядов
labelFieldName	String	Имя зависимой переменной (может быть NULL, если Таблица данных в поле "данные" имеет лишь одно поле) 1 - 50 символов Может быть NULL
valueSmoothingFactor	Integer	Коэффициент сглаживания значений серийных данных (должен быть между 0 и 1)
trendSmoothingFactor	Integer	Коэффициент сглаживания тренда (должен быть между 0 и 1)
seasonalSmoothingFactor	Integer	Коэффициент сглаживания сезонной компоненты (должен быть между 0 и 1)
seasonCycleLength	Integer	Длина сезонного цикла (например, 12 - для ежемесячных данных; 4 - для ежеквартальных)
numPredictedValues	Integer	Число значений для предсказания

Записи вывода: 0...не ограничено

Формат вывода: Динамический, включает все поля из таблицы данных в поле ввода "данных", а также поле с сглаженными и предсказанными значениями (если numPredictedValues > 0):

Имя	Тип	Описание
predictedValue	Double	Может быть NULL

Обратите внимание, что функции требуются 2*numPredictedValues + 1 точек данных для инициализации, чтобы вычислять сглаженные (а также предсказанные) значения.

Общие события [?]

Общие события: info (информация)

17.29 Модели

Этот контекст является контейнером, содержащим все модели для определенного пользователя.

Уникальные действия [?]

СОЗДАТЬ (действие по умолчанию)

Это действие используется для создания новой модели. Оно позволяет пользователю определить основные свойства для новой модели и настроить их сразу после создания.

Тип действия: создать

Права доступа: Доступно на уровне 486 прав доступа для *Менеджера*

Общие действия [? 1450]

Создать на основе шаблона 105, копировать в дочерние контексты 111, импорт 108, экспорт 108, редактировать права доступа 106, просмотр событий 109, поиск/фильтрация 110, различные групповые действия 101 согласно контекстам потомков.

Состояния и иконки контекста

У этого контекста нет состояний 44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста 43: models

Имя контекста 42: models

Описание контекста 43: модели

Путь контекста 42: users.USER_NAME.models

Маска контекста 44: users.*.models

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт модели.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [? 61]

У этого контекста нет общих переменных (свойств).

Общие функции [? 70]

Общие функции: makeCopy (сделать копию) 71, delete (удалить) 72

СОЗДАТЬ

Создает новую модель.

Имя функции: create

Права доступа: Доступно на уровне 486 прав доступа для *Менеджера*

Записи ввода: 1

Формат 50 ввода: Тот же, что формат переменной childInfo 1540 в контексте Модель 1539.

Записи вывода: 0

Формат 50 вывода: нет

Общие события [?73]

Общие события: info (информация)77

17.30 Модель

Этот контекст41 позволяет получить доступ и управлять одной моделью810.

Уникальные действия [?1450]

НАСТРОИТЬ

Данное действие настроить105 используется для редактирования свойств815 модели.

Изменение поля **Имя** во время этой операции меняет название данного контекста. Это может привести к сбою в работе других компонентов системы, использующих имя/путь контекста в качестве основного идентификатора.

Тип действия: настроить105

Общие действия [?1450]

Удалить106, сделать копию109, реплицировать110, редактировать права доступа контекста106, просмотреть события109, просмотреть статус111

Состояния и иконки контекста

Иконка	Код	Состояние
	0	Это относительная модель.
	1	Это абсолютная модель.

Дополнительная информация

Информация о контексте

Тип контекста43: model

Имя контекста42: предоставляется пользователем

Описание контекста43: предоставляется пользователем

Путь контекста42: users.USER_NAME.models.MODEL_NAME

Маска контекста44: users.*.models.*

Права доступа к контексту [?44]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Просмотр конфигурации модели. Мониторинг основных событий. Просмотр статуса.
Оператор	Те же, что у Наблюдателя.
Менеджер	Удаление модели.
Инженер	Те же, что у Менеджера.

Администратор	Конфигурация модели.

Общие переменные (свойства) [?61]

Абсолютные811 модели добавляют декларируемые ими переменные к собственному контексту. Таким образом, на каждую запись создается одна переменная в таблице модели Переменные817.

Общие переменные: groupMembership (членство группы)67, validity (пригодность)68, activeAlerts (активные тревоги)

СВОЙСТВА

См. описание переменной и ее поле здесь815.

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне486 с правами доступа для *Наблюдателя*, доступно для редактирования на уровне с правами доступа для *Менеджера*

Формат50 записи:

Имя поля	Тип поля	Примечания
name	Строка	1 - 50 символов
description	Строка	1 - 50 символов
template	Строка	
type	Строка	
validityExpression	Строка	
validityListeners	Таблица данных	
defaultContext	Строка	
containerType	Строка	
containerTypeDescription	Строка	
containerName	Строка	
objectType	Строка	
objectTypeDescription	Строка	
enabled	Булевое	
normalConcurrentBindings	Целое	
maximumConcurrentBindings	Целое	

maximumBindingQueueLength	Целое	

ПЕРЕМЕННЫЕ

Эта переменная содержит отпределения переменных модели 817.

Имя переменной: modelVariables

Записи: 0... не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для редактирования на уровне с правами доступа для *Менеджера*

Формат 50 записи:

Имя поля	Тип поля	Примечания
name	Строка	
description	Строка	
format	Талица данных	
writable	Булевое	
help	Строка	Допускает значение null
group	Строка	Допускает значение null
readPermissions	Строка	
writePermissions	Строка	
updateHistoryStorageTime	Длинное	
addPreviousValueToVariableUpdateEvent	Boolean	

ФУНКЦИИ

Эта переменная содержит определения функций модели 818.

Имя переменной: modelFunctions

Записи: 0... не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для редактирования на уровне с правами доступа для *Менеджера*

Формат 50 записи:

Имя поля	Тип поля	Примечания
name	Строка	
description	Строка	

inputformat	Таблица данных	
outputformat	Таблица данных	
help	Строка	Допускает значение null
group	Строка	Допускает значение null
permissions	Строка	
implementation	Строка	
plugin	Строка	Допускает значение null

СОБЫТИЯ

Эта переменная содержит определения событий модели 819.

Имя переменной: modelEvents

Записи: 0... не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для редактирования на уровне с правами доступа для *Менеджера*

Формат 50 записи:

Имя поля	Тип поля	Примечания
name	Строка	
description	Строка	
format	Таблица данных	
help	Строка	Допускает значение null
level	Целое	
group	Строка	Допускает значение null
permissions	Строка	
firePermissions	Строка	
historyStorageTime	Длинное	

НАБОРЫ ПРАВИЛ

Эта переменная содержит наборы правил 813 модели.

Имя переменной: ruleSets

Записи: 0... не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для редактирования на уровне с правами доступа для *Менеджера*

Формат 50 записи:

Имя поля	Тип поля	Примечания
name	Строка	
description	Строка	
type	Целое	
rules	Таблица данных	

ПРИВЯЗКИ

Эта переменная содержит привязки модели 821.

Имя переменной: bindings

Записи: 0... не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для редактирования на уровне с правами доступа для *Менеджера*

Формат 50 записи:

Имя поля	Тип поля	Примечания
target	Строка	
expression	Строка	
activator	Строка	
condition	Строка	
onstartup	Булевое	
onevent	Булевое	
periodically	Булевое	
period	Длинное	

СТАТИСТИКА ПУЛА ПОТОКОВ

Возвращает статистическую информацию о пуле потоков модели.

Имя переменной: threadPoolStatus

Записи: 0... не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*

Формат 50 записи:

Имя поля	Тип поля	Примечания
activeCount	Целое	Количество активных задач.
completedCount	Длинное	Количество завершенных задач.

totalCount	Длинное	Общее количество задач.
coreSize	Целое	Размер ядра пула.
largestSize	Целое	Самый большой (пиковый) размер пула.
maximumSize	Целое	Максимально разрешенный размер пула.
queueLength	Целое	Длина очереди задач.
rejectedCount	Длинное	Количество отклоненных заданий.

СТАТИСТИКА

Эта переменная позволяет просматривать статистику переменных модели, т.е. данные, собранные каналами статистики[821] переменных модели.

Имя переменной: statistics

Записи: 0... не ограничено

Права доступа: Доступно для чтения на уровне[486] с правами доступа для *Наблюдателя*

Формат[50] записи:

Имя поля	Тип поля	Примечания
name	Строка	Имя канала.
variable	Строка	Имя переменной, на которой основан канал.
statistics	Таблица данных	Краткие статистические данные.

Общие функции [?[70]]

Абсолютные[811] модели добавляют декларируемые ими функции к собственному контексту. Таким образом, для каждой записи в таблице функций[818] модели создается одна функция. Также одна функция создается для каждой записи в таблице наборов правил[819] модели.

Общие события [?[73]]

Абсолютные[811] модели добавляют декларируемые ими события в собственный контекст. Таким образом, для каждой записи в таблице событий[819] модели создается одно событие.

Общие события: info (информация)[77]

ВЫПОЛНЕНИЕ ПРИВЯЗОК

Это событие генерируется каждый раз при выполнении привязки модели.

Имя события: bindingExecution

Права доступа: Доступно на уровне[486] с правами доступа для *Наблюдателя*

Период действия: Непостоянный

Записи: 1

Формат[50] записи:

Имя поля	Тип поля	Примечания
context	Строка	Контекст экземпляра модели, т.е. контекст абсолютной модели, контекст, к которому привязана относительная модель, или контекст экземпляра инстанцируемой модели.
target	Строка	Цель привязки.
expression	Строка	Выражение привязки.
value	Строка	Представление результата выражения привязки в виде строки.
activator	Строка	Активатор привязки.
condition	Строка	Состояние привязки.
execution	Строка	Режим выполнения привязки (при запуске, при событии или периодически).
cause	Строка	Причина выполнения привязки (ссылка на событие или измененную переменную, которая вызвала выполнение).

ОШИБКА ПРИВЯЗКИ

Это событие генерируется каждый раз, когда привязка модели не выполняется и выдает ошибку.

Имя события: bindingError

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*

Период действия: Непостоянный

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
context	Строка	Контекст экземпляра модели, т.е. контекст абсолютной модели, контекст, к которому привязана относительная модель, или контекст экземпляра инстанцируемой модели.
target	Строка	Цель привязки.
expression	Строка	Выражение привязки.
activator	Строка	Активатор привязки.
execution	Строка	Режим выполнения привязки (при запуске, при событии или периодически).
cause	Строка	Причина выполнения привязки (ссылка на событие или измененную переменную, которая вызвала выполнение).
error	Строка	Текст сообщения об ошибке.
stack	Таблица данных	Таблица, которая содержит трассировку стека ошибки привязки.

Общие события: info (информация) 77

17.31 Запросы

Этот контекст[41] является контейнером, который содержит все запросы[829] для определенного пользователя.

Уникальные действия [?1450]

- Выполнять сырые запросы[1548]

Другие уникальные действия:

СОЗДАТЬ (действие по умолчанию[88])

Это действие используется для создания новых запросов. Оно позволяет пользователю задать основные свойства[837] нового запроса и настроить его сразу же после создания.

Тип действия:	создать[105]
Права доступа:	Доступно на уровне[486] с правами доступа для *Менеджера*.

Общие действия [?1450]

Создать на основе шаблона[105], копировать в дочерние контексты[111], импорт[108], экспорт[108], редактировать права доступа[106], просмотр событий[109], поиск/фильтрация[110], различные групповые действия[101] согласно контекстам потомков.

Относящиеся к переменным действия [?102]

СОЗДАТЬ ЗАПРОС

Это действие создает новый запрос, который выбирает:

- Значение этой переменной из разных контекстов, или
- Историю этой переменной, или
- Статистику[718] этой переменной

Поток событий:

1. Выберите, на чем должен быть основан запрос: **текущие значения** переменной, **история переменной** или **статистика переменной**. Это сформирует условие запроса ОТ.
2. Если на шаге 1 была выбрана **история переменной**, выберите временные рамки с целью извлечения исторических данных для: **всего времени** или **конкретных временных рамок**.
3. Если на шаге 2 были выбраны **конкретные временные рамки**, определите набор исторических/статистических данных для включения в результаты запроса.
4. Если на шаге 1 была выбрана **статистика переменных**, выберите статистический канал, чьи данные будут включены в результаты запроса. Создание запроса к этому моменту будет завешено.
5. Выберите ресурс или маску ресурсов, откуда запрос должен выбирать данные. Этот ресурс/маска будут включены в условие ОТ.
6. Если переменнная в одну строку была изначально выбрана для построения запроса, выберите между построением запроса для **одной переменной** или **множества переменных**. Переменные будут записаны в условие запроса ОТ.
7. Если на шаге 6 было выбрано **множество переменных**, выберите другие переменные в одну строку для включения в результаты запроса.
8. Выберите, какие поля переменных нужно включить в результаты запроса: **все поля** выбранных переменных или только **выбранные поля**. Поля будут занесены в условие запроса ВЫБРАТЬ.
9. Если на шаге 8 была выбрана опция **выбранные поля**, выберите одно или более полей для включения в результаты запроса.
10. Добавьте одно или более **условий** к условию запроса ГДЕ. Каждое условие определяется **переменной**, операцией **сравнения**, закодированным в строке **значением**, с которым будет сравниваться переменная, и **операцией** ИЛИ/И, применяемой к другим условиям.

11. В этот момент создается запрос, и его конфигурация открыта для редактирования.

Имя действия:	queryForVariable
Неинтерактивный режим:	Не поддерживается
Права доступа:	Доступно на уровне с правами доступа для *Менеджера*

Состояния и иконки контекста

У этого контекста нет состояний. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста: запросы

Имя контекста: запросы

Описание контекста: запросы

Путь контекста: users.USER_NAME.queries

Контекстная маска: users.*.queries

Права доступа к контексту [?]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт запроса.
Инженер	Те же, что у Менеджера.
Администратор	Выполнение прямого запроса в СУБД.

Общие переменные (свойства) [?]

У этого контекста нет общих переменных (свойств).

Общие функции [?]

Общие функции: makeCopy (скопировать), delete (удалить)

СОЗДАТЬ

Создает новый запрос.

Имя функции:	create
Права доступа:	Доступно на уровне с правами доступа для *Менеджера*.
Записи ввода:	1
Формат ввода:	Аналогичный формату переменной childInfo в контексте Запрос.
Записи вывода:	0
Формат вывода:	нет

Общие события [?]

Общие события: info (информация) 77

17.31.1 Действие: выполнить прямой запрос к СУБД

Это действие Вызывает функцию 103 для выполнения прямого SQL-запроса к базе данных 692 AtomMind Server . Оно вызывает функцию Выполнить прямой запрос к СУБД 1579 из корневого контекста.

Этот запрос оказывает непосредственное влияние на нижележащую базу данных AtomMind Server, обходя процессор языка запроса AtomMind Server 829 и проверку прав доступа AtomMind Server 477. Неправильные запросы могут испортить данные и привести к тому, что инсталляция AtomMind Server окажется нерабочей.

Тип действия:	вызвать функцию 103
Имя действия:	executeNativeQuery
Права доступа:	Доступно на уровне 486 с правами доступа для *Администратора*.

17.32 Запрос

Этот контекст 41 предоставляет Вам доступ к одному запросу 829 и позволяет им управлять.

Уникальные действия [?1450]

- Выполнить запрос 1550 (действие по умолчанию 88)

Другие уникальные действия:

ЗАПУСТИТЬ В РЕЖИМЕ ОТЛАДКИ

Это действие используется для выполнения запроса в режиме отладки 838. У него есть лишь одно отличие от действия Выполнить запрос 1550: если запрос производит вывод отладки, *отчет об отладке* отображается пользователю сразу после выполнения запроса при помощи GUI процедуры редактировать данные 90 в режиме реального времени. Когда пользователь закрывает отчет об отладке, отображается результат запроса.

Имя действия:	debug
Не интерактивный режим 99:	не поддерживается
Права доступа:	Доступно на уровне 486 с правами доступа для *Наблюдателя*.

НАСТРОИТЬ

Это действие используется для редактирования свойств 837 Запроса.

Изменение поля **Имя** во время этой операции меняет название данного контекста. Это может привести к сбою в работе других компонентов системы, использующих имя/путь контекста в качестве основного идентификатора.

Тип действия:	Настроить 105

Общие действия [?1450]

Удалить 106, создать копию 109, реплицировать 110, редактировать права доступа 106, просмотр событий 109, показать статус 111

Состояния и иконки контекста

У этого контекста нет состояний 44. Он представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста[43]: запросы

Имя контекста[42]: предоставляется пользователем

Описание контекста[43]: предоставляется пользователем

Путь контекста[42]: users.USER_NAME.queries.QUERY_NAME

Контекстная маска[44]: users.*.queries.*

Права доступа к контексту [?[44]]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Выполнение запроса. Мониторинг основных событий. Просмотр состояния.
Оператор	Просмотр конфигурации.
Менеджер	Конфигурация и удаление запроса.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [?[61]]

Общие переменные: groupMembership (членство группы)[67], activeAlerts (активные тревоги)

СВОЙСТВА ЗАПРОСА

См описание переменной и ее полей здесь[837].

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне[486] с правами доступа для *Наблюдателя*, доступно для записи с правами *Менеджера*.

Формат[49] записи:

Имя поля	Тип поля	Примечания
name	Строка	1 - 50 знаков
description	Строка	1 - 50 знаков
parameterized	Булевое	
query	Строка	
parameterized	Строка	
outFormat	Таблица данных	

fields	Таблица данных	
disableEditableResult	Булевое	

РЕЗУЛЬТАТЫ ЗАПРОСА

Эта переменная используется для выполнения запроса и получения его результата.

Имя переменной:	data
Записи:	0...не ограничено
Права доступа:	Доступно для чтения/записи на уровне 486 с правами доступа для *Наблюдателя.*
Формат 49 записи:	динамический

Общие функции [? 70]

ВЫПОЛНИТЬ ЗАПРОС

Выполняет запрос и возвращает его набор результатов.

Имя функции:	execute
Права доступа:	Доступно на уровне 486 прав доступа *Отсутствует*
Записи ввода:	1
Формат 50 ввода:	Динамический, может меняться, когда редактируются свойства 837 запроса. Если запрос не параметризован 841, формат ввода пуст, т.е. возможно выполнить запрос без определения ввода. Если запрос параметризован, формат ввода этой функции соотетствует формату, определенному в параметризаторе запроса.
Записи вывода:	0... не ограничено
Формат 50 вывода:	динамический

Общие события [? 73]

Общие события: info (информация) 77

17.32.1 Действие: выполнить запрос

Это действие выполняет запрос и отображает его результат пользователю. Различают простой и параметризованный 841 запросы. Выполняются они следующим образом:

Для простого (не параметризованного) запроса:

1. Запрос выполнен

2. Результат запроса отображается пользователю при помощи GUI процедуры редактировать свойства 91. Если результат запроса содержит некоторые редактируемые поля, пользователь может отредактировать их и сохранить изменения. Все сохраненные данные будут сохранены в контексты 41, их которых изначально последовал запрос.

Для параметризованного запроса:

1. Система предлагает 90 пользователю ввести параметры запроса

2. Параметры, введенные на этапе 1, используются для получения окончательного текста параметризованного запроса.

3. Запрос выполнен.

4. Результат запроса отображается пользователю при помощи GUI процедуры редактировать свойства 91. Если результат запроса содержит редактируемые поля, пользователь может отредактировать их и сохранить изменения. Все измененные данные будут сохранены в контексты 41, их которых изначально последовал запрос.

Если запрос **параметризован**, или флажок **отключить редактируемый результат** активирован в свойствах запроса 837, результат запроса отображается при помощи GUI процедуры редактировать данные 90 вместо действия Редактировать свойства.

Если во время выполнения запроса или параметризации возникают ошибки, они отображаются пользователю при помощи GUI процедуры показать ошибку 95.

Имя действия:	execute
Не интерактивный режим 99:	не поддерживается
Права доступа:	Доступно на уровне 486 с правами доступа для *Наблюдателя*
Параметры 102 выполнения:	С результатом запроса расположение окна С результатом запроса свойства инструментальных панелей 926

17.33 Отчеты

Этот контекст 41 является контейнером, который содержит все отчеты 928 для отдельного пользователя.

Уникальные действия [? 1450]

СОЗДАТЬ НОВЫЙ ОТЧЕТ (действие по умолчанию 88)

Это действие используется для создания новых отчетов.

Порядок действий:

1. Задать 90 основные свойства 935 нового отчета.
2. Установить 90 различные свойства шаблона отчета 930.
3. Шаблон отчета генерируется 930 сервером, и создается новый контекст отчета 1553.
4. Определите 89, нужно ли Вам сейчас модифицировать шаблон отчета.
5. *[Факультативно]* Отредактируйте шаблон отчета в Редакторе отчетов 416. Этот этап поддерживается только в AtomMind Cliente 359.

Иконка действия:	
Имя действия:	create
Не интерактивный режим 99:	не поддерживается
Права доступа:	Доступно на уровне 486 с правами доступа для Менеджера.

Общие действия [? 1450]

Создать на основе шаблона 105, копировать в дочерние контексты 111, импорт 108, экспорт 108, редактировать права доступа 106, просмотр событий 109, поиск/фильтрация 110, различные групповые действия 101 согласно контекстам потомков.

Действия, относящиеся к переменной [? 102]

СОЗДАТЬ ОТЧЕТ

Это действие создает новый отчет, который обрабатывает и отображает текущие, исторические и статистические значения переменной.

Схема работы действия:

- выбрать, на чем должен быть основан отчет: таблица **история переменных**, **текущее значение** табличной переменной или **использование** переменной типа "счетчик", сгруппированной по временным промежуткам.
- создать новый отчет с выражением данных источника 929, указывающих на переменную.
- подсказать пользователю задать свойства шаблона отчета 930.
- получить значение переменной и сгенерировать 930 шаблон отчета 930 согласно его формату.

Имя действия:	reportForVariable
НЕ интерактивный Mode 99:	не поддерживается
Права доступа:	Доступно на уровне 486 с правами доступа для *Менеджера*.

Действия, связанные с событием [? 102]

СОЗДАТЬ ОТЧЕТ

Это действие создает новый отчет, который отображает историю событий.

Схема работы действия:

- подсказать пользователю задать **Период** и **Максимальное количество событий**.
- подсказать пользователю задать свойства шаблона отчета 930.
- создать новый отчет с Выражением данных источника 929, указывающим на функцию Получить историю события 1516 контекста События.
- получить небольшую часть ("образец") истории событий и сгенерировать 930 шаблон отчетов 930 согласно его формату.

Имя действия:	reportForEvent
Не интерактивный режим 99:	не поддерживается
Права доступа:	Доступно на уровне 486 с правами доступа для *Менеджера*.

Состояния и иконки контекста

У этого контекста нет состояний 44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста 43: отчет

Имя контекста 42: отчеты

Описание контекста 43: отчеты

Путь контекста 42: users.USER_NAME.reports

Контекстная маска 44: users.*.reports

Права доступа к контексту [? 44]

Уровень	Описание

Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт отчетов.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [?61]

У этого контекста нет общих переменных (свойств).

Общие функции [?70]

Общие функции: makeCopy (скопировать)71, delete (удалить)72

СОЗДАТЬ

Создает новый отчет.

Имя функции: create

Права доступа: Доступно на уровне486 с правами доступа для *Менеджера*.

Записи ввода: 1

Формат49 ввода: Аналогичный формату переменной childInfo1556 в контексте Отчет1553.

Записи вывода: 0

Формат49 вывода: нет

Общие события [?73]

Общие события: info (информация)77

17.34 Отчет

Этот контекст41 предоставляет Вам доступ к отдельному отчету928 и позволяет им управлять.

Уникальные действия [?1450]

НАСТРОИТЬ

Это действие используется для редактирования свойств935 отчета.

> Изменение поля **Имя** во время этой операции меняет название данного контекста. Это может привести к сбою в работе других компонентов системы, использующих имя/путь контекста в качестве основного идентификатора.

Тип действия: настроить105

РЕДАКТИРОВАТЬ ШАБЛОН ОТЧЕТА

Это действие поддерживается только в AtomMind Client359. Оно запускает Редактор отчетов416 и позволяет редактировать шаблон отчета.

Имя действия: editTemplate

Неинтерактивный режим 99:	Не поддерживается
Права доступа:	Доступно на уровне 486 с правами доступа для *Менеджера*.

ПОКАЗАТЬ ИСХОДНЫЕ ДАННЫЕ

Это действие представляет пользователю данные, на которые ссылается Выражение данных источника 929, при помощи GUI процедуры редактировать данные 90. Данное действие может оказаться полезным при отладке отчета, поскольку оно отображает сырые данные, используемые для заполнения шаблона отчетов.

Имя действия:	viewData
Не интерактивный режим 99:	не поддерживается
Права доступа:	Доступно на уровне 486 с правами доступа для *Наблюдателя*.

ПРОСМОТРЕТЬ ИСТОРИЮ

Это действие используется для просмотра всех версий 1557 данного отчета, которые были заполнены ранее и сохранены в базе данных.

Удаление строк из таблицы **Отчет по истории**, которое открывается этим действием, будет постоянно стирать выбранные исторически отчеты в базе данных.

Тип действия:	настроить 105
Имя действия:	viewHistory
Права доступа:	Доступно на уровне 486 с правами доступа для *Менеджера*

ЭКСПОРТИРОВАТЬ В ФАЙЛ СЕРВЕРА

Это действие вызова функции 103 сохраняет заполненный отчет в файл на машине AtomMind Server. Отчет заполняется данными в момент выполнения данного действия и, таким образом, всегда содержит актуальные данные.

Отчеты могут сохраняться в одном из следующих форматов:

- Формат документов для печати (PDF)
- Microsoft Excel (XLS)
- Расширенный текстовый формат (RTF)
- Файл данных с разделителями-запятыми (CSV)
- Расширяемый язык маркировки (XML)
- Open Office (ODT)

Также можно выбрать разделитель полей для CSV формата.

Тип действия:	вызвать функцию 103
Имя действия:	export
Права доступа:	Доступны на уровне 486 прав доступа для *Оператора*

ЭКСПОРТИРОВАТЬ В ФАЙЛ КЛИЕНТА

Это действие вызова функции 103 открывает заполненный отчет как Блок данных, возволяя сохранять его в файл на AtomMind Client или на локальной машине веб пользователя. Отчет заполняется данными в момент выполнения данного действия и, таким образом, всегда содержит актуальные данные.

Отчеты могут сохраняться в одном из следующих форматов:

- Формат документов для печати (PDF)
- Microsoft Excel (XLS)
- Расширенный текстовый формат (RTF)

- Файл данных с разделителями-запятыми (CSV)
- Расширяемый язык маркировки (XML)
- Open Office (ODT)

Также можно выбрать разделитель полей для CSV формата.

Тип действия:	Call Function 103
Имя действия:	exportRemote
Права доступа:	Доступны на уровне 486 прав доступа для *Оператора*

ОТПРАВИТЬ ОТЧЕТ ПО E-MAIL

Это действие вызова функции 103 отправляет отчет по e-mail получателям, заданным в виде списка e-mail адресов, разделенных запятой. Отчет заполняется данными в момент выполнения данного действия и, таким образом, всегда содержит самые свежие данные.

Отчеты могут отправляться в одном из следующих форматов:

- Формат документов для печати (PDF)
- Microsoft Excel (XLS)
- Расширенный текстовый формат (RTF)
- Файл данных с разделителями-запятыми (CSV)
- Расширяемый язык маркировки (XML)
- Open Office (ODT)

Также можно выбрать разделитель полей для CSV формата.

Может оказаться очень удобным запланировать 823 это действие так, чтобы оно активировало периодическую отправку отчета по e-mail (например, ежедневно или еженедельно).

Тип действия:	Call Function 103
Имя действия:	sendByMail
Права доступа:	Доступно на уровне 486 с правами доступа для *Менеджера*.

ПОКАЗАТЬ ОТЧЕТ (Действие по умолчанию 88)

Это действие используется для построения и просмотра отчета.

Поток действия:

1. *[Не обязательно]* Система подсказывает пользователю ввести параметры отчета, которые используют UI процедуру редактировать данные 90.
2. Шаблон отчета составляется на сервере и заполняется данными.
3. Отчет показывается пользователю при помощи UI процедуры показать отчет 97.

Имя действия:	show
Иконка действия:	
Не интерактивный режим 99:	не поддерживается
Права доступа:	Доступно на уровне 486 с правами доступа для *Наблюдателя*.
Параметры 102 выполнения:	Отчет о расположении окна Отчет о свойствах инструментальных панелей 926

ЗАПЛАНИРОВАТЬ ОТПРАВКУ ОТЧЕТА

Это действие вызова функции 103 создает новую запланированную задачу 823 для периодической отправки заполненного отчета по e-mail в формате, заданном пользователем.

Более подробно см. Отправить отчет по E-mail 1553

Имя действия: schedule

Иконка действия: 

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*.

Общие действия [? 1450]

Удалить 106, создать копию 109, реплицировать 110, редактировать права доступа 106, просмотр событий 109, показать статус 111

Состояния и иконки контекста

Иконка	Код	Состояние
	0	Этот отчет относительный.
	1	Этот отчет абсолютный.

Дополнительная информация

Информация о контексте

Тип контекста 43: report

Имя контекста 42: предварительно определено, для каждого отчета разное.

Описание контекста 43: предварительно определено, для каждого отчета разное.

Путь контекста 42: users.USER_NAME.reports.REPORT_NAME

Контекстная маска 44: users.*.reports.*

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Запуск отчета. Просмотр исходных данных отчета. Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Конфигурация и удаление отчета.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [? 61]

Общие переменные: groupMembership (членство группы) 67, validity (пригодность) 68, activeAlerts (активные тревоги)

СВОЙСТВА ОТЧЕТА

См. описание переменной и ее полей здесь 935.

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
name	String	1 - 50 знаков
description	String	1 - 100 знаков
expression	String	
template	String	
type	Integer	
validityExpression	String	
validityListeners	Data Table	
defaultContext	String	

ИСТОРИЯ ОТЧЕТА

Предоставляет доступ к экземплярам этого отчета, которые заполнялись ранее и сохранялись в базе данных. Новая запись в этой таблице создается каждый раз, когда заполняется отчет, но только если включен флажок отчета сохранить историю 935.

Системные операторы могут удалять истории отчетов, убирая определенные записи из этой таблицы и сохраняя изменения на сервере.

Имя переменной: history

Записи: 0... не ограничено

Права доступа: Доступны для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступны для записи на уровне с правами доступа для *Менеджера*

Формат 50 записи:

Имя поля	Тип поля	Примечания
date	Date	Дата создания отчета.
user	String	Пользователь 478 системы, который заполнил отчет.
report	Data Block	Данные отчета. Могут открываться в Обозревателе отчетов 415.

ДОПОЛНИТЕЛЬНЫЕ ПАРАМЕТРЫ ОТЧЕТА

См. описание переменной и ее полей здесь 936.

Имя переменной: Parameters

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
parameter	String	
value	String	

Общие функции [? 70]

ЭКСПОРТИРОВАТЬ В ФАЙЛ СЕРВЕРА

Сохраняет отчет в файл на AtomMind Server. Файл будет расположен в установочной директории AtomMind Server до тех пор, пока обозначенный путь абсолютен.

Имя функции: export

Права доступа: Доступно на уровне 486 прав доступа для *Менеджера*

Записи ввода: 1

Формат 50 ввода:

Имя	Тип	Описание
file	String	Путь конечного файла (абсолютного или относительного, с расширением или без расширения).
format	Integer	Формат отчета.
fieldDelimiter	String	Разделитель CSV полей.
targetContextMask	String	Маска контекстов, для которой генерируется относительный 932 отчет.

Записи вывода: 0

Формат 50 вывода: Нет

ОТПРАВИТЬ ОТЧЕТ ПО EMAIL

Заполняет отчет данными и отправляет его по email 1553.

Имя функции: sendByMail

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*.

Записи ввода: 1

Формат 49 ввода:

Имя	Тип	Описание
recipients	String	Список получателей email, разделенных запятой.
format	Integer	Формат отчета.
fieldDelimiter	String	Разделитель CSV полей.

targetContextMask	String	Маска контекстов, для которых генерируется относительный[932] отчет.

Записи вывода: 0

Формат[49] вывода: Нет

Общие события [?[73]]

Общие события: info (информация)[77]

17.35 Корневой контекст

Этот контекст[41] является корнем контекстного дерева AtomMind Server. У него есть некоторые основные действия[87], которые используются для осуществления контроля за сервером.

Уникальные действия [?[1450]]

НАСТРОИТЬ СЕРВЕР

Это действие используется для просмотра или редактирования опций глобальной конфигурации[179] AtomMind Server. Его единственное отличие от стандартного действия настроить[105] заключается в том, что оно подсказывает[89] пользователю перезагрузить сервер, когда сохранены новые параметры (перезагрузка необходима для применения изменений). Дополнительную информацию о настройке сервера можно найти здесь[177].

Тип действия: настроить[105]

Имя действия: configureServer

Иконка действия: 

Права доступа: Доступно на уровне[486] с правами доступа для *Администратора*.

ПРОСМОТРЕТЬ ИНФОРМАЦИЮИ О СЕРВЕРЕ

Это действие отображает[91] информацию о среде выполнения AtomMind Server. Оно может оказаться полезным для отладки сервера и разрешения проблем, связанных с производительностью.

Доступно несколько разделов:

- **Статус сервера**. Различная информация о рабочем статусе сервера в режиме реального времени, например, о времени запуска или использовании памяти.
- **Среда сервера**. Информация о среде выполнения Java и о машинной и операционной системах, на которой запущен контекст.
- **База данных**. Информация о базе данных SQL, которая используется в текущей установке AtomMind в качестве конфигурации и хранилища событий.
- **Статистика контекстов**. Список всех контекстов и количеств переменных, функций, событий и действий в каждом контексте.
- **Лицензионная информация**. Информация о лицензии AtomMind.
- **Активные плагины**. Информация о плагинах[207], которые AtomMind Server использует в настоящий момент.
- **Активные клиентские подключения**. Список подключенных в настоящий момент клиентов, включая удаленные адреса и тип/время подключения.
- **Потоки**. Список потоков сервера, включая их состояние и трассировку стека. Эту информацию может запросить группа тех. поддержки AtomMind.
- **Статистика пулов потоков**. Список активных и законченных задач для каждого пула серверных потоков. Эта информация может оказаться полезной при анализе нагрузки на процессор сервера.
- **Статистика правил обработки событий.** Деталированная информация о событиях, обрабатываемых с помощью правил обработки событий[196]: количество отклоненных, хранимых и дедуплицированных событий.

- **Детали очереди событий.** Когда очередь событий AtomMind Server не пустует, то здесь указываются детали событий, количество событий в очереди для каждого контекста.

Тип действия: настроить 105 (режим только для чтения)

Имя действия: viewServerInfo

Иконка действия: 

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*.

СОЗДАТЬ РЕСУРСЫ

Это действие используется для создания предварительно сформированных ресурсов (тревог, отчетов, виджетов и пр.), которые включены в дистрибутив AtomMind. См. Управление предварительно созданными ресурсами 208.

Имя действия: createResources

Иконка действия: 

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*.

УДАЛИТЬ РЕСУРСЫ

Это действие используется для удаления ранее созданных ресурсов (тревог, отчетов, виджетов и т.д.), включенных в дистрибутив AtomMind. См. управление ранее созданными ресурсами 208 для получения более подробной информации.

Имя действия: deleteResources

Иконка действия: 

Права доступа: Доступны на уровне 486 с правами доступа для *Менеджера*

ОСТАНОВИТЬ СЕРВЕР

Это действие позволяет остановить AtomMind Server. Существуют два варианта: немедленное и отложенное выключение. В случае отложенного выключения все активные операторы информируются о времени и причине выключения. Это позволяет им отложит текущие задачи в сторону и сохранить все изменения перед выключением.

Тип действия: Вызвать функцию 103

Имя действия: stop

Иконка действия: 

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*.

ПЕРЕЗАПУСТИТЬ СЕРВЕР

Это действие позволяет перезапустить AtomMind Server. Существует два варианта: немедленный или отложенный перезапуск. В случае отложенного перезапуска все активные операторы информируются о времени и причине перезапуска. Это позволяет им отложит текущие задачи в сторону и сохранить все изменения перед перезапуском.

Автоматический перезапуск возможен только в том случае, когда AtomMind Server запущен в сервисном режиме 159. В других случаях сервер следует остановить и запустить вновь вручную.

Тип действия: Вызвать функцию 103

Имя действия: restart

Иконка действия: 

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*.

НАЧАТЬ РЕЖИМ ОБСЛУЖИВАНИЯ

Это действие заставляет сервер войти в режим обслуживания 172.

Тип действия: Вызвать функцию 103

Имя действия: startMaintenanceMode

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*.

ЗАКОНЧИТЬ РЕЖИМ ОБСЛУЖИВАНИЯ

Это действие заставляет сервер выйти из режима обслуживания 172.

Тип действия: Вызвать функцию 103

Имя действия: stopMaintenanceMode

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора.*

УДАЛЕННОЕ ОБНОВЛЕНИЕ

Это действие вызов функции 103 используется для удаленного обновления AtomMind Server. Более подробно об этом см. Удаленное обновление 156.

Тип действия: Вызвать функцию 103

Имя действия: upgrade

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора.*

СМЕНИТЬ ПАРОЛЬ

Это действие вызов функции 103 используется для смены пароля авторизованного в текущий момент пользователя.

Тип действия: вызвать функцию 103

Имя действия: changePassword

Права доступа: Доступно на уровне 486 с правами доступа для *Оператора*.

ПОИСК

Это действие используется для получения доступа к диалогу универсального поиска 1424.

Имя действия: search

Иконка действия: 

Права доступа: Доступно на уровне 486 прав доступа *Нет прав*.

ПОКАЗАТЬ ИСТОРИЮ ПЕРЕМЕННОЙ

Это действие вызывает функцию 103 и используется для доступа и просмотра обновленной истории определенной переменной 61. Она предлагает пользователю задать 90 следующие опции:

- **Контекст**, в котором определена переменная;
- **Имя** переменной;
- **Дата начала** - для отслеживания обновления истории с определенной даты.

Это действие выводит историю изменений переменной в форме таблицы. Каждое поле формата 49 отслеживаемой переменной отображается в отдельном поле итоговой таблицы. Если значение переменной содержит несколько рядов, отображаются данные лишь первого ряда.

Тип действия: вызвать функцию 103

Имя действия: variableHistory

Иконка действия: 

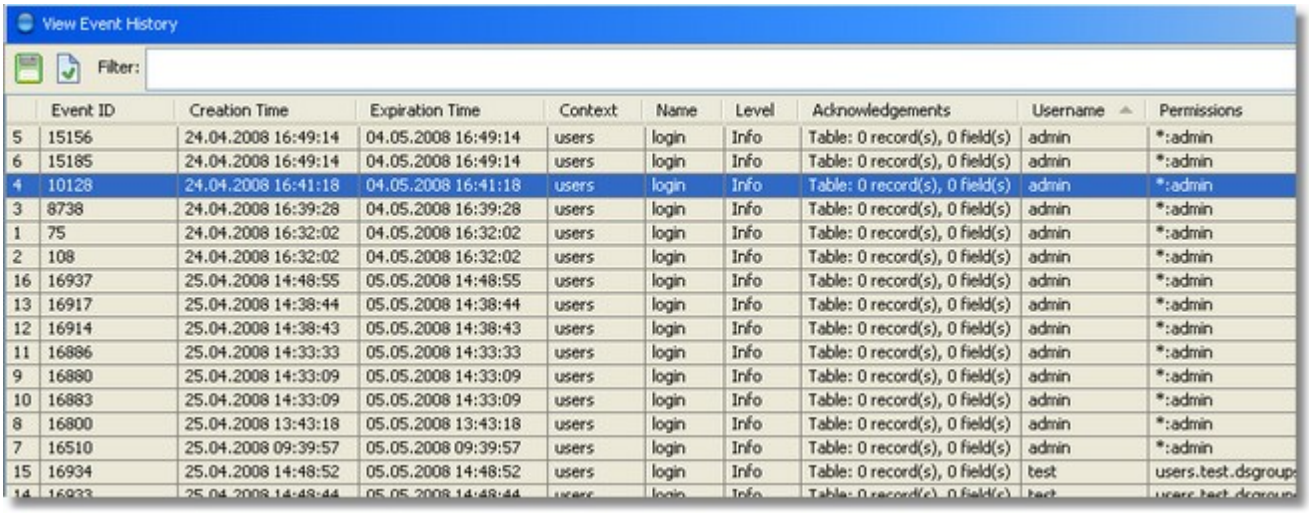
Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*.

ПОКАЗАТЬ ИСТОРИЮ СОБЫТИЯ

Это действие вызов функции 103 используется для выбора и просмотра определенных событий из истории событий 73. Оно предлагает пользователю задать 90 выбор события и критерии его отображения:

- **Маска** 44 контекстов, из которой следует выбирать события;
- **Имя события**;
- **Выражение** 112 **фильтрации** событий;
- **Дата начала** - для выбора лишь тех событий, которые происходили после определенной даты.

Ниже представлен пример вывода (выбор событий входа в систему пользователя 478, Контекстная маска = **пользователи**, Имя события = **login**, Выражение фильтра = ""):



	Event ID	Creation Time	Expiration Time	Context	Name	Level	Acknowledgements	Username	Permissions
5	15156	24.04.2008 16:49:14	04.05.2008 16:49:14	users	login	Info	Table: 0 record(s), 0 field(s)	admin	*:admin
6	15185	24.04.2008 16:49:14	04.05.2008 16:49:14	users	login	Info	Table: 0 record(s), 0 field(s)	admin	*:admin
4	10128	24.04.2008 16:41:18	04.05.2008 16:41:18	users	login	Info	Table: 0 record(s), 0 field(s)	admin	*:admin
3	8738	24.04.2008 16:39:28	04.05.2008 16:39:28	users	login	Info	Table: 0 record(s), 0 field(s)	admin	*:admin
1	75	24.04.2008 16:32:02	04.05.2008 16:32:02	users	login	Info	Table: 0 record(s), 0 field(s)	admin	*:admin
2	108	24.04.2008 16:32:02	04.05.2008 16:32:02	users	login	Info	Table: 0 record(s), 0 field(s)	admin	*:admin
16	16937	25.04.2008 14:48:55	05.05.2008 14:48:55	users	login	Info	Table: 0 record(s), 0 field(s)	admin	*:admin
13	16917	25.04.2008 14:38:44	05.05.2008 14:38:44	users	login	Info	Table: 0 record(s), 0 field(s)	admin	*:admin
12	16914	25.04.2008 14:38:43	05.05.2008 14:38:43	users	login	Info	Table: 0 record(s), 0 field(s)	admin	*:admin
11	16886	25.04.2008 14:33:33	05.05.2008 14:33:33	users	login	Info	Table: 0 record(s), 0 field(s)	admin	*:admin
9	16880	25.04.2008 14:33:09	05.05.2008 14:33:09	users	login	Info	Table: 0 record(s), 0 field(s)	admin	*:admin
10	16883	25.04.2008 14:33:09	05.05.2008 14:33:09	users	login	Info	Table: 0 record(s), 0 field(s)	admin	*:admin
8	16800	25.04.2008 13:43:18	05.05.2008 13:43:18	users	login	Info	Table: 0 record(s), 0 field(s)	admin	*:admin
7	16510	25.04.2008 09:39:57	05.05.2008 09:39:57	users	login	Info	Table: 0 record(s), 0 field(s)	admin	*:admin
15	16934	25.04.2008 14:48:52	05.05.2008 14:48:52	users	login	Info	Table: 0 record(s), 0 field(s)	test	users.test.dsgroup
14	16933	25.04.2008 14:48:44	05.05.2008 14:48:44	users	login	Info	Table: 0 record(s), 0 field(s)	test	users.test.dsgroup

Тип действия: Вызвать функцию 103

Имя действия: eventHistory

Иконка действия:

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*.

СТЕРЕТЬ ИСТОРИЮ СОБЫТИЯ

Это действие вызов функции 103 используется для удаления определенных событий из истории событий 73. Оно предлагает пользователю задать 90 критерий выбора события:

- **Маска** 44 **контекстов,** из которой следует удалить события**;**
- **Имя** события;
- **Начальная дата** - для удаления лишь тех событий, которые возникли после определенной даты (факультативно);
- **Конечная дата** - для удаления лишь тех событий, которые произошли до определенной даты (факультативно).

Тип действия: Вызвать функцию 103

Имя действия: deleteEvents

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*.

ПОКАЗАТЬ СТАТИСТИКУ КАНАЛА

Это действие вызов функции 103 используется для просмотра статистики 718 для определенного статистического канала. Он предлагает пользователю задать 90 свойства канала:

- **Маска** контекстов, где определен канал;
- **Имя** канала;
- **Ключ** пакета данных статистики;
- Период **группировки** (час, день и т.д.);
- Типы **агрегирования** (среднее, максимальное и др.)

Это действие выводит статистику канала в форме таблицы.

Тип действия: Вызвать функцию 103

Имя действия: viewStatistics

Иконка действия: 

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*.

ПРОСМОТР НЕОБРАБОТАННОЙ СТАТИСТИКИ КАНАЛА

Это действие вызов функции 103 используется для просмотра необработанной статистики 718 определенного статистического канала. Оно помогает пользователю определить 90 свойства канала:

- **Маска** контекстов, где определяется канал;
- **Имя** канала.

Это действие выводит статистику каналов в табличной форме.

Тип действия: Вызвать функцию 103

Имя действия: viewRawStatistics

Иконка действия: 

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*.

ОЧИСТИТЬ КАНАЛ СТАТИСТИКИ

Это действие вызов функции 103 используется для удаления всех данных, собранных определенным стаистическим каналом. Оно помогает пользователю определить 90 свойства канала:

- **Маску** контекстов, где определяется канал;
- **Имя** канала.

Это действие выводит статистику каналов в табличной форме.

Тип действия: Вызвать функцию 103

Имя действия: deleteStatistics

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*.

ЗАПОЛНИТЬ СТАТИСТИКУ ПО ИСТОРИИ ПЕРЕМЕННОЙ

Это действие вызов функции 103 используется для повторной инициализации статистического канала путем загрузки необработанных исторических значений переменной и их внесения в канал вместе с их историческими временными отметками.

Действие применимо для возобновления статистики, если файл статистики поврежден или отсутствует по какой-либо причине.

Вся статистика, собранная в канале, будет утеряна после выполнения этого действия.

Это действие сработает только в том случае, если включено хранение истории необработанных значений для переменной, на которой основан статистический канал. Если хранение необработанной истории отключено, это действие просто удалит статистический канал.

Если необработанная история доступна только для определенного периода, тогда канал будет содержать статистику для этого периода.

Действие помогает пользователю определить 90 свойства канала:

- **Маску** контекстов, где определяется канал;
- **Имя** канала.

Тип действия: Вызвать функцию 103

Имя дествия: fillStatisticsFromHistory

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*.

СГЕНЕРИРОВАТЬ ДАМП ПОТОКОВ

Это действие используется, чтобы сгенерировать дамп потоков. Дамп потоков - это текстовый файл со списом всех потоков Java, которые в данный момент активизированы в виртуальной машине Java.

Тип действия: Вызвать функцию 103

Имя действия: generateThreadDump

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*.

СГЕНЕРИРОВАТЬ ДАМП ПАМЯТИ

Это действие используется, чтобы сгенерировать дамп памяти. Дамп памяти - это файл в двоичном формате, который содержит детали распределения памяти внутри JVM сервера.

Тип действия: Вызвать функцию 103

Имя действия: generateHeapDump

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*.

ПОКАЗАТЬ СТАТИСТИКУ КОНТЕКСТОВ

Это действие показывает подробную статистику контекстов сервера, в том числе информацию о размере удерживаемой памяти поконтекстно.

Тип действия: Вызвать функцию 103

Имя действия: viewContextsInfo

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*.

ИМПОРТ

Используется для импортирования общих данных в систему при помощи скрипта.

Порядок действий:

1. Выбрать 94 скрипт, который будет использоваться для обработки импортированных данных.
2. Выбрать 90 импортируемый файл.
3. *[Не обязательно]* Задать 90 опции импорта, если есть опции, доступные для формата выбранного файла.
4. На этом этапе данные считываются из файла и преобразовываются в таблицу данных.
5. Просмотреть 90 импортируемые данные.
6. На этом этапе запускается скрипт импорта, импортируемая Таблица данных передается в его входных параметрах. Скрипт должен последовательно обработать записи данных таблицы и выполнить необходимые изменения в системе (например, создать/изменить ресурсы).

Имя действия: import

Иконка действия: 

Неинтерактивный режим 99**:** не поддерживается

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*.

ВЫПОЛНИТЬ ПРИЛОЖЕНИЕ

Этот вызов функции 103 выполнить 1615 используется для запуска внешнего приложения. Оно может быть использовано, например, для запуска программы в ответ на вызванную тревогу (см. Выполнение неинтерактивных действий при тревожном сообщении 793).

Тип действия: Вызвать функцию 103

Имя действия: execute

Права доступа Доступно на уровне 486 с правами доступа для *Администратора.*

Группа действия: Дополнительные действия

ЗАПУСТИТЬ ОЧИСТКУ ПАМЯТИ

Действие используется для запуска цикла полной очистки динамической памяти виртуальной машины Java с работающим AtomMind Server. Используйте это действие до проверки **Использования памяти,** параметра **% максимально допустимого использования** переменной **Статус сервера**, чтобы узнать реальное потребление памяти сервера.

Выполнение данного действия заставит JVM заморозить все активности до окончания процесса очистки (так называемый "stop the world"). Это приведет к простою AtomMind Server и всех приложений, работающих на нем. Обычно простой длится несколько секунд, но в случае большого размера кучи JVM время простоя может увеличиться до нескольких минут.

Более подробно см. в разделе Использование памяти 211.

Тип действия: Вызвать функцию 103

Имя действия: runGarbageCollection

Права доступа Доступно на уровне 486 с правами доступа для *Администратора.*

Группа действия: Дополнительные действия

ПОКАЗАТЬ СТАТИСТИКУ БАЗЫ ДАННЫХ

Это действие отображает ряд событий и свойств во всех таблицах базы данных 692 AtomMind Server . Ряд событий/свойств сгруппированы согласно таблицам БД, пути контекста и именам события/свойства.

Тип действия: Редактировать свойства 91

Имя действия: viewDatabaseStatistics

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*

Группа действия: Дополнительные действия

ОТКРЫТЬ В БРАУЗЕРЕ

Это действие позволяет пользователю открыть определенный URL-адрес в клиентском браузере по умолчанию.

Имя действия: browse

Неинтерактивный режим 99: не поддерживается

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*

Параметры 102 выполнения:

- **URL**. Может быть определено как поле Строка `url` таблицы ввода действия.

СРАВНИТЬ

Это действие позволяет пользователю сравнивать две строки в графическом модуле поиска различий (Diff Viewer).

Имя действия: compare

Неинтерактивный режим 99: не поддерживается

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*

Параметры 102 выполнения

- **Левый заголовок**. Заголовок левого поля diff viewer.
- **Левое значение.** Значение, которое отображается в левом поле diff viewer.

- **Правый заголовок.** Заголовок правого поля diff viewer.
- **Правое значение.** Значение, которое отображается в правом поле diff viewer.

ПРЕКРАТИТЬ КЛИЕНТСКОЕ СОЕДИНЕНИЕ

Это действие используется для принудительной деаутентификации и отключения одной или более клиентских сессий, независимо от их типа (дектоп, web, API, с т.д.)

Тип действия:	Вызвать функцию 103
Имя действия:	terminateClientConnection
Права доступа:	Доступно на уровне 486 с правами доступа для *Администратора*
Группа действия:	Дополнительные действия

Общие действия [? 1450]

Редактировать права доступа к контексту 106, Просмотр событий 109

Состояния и иконки контекста

У данного контекста нет состояний 44. Он обычно представлен иконкой  . В AtomMind Client 359 иконка, представляющая контекст, зависит от состояния подключения. См. дополнительную информацию здесь 376.

Дополнительная информация

Информация о контексте

Тип контекста 43: корневой

Имя контекста 42: "" (пустая строка)

Описание контекста 43: "" (пустая строка)

Путь контекста 42: "" (пустая строка)

Контекстная маска 44: "" (пустая строка)

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Вход в систему пользователей. Саморегистрация пользователей.
Наблюдатель	Выполнение запросов AtomMind в текстовом виде.
Оператор	Изменение пароля пользователя.
Менеджер	Создание ресурсов 208.
Инженер	Запросы на проверку и обработку входящей почты.
Администратор	Просмотр информации о сервере. Остановка и перезапуск сервера. Выполнение запросов в текстовом формате. Выполнение внешних приложений. Доступ к истории и статистике переменных/событий. Просмотр статистики базы данных. Отправка E-mail и SMS сообщений.

Общие переменные (свойства) [? 61]

ВЕРСИЯ СЕРВЕРА

Возвращает версию AtomMind Server.

Имя переменной: version

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
version	Строка	Строка версии сервера.

СТАТУС СЕРВЕРА

Возвращает информацию о времени работы AtomMind Server.

Имя переменной: status

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
name	String	Описание экземпляра сервера.
version	String	Версия сервера.
startTime	Date	Временная метка запуска сервера.
startupDuration	Long	Время, требуещееся для запуска сервера.
uptime	Long	Оперативное время сервера.
freeMemory	Long	Свободная память в выделенной куче JVM.
maxMemory	Long	Максимальный разрешенный размер кучи JVM. Контролируется -Xmx параметром 158.
totalMemory	Long	Текущий размер кучи JVM.
memoryUsage	Double	Текущий процент использования максимального объема кучи JVM. Показывает реальное использование памяти сервера при проверке сразу после выполнения действия Запустить очистку памяти 1559.
cpuLoad	Double	Загрузка процессора виртуальной машиной Java.
cpuLoadSystem	Double	Общая загрузка процессора CPU load.
eventQueueLength	Integer	Длина очереди событий контекста сервера.

eventsScheduled	Long	Общее число запланированных событий.
eventsProcessed	Long	Общее число обрабатываемых событий.
diskUtilization	Data Table	Информация об использовании диска.

СТАТИСТИКА БАЗЫ ДАННЫХ

Возвращает обзор статистики базы данных AtomMind Server.

Имя переменной: database

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
queries	Long	Количество выполненных запросов с момент запуска сервера.
transactions	Long	Количество выполненных транзакций с момент запуска сервера.
loaded	Long	Количество вызванных из базы данных объектов с момент запуска сервера.
updated	Long	Количество обновленных в базе данных объектов с момент запуска сервера.
inserted	Long	Количество внесенных в базу данных объектов с момента запуска сервера.
deleted	Long	Количество удаленных из базы данных объектов с момента запуска сервера.
maxQueryTime	Long	Максимальное время выполнения всех выполненных запросов базы данных с момента запуска сервера.

СТАТИСТИКА ТАБЛИЦ БАЗЫ ДАННЫХ

Возвращает подробную статистику таблиц базы данных AtomMind Server.

Имя переменной: tables

Записи: 0... не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
table	String	Имя таблицы.

loaded	Long	Количество вызванных из таблицы объектов с момента запуска сервера.
updated	Long	Количество обновленных в таблице объектов с момента запуска сервера.
inserted	Long	Количество внесенных в таблицу объектов с момента запуска сервера.
deleted	Long	Количество удаленных из таблицы объектов с момента запуска сервера.

СТАТИСТИКА КЛАСТЕРА БАЗЫ ДАННЫХ

Возвращает подробную статистику кластера базы данных AtomMind Server.

Имя переменной: databaseCluster

Записи: 0... не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*

Формат 49 записи:

Имя поля	Тип поля	Примечания
node	String	Имя узла кластера базы данных.
location	String	Расположение (URL) узла кластера базы данных.
alive	Boolean	Показывает, доступна ли база данных для подключения.
active	Boolean	Показывает, используется ли база данных кластером в данный момент.
synchronizationDuration	Long	Продолжительность синхронизации текущего кластера базы данных или NULL, если синхронизация в даный момент не происходит.

СТАТИСТИКА КОНТЕКСТОВ

Возвращает ститистическую информацию о контекстах, функциях, событиях и действиях AtomMind Server.

Имя переменной: sysinfo

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
context	Integer	Имя и описание контекста AtomMind Server.
variableCount	Integer	Итоговое количество определений переменной в контексте.
functionCount	Integer	Итоговое количество определений функции в контексте.

eventCount	Integer	Итоговое количество определений события в контексте.
actionCount	Integer	Итоговое количество определений события в контексте.
variablesRead	Long	Количество выполненных операций чтения переменных с момента запуска сервера.
variablesWritten	Long	Количество выполненных операций записи переменных с момента запуска сервера.
functionsCalled	Long	Количество выполненных операций вызова функции с момента запуска сервера.
eventsFired	Long	Количество появившихся событий с момента запуска сервера.
eventHandleOffers	Long	Количество предложений обработчикам событий на обработку данного события.
eventHandleExecutions	Long	Количество обработок данного события обработчиком событий.
eventListenerCount	Long	Количество слушателей данного события.
eventQueuesLength	Long	Очередь внутренних событий контекста для данного события.
memory	Long	Примерный объем динамической памяти, удерживаемый данным контекстом.

СТАТИСТИКА УСТРОЙСТВ

Возвращает краткую статистическую информацию об устройствах AtomMind Server.

Имя переменной: deviceStatistics

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя.*

Формат 50 записи:

Имя поля	Тип поля	Примечания
variablesRead	Long	Количество операций чтения переменной, выполненных с момента запуска сервера.
variablesWritten	Long	Количество операций записи переменной, выполненных с момента запуска сервера.
functionsCalled	Long	Количество операция вызова функций, выполненных с момента запуска сервера.
eventsFired	Long	Количество событий, произошедших с момента запуска сервера.

ЛИЦЕНЗИОННАЯ ИНФОРМАЦИЯ

Возвращает информацию об активной лицензии AtomMind Server.

Имя переменной: license

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
issueDate	Data	Дата выдачи лицензии.
holder	String	Владелец лицензии.
version	String	Шаблон версии сервера.
trialPeriod	Integer	Период тестирования (количество дней) или NULL для лицензии без периода тестирования.
trialRemaining	Integer	Оставшийся период тестирования (количество дней) или NULL для лицензии без периода тестирования.
maxDevices	Integer	Максимальное количество устройств, разрешенных лицензией.
curDevices	Integer	Текущее количество зарегистрированных устройств.
activationKey	String	Ключ активации сервера.
pluginGroups	Data Table	Группы плагинов, включенные в лицензию: • Описание группы плагинов • Плагины, входящие в группу • Параметры группы, т.е. различные ограничения по лицензии для плагинов группы

АКТИВНЫЕ РАСШИРЕНИЯ

Возвращает информацию об активных плагинах 207 AtomMind Server .

Имя переменной: plugins

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
id	String	ID плагина.
type	String	Тип плагина.

name	String	Описание плагина.

АКТИВНЫЕ КЛИЕНТСКИЕ ПОДКЛЮЧЕНИЯ

Возвращает информацию об активных подключениях клиента.

Имя переменной: connections

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа *Наблюдатель*

Формат 49 записи:

Имя поля	Тип поля	Примечания
user	String	Имя авторизовавшегося пользователя 478.
login	String	Вход в систему авторизованного пользователя. Может отличаться от имени пользователя при использовании внешней аутентификации.
type	String	Тип подключения (Клиент, Web, Web-сервис и пр.)
date	Date	Время установки соединения.
address	String	IP-адрес клиента.
eventsQueued	Long	Текущее количество необработанных событий в очереди в текущей сессии клиента.
eventsDiscarded	Long	Количество исключенных событий во время текущей сессии клиента из-за переполнения очереди.
contextLocks	Data Table	Список активных блокировок контекстов.

ТЕКУЩЕЕ ИМЯ ПОЛЬЗОВАТЕЛЯ СЕССИИ

Возвращает текущее имя пользователя сессии.

Имя переменной: username

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*

Формат 49 записи:

Имя поля	Тип поля	Примечания
value	String	Имя авторизованного пользователя 478.

ТЕКУЩИЙ ВХОД В СИСТЕМУ

Возвращает текущий вход в систему.

Имя переменной: login

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*

Формат 49 записи:

Имя поля	Тип поля	Примечания
value	Строка	Вход авторизованного пользователя. Может отличаться от имени пользователя при использовании внешней аутентификации.

ТЕКУЩИЙ ТИП СОЕДИНЕНИЯ СЕССИИ

Возвращает текущий тип соединения сессии.

Имя переменной: type

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*

Формат 49 записи:

Имя поля	Тип поля	Примечания
value	String	Тип соединения (Клиент, Веб-клиент, Веб-сервис и т.д.)

СТАТУС КЛАСТЕРА

Возвращает информацию о статусе узла отказоустойчивого кластера 1326.

Имя переменной: cluster

Записи: 0... не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*

Формат 49 записи:

Имя поля	Тип поля	Примечания
id	String	Идентификатор узла кластера.
role	Integer	Роль узла кластера.
time	Long	Истекшее время с момента подтверждения текущего статуса узла.

ПОТОКИ

Возвращает полный список потоков сервера.

Имя переменной: threads

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне[486] с правами доступа *Администратор*

Формат[49] записи:

Имя поля	Тип поля	Примечания
id	Long	Уникальный ID потока.
name	String	Имя потока.
group	String	Имя группы потока.
priority	Integer	Приоритет потока.
state	String	Состояния потока.
daemon	Boolean	True, если поток является потоком демона.
interrupted	Boolean	True, если поток был прерван.
cpu	Double	Средняя нагрузка на ЦП потока в последнюю секунду работы сервера. Измеряется в процентах от общей нагрузки на ЦП сервера. Заблокированные потоки имеют нулевую нагрузкеу на ЦП, новые потоки отображаются с нагрузкой NULL.
stack	Data Table	Трек стека потока со следующими полями: • имя класса • имя метода • имя файла • номер строки

СТАТИСТИКА ПУЛА ПОТОКОВ

Возвращает статистическую информацию о пулах потоков сервера. Позволяет диагнозировать здоровье и производительность различных пулов потоков сервера, включая, но не ограничиваясь:

- **Исполнителями синхронизации**, то есть потоками, выполняющими задания синхронизации устройства[514]
- **Таймерами синхронизации**, инициирующими новые синхронизации устройств согласно расписанию
- **Исполнителями операций контекста**, которые позволяют производить одновременно включение/выключение сервера, а также параллельно выполнять задания, требуемые контекстами[41]
- **Процессорами команд клиентов**, чьи потоки отвечают за обработку запросов, полученных от экземпляров AtomMind Client посредством API сервера[1340]
- **Отправителями команд**, которые посылают команды по протоколу AtomMind[2119] в сокеты сети
- **Асинхронными обработчиками обновлений**, которые обрабатывают события и обновления значений, полученных с помощью протокола AtomMind
- **Пользовательскими пулами потоков**, которые используются такими модулями, как Тревоги[779], Трекеры[2181] и т.д.

Имя переменной: pools

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа *Администратор*

Формат 49 записи:

Имя поля	Тип поля	Примечания
poolName	String	Имя пула.
activeCount	Integer	Количество активных задач.
completedCount	Long	Количество выполненных задач.
totalCount	Long	Итоговое количество задач.
coreSize	Integer	Размер ядра пула.
largestSize	Integer	Самый большой размер пула.
maximumSize	Integer	Максимально разрешенный размер пула.
queueLength	Integer	Длина очереди задач.

СТАТИСТИКА ПОТОКОВ

Возвращает общую статистическую информацию о потоках сервера.

Имя переменной: threadStatistics

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа *Администратор*

Формат 49 записи:

Имя поля	Тип поля	Примечания
live	Integer	Количество активных на текущий момент потоков.
maximumLive	Integer	Максимальное количество одновременно запущенных текущих потоков с момента запуска сервера.
totalStarted	Integer	Общее количество потоков, запущенных с момента запуска сервера.

СРЕДА СЕРВЕРА

Возвращает информацию о JVM, запускающей AtomMind Server, и о сервере/компьютере, который запущен.

Имя переменной: environment

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*

Формат 49 записи:

Имя поля	Тип поля	Примечания
property	String	Имя свойства.
value	String	Значение свойства.

СТАТИСТИКА ПРАВИЛ ОБРАБОТКИ СОБЫТИЙ

Возвращает статистическую информацию, собранную правилами обработки событий 196. Следует отметить, что статистика правил событий перенастраивается при перезапуске сервера.

Имя переменной: eventRuleStatistics

Записи: 0... не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Администратора*

Формат 49 записи:

Имя поля	Тип поля	Примечания
context	String	Путь контекста.
event	String	Имя события.
filtered	Long	Количество событий, блокированных этим правилом.
stored	Long	Количество событий, прошедших правила и сохранившихся в базе даных сервера.

ДАННЫЕ ОЧЕРЕДИ СОБЫТИЙ

Возвращает статистическую информацию об очереди событий. Если в очереди нет событий, у данной переменной не будет записей.

Имя переменной: eventQueueStatistics

Записи: 0..не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Администратора*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
context	String	Путь контекста.
eventCount	Long	Количество необработанных событий в очереди для данного контекста.

СТАТИСТИКА СОБЫТИЙ

Возвращает количество событий[73], сгруппированных согласно таблице баз данных[692] AtomMind Server, пути контекста и имени события;

Имя переменной: eventStatistics

Записи: 0..не ограничено

Права доступа: Доступно для чтения на уровне[486] с правами доступа для *Администратора*.

Формат[49] записи:

Имя поля	Тип поля	Примечания
table	String	Имя таблицы БД.
context	String	Путь контекста.
event	String	Имя события.
count	Long	Количество событий выше упомянутого типа, которые возникают в выше упомянутом контексте. Обратите внимание, что все события определенного типа, которые возникают в одном контексте, **всегда** хранятся в одной таблице.

СТАТИСТИКА ПЕРЕМЕННЫХ

Возвращает информацию о том, сколько значений переменных[61] хранятся в БД для каждого контекста AtomMind Server.

Имя переменной: variableStatistics

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне[486] с правами доступа для *Администратора*.

Формат[49] записи:

Имя поля	Тип поля	Примечания
context	String	Путь контекста.
count	Long	Количество значений переменной контекста, который в настоящий момент хранятся в БД. Обратите внимание, что значения переменной, выбранные по умолчанию, не сохраняются в БД, и, т.о., количество может быть очень низким сразу после инсталляции сервера.

Общие функции [?[70]]

ЗАРЕГИСТРИРОВАТЬ НОВОЮ ПОЛЬЗОВАТЕЛЬСКУЮ УЧЕТНУЮ ЗАПИСЬ

Регистрирует новую учетную запись пользователя[478].

Ия функции: register

Права доступа: Доступно на уровне 486 с правами доступа *Нет прав*

Записи ввода: 1

Формат 49 ввода:

Имя	Тип	Описание
username	String	Имя учетной записи.
password	String	Пароль учетной записи.
passwordre	String	Пароль учетной записи.
Права доступа	String	Уровень 486 с правами доступа для нового пользователя. Используется только в том случае, если регистрация была выполнена администратором.

Записи вывода: 0

Формат 49 вывода: нет

ВХОД

Передает информацию об аутентификации/авторизации и продолжает сессию с правами доступа для определенного пользователя 478.

Имя функции: login

Права доступа: Доступно на уровне 486 с правами доступа *Нет прав*

Записи ввода: 1

Формат 49 ввода:

Имя	Тип	Описание
username	String	Имя пользователя для авторизации.
password	String	Пароль учетной записи пользователя.
code	String	Код авторизации (факультативно).
state	String	Статус авторизации (факультативно).
provider	String	Провайдер авторизации (факультативно).
countAttempts	Boolean	Определяет, включен ли подсчет попыток входа (факультативно).

Записи вывода: 0

Формат 49 вывода: нет

ВЫХОД

Выход из системы с продолжением сессии с правами доступа *Нет прав*.

Имя функции: logout

Права доступа: Доступно на уровне 486 с правами доступа *Нет прав*.

Записи ввода: 0

Формат 49 ввода: нет

Записи вывода: 0

Формат вывода: нет

При вызове данной функции в виджете внутри веб окружения, вы увидите изменения только после обновления страницы. Вместо этого вы можете использовать операцию Выход из системы 1044 корневой панели виджета.

ИЗМЕНИТЬ ПАРОЛЬ

Изменяет пароль для авторизованного в настоящий момент пользователя учетной записи.

Имя функции: changePassword

Права доступа: Доступно на уровне 486 с правами доступа для *Оператора*.

Записи ввода: 1

Формат ввода:

Имя	Тип	Описание
oldPassword	String	Старый пароль для учетной записи.
newPassword	String	Новый пароль.
repeatPassword	String	Должно соответствовать новому паролю.

Записи вывода: 0

Формат вывода: нет

ВЫПОЛНИТЬ ЗАПРОС

Выполняет пользовательский запрос 829 с правами доступа текущего пользователя.

Имя функции: executeQuery

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*.

Записи ввода: 1

Формат ввода:

Имя	Тип	Описание
query	String	Текст запроса.

Записи вывода: 0...не ограничено

Формат вывода: динамический

ВЫПОЛНИТЬ ПРЯМОЙ ЗАПРОС К СУБД

Выполняет прямой запрос в БД AtomMind Server. Дополнительную информацию см. здесь 1548.

Имя функции: executeNativeQuery

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*.

Записи ввода: 1

Формат ввода:

Имя	Тип	Описание
query	String	Текст запроса.
update	Boolean	Флажок, указывающий, что это запрос на обновление (DELETE, INSERT, UPDATE и пр.).

Записи вывода: 0...не ограничено

Формат 49 вывода: динамический

ПЕРЕЗАПУСТИТЬ СЕРВЕР

Перезапускает AtomMind Server.

Имя функции: restart

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*

Записи ввода: 1

Формат 49 ввода:

Имя	Тип	Описание
instantly	Boolean	Позволяет выбирать между немедленным и отложенным перезапуском.
delay	Long	Задержка перед запуском по расписанию.
reason	String	Причина для перезапуска по расписанию.

Записи вывода: 0

Формат 49 вывода: нет

ОСТАНОВИТЬ СЕРВЕР

Останавливает AtomMind Server.

Имя функции: stop

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*

Записи ввода: 1

Формат 49 ввода:

Имя	Тип	Описание
instantly	Boolean	Позволяет выбирать между немедленным и отложенным выключением.
delay	Long	Задержка перед выключением по расписанию.
reason	String	Причина для выключения по расписанию.

Записи вывода: 0

Формат 49 вывода: нет

НАЧАТЬ РЕЖИМ ОБСЛУЖИВАНИЯ

Заставляет сервер войти в режим обслуживания 172.

Имя функции: startMaintenanceMode

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*

Записи ввода: 0

Формат 49 ввода: нет

Записи вывода: 0

Формат 49 вывода: нет

ЗАКОНЧИТЬ РЕЖИМ ОБСЛУЖИВАНИЯ

Заставляет сервер выйти из режима обслуживания[172].

Имя функции: stopMaintenanceMode

Права доступа: Доступно на уровне[486] с правами доступа для *Администратора*

Записи ввода: 0

Формат[49] ввода: нет

Записи вывода: 0

Формат[49] вывода: нет

УСТАНОВИТЬ СЕССИЮ

Устанавливает значение переменной сессии. Переменная доступна до тех пор, пока текущая сессия пользователя открыта.

Имя функции: sessionSet

Права доступа: Доступно на уровне[486] с правами доступа для *Оператора*

Записи ввода: 1

Формат[49] ввода:

Имя	Тип	Описание
sessionVariable	String	Имя переменной сессии.
sessionValue	Data Table	Значение переменной.

Записи вывода: 0...без ограничений

Формат[49] вывода: нет

ПОЛУЧИТЬ СЕССИЮ

Получает значение упомянутой переменной сессии.

Имя функции: sessionGet

Права доступа: Доступно на уровне[486] с правами доступа для *Оператора*

Записи ввода: 1

Формат[49] ввода:

Имя	Тип	Описание
sessionVariable	String	Имя переменной сессии.

Записи вывода: 0...без ограничений

Формат[49] вывода: Динамический

Общие события [?[73]]

Общие события: info (информация)[77]

ДОБАВЛЕНИЕ КОНТЕКСТА

Возникает при добавлении контекста любого типа к контекстному дереву сервера. Это событие может возникать множество раз для одного и того же контекста, т.е. при каждом новом запуске сервера.

Имя события: contextAdded

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*

Период действия: непостоянный

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
context	String	Путь контекста.

УДАЛЕНИЕ КОНТЕКСТА

Возникает при удалении контекста любого типа из контекстного дерева сервера. Это событие может возникать множество раз для одного и того же контекста, т.е. при каждом новом выключении сервера.

Имя события: contextRemoved

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*

Период действия: Непостоянный

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
context	String	Путь контекста.

СОЗДАНИЕ КОНТЕКСТА

Возникает при создании нового контекста ресурса (например, тревоги 779, виджета 943 или устройства 497). В отличие от события **contextAdded**, это событие возникает один раз для каждого ресурса при его первоначальном создании. Оно не появится снова при запуске сервера.

Имя события: contextCreated

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*

Период действия: Непостоянный

Записи: 1

Формат 50 записи:

Field Name	Тип поля	Примечания
context	String	Путь контекста.

УНИЧТОЖЕНИЕ КОНТЕКСТА

Возникает, когда контекст ресурса (например, тревога 779, виджет 943 или устройство 497) навсегда уничтожается. В отличие от события **contextRemoved**, это событие возникает один раз при удалении ресурса. Оно не возникает при выключении сервера.

Имя события: contextDestroyed

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*

Период действия: Непостоянный

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
context	String	Путь контекста.

ДОБАВЛЕНИЕ СУЩНОСТИ КОНТЕКСТА

Инициируется, когда новое определение сущности (переменной, функции, события, действия) добавляется в любой контекст сервера.

Имя события: contextEntityAdded

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*

Период действия: Непостоянный

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
context	String	Путь контекста, где была добавлена сущность.
entity	String	Имя сущности.
entityType	Integer	Тип сущности: • 1 - Переменная 61 • 2 - Функция 70 • 3 - Событие 73 • 4 - Действие 87

УДАЛЕНИЕ СУЩНОСТИ КОНТЕКСТА

Инициируется, когда новое определение сущности (переменной, функции, события, действия) удаляется из любого контекста сервера.

Имя события: contextEntityRemoved

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*

Период действия: Непостоянный

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
context	String	Путь контекста, где была удалена сущность.
entity	String	Имя сущности.
entityType	Integer	Тип сущности: • 1 - Переменная 61 • 2 - Функция 70

		• 3 - Событие 73 • 4 - Действие 87

ОБРАТНАЯ СВЯЗЬ

Сервер генерирует это событие для отчета об активности авторизованного в настоящий момент пользователя 478.

Имя события feedback

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*.

Период действия: Непостоянный

Записи: 1

Формат 49 записи:

Имя поля	Тип поля	Примечания
message	String	Сообщение сервера.

ЗАГРУЗКА СЕРВЕРА

Сервер создает это событие, чтобы создать отчет о завершении загрузки.

Имя события: serverStarted

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*.

Период действия: Непостоянный

Записи: 1

Формат 49 записи:

Имя поля	Тип поля	Примечания
startupDuration	Long	Длительность загрузки в милисекундах.

ОБНОВЛЕНИЕ ПЕРЕМЕННОЙ СЕССИИ

Инициируется, когда обновляется переменная сессии.

Имя события: sessionVariableUpdated

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*.

Период действия: Непостоянный

Записи: 1

Формат 49 записи:

Имя поля	Тип поля	Примечания
sessionVariable	String	Имя обновленной переменной сессии.

sessionOldValue	Data Table	Старое значение переменной сессии.
sessionNewValue	Data Table	Новое значение переменной сессии.

17.36 Планировщик

Этот контекст 41 является контейнером, который содержит все запланированные задачи 823 для отдельного пользователя.

Уникальные действия [? 1450]

СОЗДАТЬ (действие по умолчанию 88)

Это действие используется для планирования новых задач. Оно позволяет задать основные свойства 827 для новой задачи.

Тип действия: создать 105

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*.

ЗАПЛАНИРОВАТЬ НОВОЕ ЗАДАНИЕ

Это действие также добавляет новую запланированную задачу. Оно реализуется как действие перетаскивания мышью 101, которое принимает контексты любого типа. Система подсказывает пользователю выбрать действие для планирования из контекста, которое было перенесено мышью в контекст Запланированные задачи.

Тип действия: перетаскивание мышью 101

Имя действия: createByDnD

Не интерактивный режим 99: не поддерживается

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*

Общие действия [? 1450]

Создать на основе шаблона 105, копировать в дочерние контексты 111, импорт 108, экспорт 108, редактировать права доступа 106, просмотр событий 109, поиск/фильтрация 110, различные групповые действия 101 согласно контекстам потомков.

Состояния и иконки контекста

У данного контекста нет состояний 44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста 43: задачи

Имя контекста 42: задачи

Описание контекста 43: запланированные задачи

Путь контекста 42: users.USER_NAME.jobs

Контекстная маска 44: users.*.jobs

Права доступа к контексту [?44]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт задач.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [?61]

У этого контекста нет общих переменных (свойств).

Общие функции [?70]

Общие функции: makeCopy (сделать копию)71, delete (удалить)72

СОЗДАТЬ

Создает новую запланированную задачу.

Имя функции: create

Права доступа: Доступно на уровне486 с правами доступа для *Менеджера*

Записи ввода: 1

Формат49 ввода: Аналогичный формату переменной jobDetailsView1587 в контексте запланированная задача1586.

Записи вывода: 0

Формат49 вывода: нет

Общие события [?73]

Общие события: info (информация)77

17.37 Запланированная задача

Этот контекст41 предоставляет Вам доступ к одной запланированной задаче823 и позволяет ею управлять.

Уникальные действия [?1450]

НАСТРОИТЬ

Это действие используется для редактирования свойств827 запланированной задачи.

Изменение поля **Имя** во время этой операции меняет название данного контекста. Это может привести к сбою в работе других компонентов системы, использующих имя/путь контекста в качестве основного идентификатора.

Тип действия: настроить105

ВЫПОЛНИТЬ ЗАДАНИЕ

Заставляет AtomMind Server выполнить одну запланированную задачу в не интерактивном режиме.

Тип действия: вызвать функцию 103

Имя действия: execute

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*.

Общие действия [? 1450]

Удалить 106, создать копию 109, реплицировать 110, редактировать права доступа 106, просмотр событий 109, показать статус 111

Состояния и иконки контекста

Иконка	Код	Состояние
	0	Задача активирована
	1	Задача деактивирована

Дополнительная информация

Информация о контексте

Тип контекста 43: задача

Имя контекста 42: предоставляется пользователем

Описание контекста 43: предоставляется пользователем

Путь контекста 42: users.USER_NAME.jobs.SCHEDULED_JOB_NAME

Контекстная маска 44: users.*.jobs.*

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Конфигурация и удаление задач. Принудительное выполнение задач.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие свойства [? 61]

Общие переменные: groupMembership (членство группы) 67, activeAlerts (активные тревоги)

СВОЙСТВА ЗАПЛАНИРОВАННОЙ ЗАДАЧИ

См. описание переменной и ее полей здесь 827.

Имя переменной: jobDetailsView

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
name	Строка	1 - 50 знаков
description	Строка	1 - 50 знаков
enabled	Булевое	
mask	Строка	
action	Строка	
input	Таблица данных	

ПРОСТОЕ РАСПИСАНИЕ

См. описание переменной и ее полей здесь 828.

Имя переменной: triggersView

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
startTime	Дата	
endTime	Дата	
repeatCount	Целое	
repeatInterval	Длинное	

ДОПОЛНИТЕЛЬНОЕ РАСПИСАНИЕ

См. описание переменной и ее полей здесь 828.

Имя переменной: cronTriggersView

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера.*

Формат 49 записи:

Имя поля	Тип поля	Примечания

dayOfMonth	Таблица данных	
month	Таблица данных	
dayOfWeek	Таблица данных	
year	Строка	
time	Дата	Используется только редактор времени.
startTime	Дата	
endTime	Дата	

СТАТУС ЗАДАНИЯ

Переменная предоставляет данные о статусе запланированной задачи. Ее отображает действие Просмотреть статус 111.

Имя переменной: status

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*

Формат 49 записи:

Имя поля	Тип поля	Примечания
previousFireTime	Дата	Время, прошедшее с момента последнего выполнения задачи, или NULL, если оно еще не было выполнено.
nextFireTime	Дата	Время до выполнения следующей задачи или NULL, если задача больше не будет выполняться.

Общие функции [? 70]

У этого контекста нет общих функций.

ВЫПОЛНИТЬ ЗАДАНИЕ

Заставляет AtomMind Server немедленно выполнять одну запланированную задачу в неинтерактивном режиме. Функция синхронная, т.е. она не закончится до тех пор, пока не закончится выполнение задачи.

Имя функции: execute

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*.

Записи ввода: 0

Формат 49 ввода: нет.

Записи вывода: 0

Формат 49 вывода: нет.

Общие события [? 73]

Общие события: info (информация) 77, contextStatusChanged (статус изменен) 83

17.38 Скрипты

Этот контекст[41] является контейнером, который содержит все скрипты[879] для отдельного пользователя.

Уникальные действия [?1450]

СОЗДАТЬ (действие по умолчанию[88])

Это действие используется для создания новых скриптов. Оно позволяет пользователю задать основные параметры[880] для нового скрипта.

Тип действия:	создать[105]
Права доступа:	Доступно на уровне[486] с правами доступа для *Менеджера*.

Общие действия [?1450]

Создать на основе шаблона[105], копировать в дочерние контексты[111], импорт[108], экспорт[108], редактировать права доступа[106], просмотр событий[109], поиск/фильтрация[110], различные групповые действия[101] согласно контекстам потомков.

Состояния и иконки контекста

У этого контекста нет состояний[44]. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста[43]: скрипты

Имя контекста[42]: скрипты

Описание контекста[43]: скрипты

Путь контекста[42]: users.USER_NAME.scripts

Контекстная маска[44]: users.*.scripts

Права доступа к контексту [?44]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт скрипта.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [?61]

У данного контекста нет общих переменных (свойств).

Общие функции [?70]

Общие функции: makeCopy (скопировать)[71], delete (удалить)[72]

СОЗДАТЬ

Создает новый скрипт.

Имя функции:	create
Права доступа:	Доступно на уровне 486 с правами доступа для *Менеджера*
Записи ввода:	1
Формат 49 ввода:	Аналогичный формату переменной childInfo 1592 в контексте Скрипты 1591, за исключением ситуации, когда **текстовое** поле скрыто.
Записи вывода:	0
Формат 49 вывода:	нет

Общие события [? 73]

Общие события: info (информация) 77

17.39 Скрипт

Этот контекст 41 предоставляет Вам доступ к одному скрипту 879 и позволяет им управлять.

Уникальные действия [? 1450]

ВЫПОЛНИТЬ СКРИПТ (действие по умолчанию 88)

Это действие заставляет сервер скомпилировать и запустить скрипт. Если во время компиляции возникают ошибки, или же они возникают по вине самого скрипта, пользователь оповещается через GUI процедуру показать ошибку 95.

Имя действия:	execute
Не интерактивный режим 99:	не поддерживается
Права доступа:	Доступно на уровне 486 с правами доступа для *Инженера*.

НАСТРОИТЬ

Это действие используется для редактирования свойств 880 Скрипта. Следует отметить, что для изменения текста скрипта необходим уровень 486 с правами доступа *Администратора*

Изменение поля **Имя** во время этой операции меняет название данного контекста. Это может привести к сбою в работе других компонентов системы, использующих имя/путь контекста в качестве основного идентификатора.

Тип действия:	настроить 105

РЕДАКТИРОВАТЬ ИСТОЧНИК

Это действие запускает встроенный в текст редактор (с подсветкой синтаксиса) и позволяет пользователю редактировать источник скрипта. Текстовый редактор, доступный в AtomMind Client 359, описан здесь 408.

Имя действия:	editSource
Не интерактивный Mode 99:	не поддерживается
Права доступа:	Доступно на уровне 486 с правами доступа для *Администратора*.

Общие действия [? 1450]

Удалить 106, создать копию 109, реплицировать 110, редактировать права доступа 106, просмотр событий 109, показать статус 111

Состояния и иконки контекста

У этого контекста нет состояний 44. Он представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста 43: скрипт

Имя контекста 42: предоставляется пользователем

Описание контекста 43: предоставляется пользователем

Путь контекста 42: users.USER_NAME.scripts.SCRIPT_NAME

Контекстная маска 44: users.*.scripts.*

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Конфигурация и удаление скрипта.
Инженер	Выполнение скрипта.
Администратор	Редактирование источника скрипта.

Общие переменные (свойства) [? 61]

Общие переменные: groupMembership (членство группы) 67, activeAlerts (активные тревоги)

СВОЙСТВА СКРИПТА

См описание переменной и ее полей здесь 880.

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
name	Строка	1 - 50 знаков
description	Строка	1 - 50 знаков
text	Строка	
autorun	Булевое	

Общие функции [?70]

ВЫПОЛНИТЬ СКРИПТ

Выполняет скрипт и возвращает сгенерированные им данные. Список входных параметров, передаваемых в скрипт, строится на основе Таблицы входных данных путем последовательного извлечения значений из полей этой таблицы, начиная с первого ряда. Если скрипт возвращает объект Таблицы данных, то он возвратит его неизмененным. Если скрипт возвращает другой тип значения, поддерживаемый форматом[49] таблицы, этот объект помещается в таблицу с одной ячейкой с полем соответствующего формата. Если скрипт возвращает значение такого типа, который не поддерживается форматом Таблицы данных, это значение преобразовывается в строку (при помощи Java-метода `Object.toString()`) и помещается в Таблицу данных с одной ячейкой и полем строки.

Имя функции:	execute
Права доступа:	Доступно на уровне[486] с правами доступа для *Инженера*.
Записи ввода:	0...не ограничено
Формат[49] ввода:	динамический
Записи вывода:	0...не ограничено
Формат[49] вывода:	динамический

Общие события [?73]

Общие события: info (информация)[77]

17.40 Датчики

Этот контекст[41] является контейнером, который содержит все датчики[2181] для определенного пользователя.

Уникальные действия [?1450]

- Сконструировать датчик[1595]

Другие уникальные действия:

СОЗДАТЬ ДАТЧИК (действие по умолчанию[88])

Это действие используется для создания новых датчиков. Оно позволяет пользователю задать основные свойства[2183] для нового датчика и настроить его сразу же после создания.

Тип действия:	создать[105]
Права доступа:	Доступно на уровне[486] с правами доступа для *Менеджера*.

Действия, связанные с переменной [?102]

СОЗДАТЬ ДАТЧИК

Это действие создает новый датчик, который отслеживает значение переменной.

Если у переменной больше одного поля, система подсказывает пользователю выбрать поле, которое датчик будет отображать.

Если переменная принадлежит контексту устройства[1494] (например, это параметр устройства), система подсказывает[89] пользователю создать статус[2182] в режиме **Offline** для только что созданного датчика. Датчик в режиме Offline подсвечивается серым цветом (этот цвет выбран по умолчанию, и его можно поменять). Это состояние активируется, когда устройство отключено от сервера.

Имя действия:	trackerForVariable

Не интерактивный режим[99]: не поддерживается

Права доступа: Доступно на уровне[486] с правами доступа для *Менеджера*

Общие действия [?[1450]]

Создать на основе шаблона[105], копировать в дочерние контексты[111], импорт[108], экспорт[108], редактировать права доступа[106], просмотр событий[109], поиск/фильтрация[110], различные групповые действия[101] согласно контекстам потомков.

Состояние и иконки контекста

У этого контекста нет состояний[44]. Его всегда представляет иконка  .

Дополнительная информация

Информация о контексте

Тип контекста[43]: датчики

Имя контекста[42]: датчики

Описание контекста[43]: датчики

Путь контекста[42]: users.USER_NAME.trackers

Контекстная маска[44]: users.*.trackers

Права доступа к контексту [?[44]]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт датчиков.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [?[61]]

У этого контекста нет общих переменных (свойств).

Общие функции [?[70]]

Общие функции: makeCopy (сделать копию)[71], delete (удалить)[72]

СОЗДАТЬ

Создает новый датчик.

Имя функции: create

Права доступа: Доступно на уровне[486] с правами доступа для *Менеджера*

Записи ввода: 1

Формат[49] **ввода:** Такой же как и формат переменной childInfo[1597] в контексте датчика[1596].

Записи вывода: 0

Формат 49 **вывода:** нет

Общие события [? 73]

Общие события: info (информация) 77

ДАТЧИК

Это событие запускается, когда переоценивается значение датчика и его состояние.

Имя события	track
Права доступа:	Доступно на уровне 486 с правами доступа для *Наблюдателя*.
Период действия:	Непостоянный
Записи:	1

Формат 49 записи:

Имя поля	Тип поля	Примечания
name	Строка	Имя датчика.
value	Строка	Представление строки значения датчика.
status	Строка	Описание статуса датчика.
color	Цвет	Цвет, связанный со статусом датчика.

17.40.1 Действие: сконструировать датчик

Это пошаговый помощник по созданию датчика.

Порядок действий:

1. Выберите контекст, как описано в статье Действия по перетаскиванию мышью 101. Например, следует кликнуть по узлу Системного дерева, к которому следует применить это действие. Удерживайте левую кнопку мыши и перетащите ее на узел Избранное.
1. Выберите 90 переменную, состояние которой необходимо отслеживать.
2. Выберите 90 отслеживаемое поле этой переменной.
3. На этом этапе будет создан новый Датчик.
4. Отредактируйте 91 свойства только что созданного датчика, например, добавьте его статусы.

Тип действия:	Перетаскивание мышью 101
Имя действия:	build
Неинтерактивный режим 99**:**	не поддерживается
Права доступа:	Доступно на уровне 486 с правами доступа для *Менеджера*.

17.41 Датчик

Этот контекст[41] предоставляет Вам доступ к одному датчику[2181] и позволяет им управлять.

Уникальные действия [?[1450]]

НАСТРОИТЬ (действие по умолчанию[88])

Это действие используется для редактирования свойств[2183] датчика.

Изменение поля **Имя** во время этой операции меняет название данного контекста. Это может привести к сбою в работе других компонентов системы, использующих имя/путь контекста в качестве основного идентификатора.

Тип действия: настроить[105]

Общие действия [?[1450]]

Удалить[108], создать копию[109], реплицировать[110], редактировать права доступа[106], просмотр событий[109], показать статус[111]

Состояния и иконки контекста

Иконка	Код	Состояние
	0	Датчик включен
	1	Датчик отключен

Дополнительная информация

Информация о контексте

Тип контекста[43]: датчик

Имя контекста[42]: предоставляется пользователем

Описание контекста[43]: предоставляется пользователем

Путь контекста[42]: users.USER_NAME.trackers.TRACKER_NAME

Контекстная маска[44]: users.*.trackers.*

Права доступа к контексту [?[44]]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг датчиков. Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Конфигурация и удаление датчиков.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [?[61]]

Общие переменные: groupMembership (членство группы)[67], activeAlerts (активные тревоги)

СВОЙСТВА

См. описание переменной и ее полей здесь.

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*.

Формат записи:

Имя поля	Тип поля	Примечания
name	Строка	1 - 50 знаков
description	Строка	1 - 50 знаков
expression	Строка	
enabled	Булевое	
period	Длинное	Измеряется в миллисекундах

ТАБЛИЦА СТАТУСОВ

См. описание переменной и ее полей здесь.

Имя переменной: replication

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*

Формат записи:

Имя поля	Тип поля	Примечания
description	Строка	Один знак или более.
expression	Строка	
color	Цвет	Равно нулю.
level	Целое	

КАНАЛЫ СТАТИСТИКИ

Данная переменная позволяет управлять каналами статистики этого датчика.

Имя переменной: statisticsProperties

Записи: 0... не ограничено

Права доступа: Доступно для чтения на уровне с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*

Формат записи:

Имя поля	Тип поля	Примечания
name	Строка	Имя канала.
variable	Строка	Имя переменной, на которой базируется канал.
properties	Таблица данных	Свойства 719 канала.

ДАТЧИК

Эта переменная предоставляет доступ к текущему значению датчика. Если в параметрах датчика 2183 установлено ненулевое **время хранения истории**, изменения этой переменной остаются в БД и могут быть использованы для построения диаграмм и пр.

Имя переменной: tracker

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*

Формат 49 записи:

Имя поля	Тип поля	Примечания
value	Строка	Текущее значение датчика преобразовывается в строку.

СТАТИСТИКА

Эта переменная позволяет просматривать статистику датчика, т.е. агрегировать данные, собранные статистическим каналом 2184 датчика.

Имя переменной: statistics

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*

Формат 49 записи:

Имя поля	Тип поля	Примечания
name	Строка	Имя канала.
variable	Строка	Имя переменной, на которой базируется канал.
statistics	Таблица данных	Краткие статистические данные.

Общие функции [? 70]

У этого контекста нет общих функций.

Общие события [? 73]

Общие события: info (информация)

ИЗМЕНЕНИЕ СТАТУСА

Это событие запускается при изменении состояния датчика.

Имя события statusChange

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*.

Период действия: 100 дней

Записи: 1

Формат 49 записи:

Имя поля	Тип поля	Примечания
value	Строка	Представление строки текущего значения датчика.
status	Строка	Описание нового статуса датчика.

17.42 Обучаемый модуль

Этот контекст 41 позволяет получить доступ и управлять отдельным обучаемым модулем, который является потомком контекста Машинное обучение 865.

Уникальные действия [? 1450]

НАСТРОИТЬ

Действие настроить 105 используется, чтобы изменить свойства 870 обучаемого модуля.

Изменение поля **Имя** во время этой операции меняет название данного контекста. Это может привести к сбою в работе других компонентов системы, использующих имя/путь контекста в качестве основного идентификатора.

Тип действия: настроить 105

СБРОСИТЬ

Действие используется, чтобы сбросить состояние обучаемого модуля. Если обучаемый модуль уже обучен, он перестанет быть обученным в результате такого действия. Статистика вычисления (если такая накопилась) будет также удалена.

СБРОСИТЬ ВЫЧИСЛЕНИЕ

Это действие очищает накопленную статистику вычисления обучаемого модуля.

Общие действия [? 1450]

Удалить 106, создать копию 109, реплицировать 110, редактировать права доступа контекста 106, просмотр событий 109, показать статус 111

Состояния и иконки контекста

Иконка	Код	Состояние
	0	Обучаемый модуль не обучен.
	1	Обучаемый модуль доступен к обучению.

Дополнительная информация

Информация о контексте

Тип контекста[43]: trainableUnit

Имя контекста[42]: предоставляется пользователем

Описание контекста[43]: предоставляется пользователем

Путь контекста[42]: users.USER_NAME.machineLearning.TRAINABLE_UNIT_NAME

Права доступа к контексту [?[44]]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Просмотр настроек обучаемого модуля. Мониторинг основных событий. Просмотр статуса.
Оператор	Те же, что у Наблюдателя.
Менеджер	Удаление обучаемого модуля.
Инженер	Те же, что у Менеджера.
Администратор	Конфигурация обучаемого модуля.

Общие переменные (Свойства) [?[61]]

У этого контекста нет общих переменных.

Общие переменные: groupMembership (членство группы)[67], validity (пригодность)[68], activeAlerts (активные тревоги)[67]

СВОЙСТВА

См. описание переменной и его полей здесь[870].

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне[486] с правами доступа для *Наблюдателя*, доступно для записи с правами доступа для *Менеджера*.

Формат[49] записи:

Имя поля	Тип поля	Примечания
name	Строка	1 - 50 символов
description	Строка	1 - 50 символов
task	Строка	
algorithm	Строка	
hyperparameters	Таблица данных	Содержит гиперпараметры алгоритма[870].
labelFieldName	Строка	1 - 50 символов
hasWeights	Булевое	
weightsFieldName	Строка	1 - 50 символов или NULL

filters	Таблица данных	Содержит список внутренних фильтров для предварительной обработки данных. Каждый фильтр описан тремя полями: • Описание. Текстовое описание фильтра. Может показывать тип, назначение и другие характеристики. • Тип. Тип фильтра, выбранный из списка доступных фильтров 878. • Параметры. Параметры 878 фильтра.
cvNumFolds	Целочисленное	
cvRandomSeed	Целочисленное	

Общие функции [? 70]

ОБУЧАТЬ

Обучает модуль определенному учебному набору. Может обновить обучаемый модуль для обновляемых алгоритмов (Filtered Predictor, Stochastic Gradient Descent и Multiclass Updateable Classifer), используя новый учебный набор.

Имя функции: train

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*.

Записи входа: 1...без ограничений

Формат 50 входа: Динамический, зависит от структуры учебного набора. Для получения подробной информации обратитесь к разделу Аргументы функций обучаемого модуля 869.

Записи выхода: 1

Формат 50 выхода:

Имя	Тип	Описание
trainedUnitInfo	Строка	Информация об обучаемом модуле

ВЫПОЛНЯТЬ

Предугадывает значение целевой переменной для экземпляров данных текущего набора данных.

Имя функции: operate

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*

Записи входа: 0...без ограничений

Формат 50 входа: Динамический, зависит от структуры набора данных. Должен сходиться с форматом набора данных (учебным набором), который был передан функции Обучать.

Для получения подробной информации обратитесь к разделу Аргументы функций обучаемого модуля 869.

Записи выхода: 0...без ограничений

Формат 50 выхода: Состоящий из полей формата входа и из следующих полей:

Для проблем регрессии:

Имя	Тип	Описание
predicted	Двойной	Прогнозируемое значение целевой переменной (метка). Обратите внимание, что прогнозируемое значение всегда Двойного типа, хотя целевая переменная может быть любого цифрового типа.

error	Двойной	Различие между прогнозируемым значением и действительным значением целевой переменной. Если не дано действительное значение (равно нулю), то значение ошибки тоже будет NULL.

Для проблем классификации:

Имя	Тип	Описание
predicted	Строка	Прогнозируемый класс. Если у целевой переменной тип не Строка, то дается представление Строки значения класса.
probability	Двойной	Вероятность того, что этот экземпляр данных принадлежит прогнозируемому классу.

Для проблем определения аномалии:

Имя	Тип	Описание
predicted	Строка	Прогноз, будет ли текущий экземпляр данных "Нормальным" или "Аномальным".

ВЫЧИСЛЯТЬ

Вычисляет производительность обучаемого модуля с этим учебным набором и возвращает набор метрик вычисления, зависящих от задачи.

Имя функции: evaluate

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*

Записи входа: 1...без ограничений

Формат 50 входа: Динамический, зависит от структуры набора данных. Должен сходиться с форматом набора данных (учебным набором), который был передан в функцию Обучать.

Для получения подробной информации обратитесь к разделу Аргументы функций обучаемого модуля 869.

Записи выхода: 1

Формат 50 выхода: Для проблем регрессии:

Имя	Тип	Описание
correlation	Двойной	Коэффициент корреляции.
meanAbsoluteError	Двойной	Средняя ошибка.
rootMeanSquaredError	Двойной	Среднеквадратичная ошибка.
relativeAbsoluteError	Двойной	Относительная ошибка.
rootRelativeSquaredError	Двойной	Относительная среднеквадратичная ошибка.
unclassified	Длинный	Количество неклассифицированных экземпляров.
pctUnclassified	Двойной	Неклассифицированные экземпляры в процентном соотношении с полным количеством экземпляров.

totalNumInstances	Длинный	Общее число экземпляров.

Для проблем классификации и определения аномалии:

Имя	Тип	Описание
correct	Длинный	Количество верно классифицированных экземпляров.
pctCorrect	Двойной	Верно классифицированные экземпляры в процентном соотношении от общего числа экземпляров.
incorrect	Длинный	Количество неверно классифицированных экземпляров.
pctIncorrect	Двойной	Неверно классифицированные экземпляры в процентном соотношении от общего числа экземпляров.
unclassified	Длинный	Количество неклассифицированных экземпляров.
pctUnclassified	Двойной	Неклассифицированные экземпляры в процентном соотношении от общего числа экземпляров.
totalNumInstances	Длинный	Общее число экземпляров.
kappa	Двойной	Каппа-статистика.
meanAbsoluteError	Двойной	Средняя ошибка.
rootMeanSquaredError	Двойной	Среднеквадратичная ошибка.
relativeAbsoluteError	Двойной	Относительная ошибка.
rootRelativeSquaredError	Двойной	Относительная среднеквадратичная ошибка.
detailedAccuracy	Таблица данных	Метрики детальной точности по классам. Включают следующие поля: • Класс • Истинно-положительный показатель • Ложно-положительный показатель • Точность • Полнота • F-Мера • Коэффициент корелляции Matthews • Область под Кривой ROC (Receiver Operating Characteristic) • Область под Кривой Точность-Полнота
confusionMatrix	Таблица данных	Матрица неточности.

ПРОВЕРИТЬ ПЕРЕКРЕСТНО

Вычисляет производительность обучаемого модуля с этим набором данных с помощью кросс-валидации и возвращает набор метрик вычисления, характерных для задачи. В отличие от функции **вычислить**, обучаемый модуль может быть либо обученным, либо необученным. Функция не меняет состояние обучаемого модуля.

Имя функции:	crossValidate
Права доступа:	Доступно на уровне 486 с правами доступа для *Наблюдателя*
Записи входа:	1...без ограничений
Формат 50 входа:	Динамический, зависит от структуры набора данных. Для получения подробной информации обратитесь к разделу Аргументы функций обучаемого модуля 869.
Записи выхода:	1
Формат 50 выхода:	Тот же, что у функции **вычислить**.

СБРОСИТЬ

Действие используется, чтобы сбросить состояние обучаемого модуля. Если обучаемый модуль обучен этому, он станет необученным в результате этого действия. Эта функция вызывается действием **Сбросить**.

Имя функции:	reset
Права доступа:	Доступно на уровне 486 с правами доступа для *Наблюдателя*
Записи входа:	0
Формат 50 входа:	нет
Записи выхода:	0
Формат 50 выхода:	нет

СБРОСИТЬ ВЫЧИСЛЕНИЕ

Это действие очищает накопленную статистику вычисления обучаемого модуля. Эта функция вызывается действием **Сбросить вычисление**.

Имя функции:	resetEvaluation
Права доступа:	Доступно на уровне 486 с правами доступа для *Наблюдателя*
Записи входа:	0
Формат 50 входа:	нет
Записи выхода:	0
Формат 50 выхода:	нет

Общие события [? 73]

Общие события: info (Информация) 77

17.43 Пользователи

Данный контекст 41 - это контейнер, который содержит все контексты пользователя 1608 (учетные записи пользователя 478).

Уникальные действия [? 1450]

- Новый аккаунт пользователя 1607 (действие по умолчанию 88)
- Показать сводную таблицу пользователей 1608

Другие уникальные действия:

Общие действия [?1450]

Создать на основе шаблона 105, копировать в дочерние контексты 111, импорт 108, экспорт 108, редактировать права доступа 106, просмотр событий 109, поиск/фильтрация 110, различные групповые действия 101 согласно контекстам потомков.

Состояния и иконки контекста

У этого контекста нет состояний 44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста 43: пользователи

Имя контекста 42: пользователи

Описание контекста 43: пользователи

Путь контекста 42: users

Контекстная маска 44: users

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Нет доступа.
Оператор	Нет доступа.
Менеджер	Нет доступа.
Инженер	Нет доступа.
Администратор	Все операции.

Общие переменные (свойства) [? 61]

У этого контекста нет общих переменных (свойств).

Общие функции [? 70]

СПИСОК ПОЛЬЗОВАТЕЛЕЙ

Эта функция возвращает информацию обо всех Пользовательских учетных записях, доступ к которым осуществляется с текущими правами доступа. Свойства учетной записи описаны здесь 48.

Имя функции:	список
Права доступа:	Доступно на уровне 486 с правами доступа для *Наблюдателя*
Записи ввода:	0
Формат 49 ввода:	нет
Записи вывода:	0...не ограничено

Формат 49 **вывода:**

Имя	Тип	Описание
имя	строка	
firstname	строка	может не иметь значения
lastname	строка	может не иметь значения
password	строка	
country	целое	
region	строка	может не иметь значения
zip	строка	может не иметь значения
city	строка	может не иметь значения
address1	строка	может не иметь значения
address2	строка	может не иметь значения
comments	строка	может не иметь значения
company	строка	может не иметь значения
department	строка	может не иметь значения
email	строка	может не иметь значения
phone	строка	может не иметь значения
fax	строка	может не иметь значения
timezone	строка	
locale	строка	
datepattern	строка	
timepattern	строка	
dsautoregistration	булевое	

Общие события [? 73]

Общие события: info (информация)

ВХОД В СИСТЕМУ

Это событие генерируется при успешной авторизации пользователя 478.

Имя события login

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*

Период действия: 100 дней

Записи: 1

Формат 49 записи:

Имя поля	Тип поля	Примечания
username	String	Имя авторизовавшегося пользователя.
type	String	Тип связи (desktop, web, web-сервис и пр.)
address	String	IP-адрес клиента или имя хоста.
permissions	String	Права доступа для пользователя на момент авторизации.
login	String	Логин для входа пользователя в систему. Может отличаться от имени пользователя. Данное поле используется для внешней аутентификации 490.
sessionExpirationTIme	Date	Дата/время, когда истечет сессия пользователя.

ВЫХОД ИЗ СИСТЕМЫ

Это событие генерируется, когда пользователь 478 выходит из системы.

Имя события: logout

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*

Период действия: 100 дней

Записи: 1

Формат 49 записи:

Имя поля	Тип поля	Примечания
username	String	Имя пользователя, вышедшего из системы.
id	String	ID сессии.
type	String	Способ подключения (десктоп, web, web сервис, и т.д.)
address	String	IP адрес клиента или имя хоста.
login	String	Имя логина пользователя, вышедшего из системы. Может отличаться от имени пользователя. Данное поле используется для внешней аутентификации 490.

17.43.1 Действие: новая пользовательская учетная запись

Это действие вызова функции 103 используется для создания новой пользовательской учетной записи. Следует задать следующие параметры:

- имя пользователя
- пароль
- уровень 486 с правами доступа

Дополнительную информацию о настройках по умолчанию для новой учетной записи см в разделе пользователи 478.

Тип действия: функция вызова 103

Имя действия: создать

Иконка действия: 

17.43.2 Действие: просмотреть общую информацию о пользо

Это действие показывает 90 основные свойства 481 всех пользовательских учетных записей, которые доступны выполняющему это действие пользователю.

Тип действия: вызвать функцию 103

Имя действия: list

17.44 Пользователь

Этот контекст 41 предоставляет доступ к одной пользовательской учетной записи 478 и позволяет ею управлять.

Уникальные действия [?1450]

РЕДАКТИРОВАТЬ НАСТРОЙКИ АККАУНТА (действие по умолчанию 88)

Это действие используется для редактирования свойств 480 учетной записи.

Тип действия: настроить 105

УДАЛИТЬ АККАУНТ

Это стандартное действие удалить 106, которое используется для удаления пользовательской учетной записи.

Имя действия: удалить

Иконка действия: 

Не интерактивный режим 99: поддерживается

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*.

Общие действия [?1450]

Удалить 106, создать копию 109, реплицировать 110, редактировать права доступа 106, просмотр событий 109, показать статус 111

Иконки и состояние контекста

У этого контекста нет состояний 44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста 43: пользователь

Имя контекста[42]: предоставляется пользователем

Описание контекста[43]: предоставляется пользователем, содержит имя и фамилию пользователя.

Путь контекста[42]: users.USER_NAME

Контекстная маска[44]: users.*

Права доступа к контексту [?[44]]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Просмотр профайла пользователя. Редактирование фотографии пользователя.
Инженер	Те же, что у Менеджера.
Администратор	Редактирование профайла пользователя. Редактирование прав доступа пользователя. Удаление пользователя.

Общие переменные (свойства) [?[61]]

Общие переменные: groupMembership (членство группы)[67], activeAlerts (активные тревоги)

ИНФОРМАЦИЯ О ПОЛЬЗОВАТЕЛЕ

Возвращает свойства[481] учетной записи пользователя.

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне[486] с правами доступа для *Наблюдателя*

Формат[49] записи:

Имя	Тип	Описание
name	String	
firstname	String	может не иметь значения
lastname	String	может не иметь значения
password	String	
initializationPassword	String	
ignoreConnectionMode	Boolean	

useExternalAuthentication	Boolean	
country	Integer	
region	String	может не иметь значения
zip	String	может не иметь значения
city	String	может не иметь значения
address1	String	может не иметь значения
address2	String	может не иметь значения
comments	String	может не иметь значения
company	String	может не иметь значения
department	String	может не иметь значения
email	String	может не иметь значения
phone	String	может не иметь значения
fax	String	может не иметь значения
timezone	String	
locale	String	
datepattern	String	
timepattern	String	
dsautoregistration	Boolean	

ПРАВА ДОСТУПА

Возвращает права доступа 483 учетной записи пользователя.

Имя переменной: permissions

Записи: 0...не ограничено

Права доступа: доступно для чтения на уровне 486 с правами доступа для *Наблюдателя*,
доступно для записи на уровне 486 с правами доступа для *Администратора*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
mask	String	Контекстная маска, для которой применен уровень доступа.

type	String	Уровень прав доступа.

НАСТРОЙКИ АККАУНТА

Параметры учетной записи[482] пользователя.

Имя переменной: settings

Записи: 1

Права доступа: Доступно для чтения/записи на уровне[486] с правами доступа для *Администратора*.

Формат[49] записи:

Имя поля	Тип поля	Примечания
connectionRestrictionExpression	String	
enabled	Boolean	Вход в систему пользователя не разрешен, если данный флажок деактивирован.
activationTime	Date	Если не NULL, пользователю не разрешен вход в систему до этого времени.
expirationTime	Date	Если не NULL, пользователю не разрешен вход в систему после этого времени.
inheritPermissions	Boolean	
inheritPermissionsUser	String	
useRoleBasedPermissions	Boolean	

РЕСУРСЫ

Таблица доступных ресурсов[482] пользователя.

Имя переменной: resources

Записи: 0... не ограничено

Права доступа: Доступно для чтения/записи на уровне[486] с правами доступа для *Администратора*.

Формат[49] записи:

Имя поля	Тип поля	Примечания
description	String	Описание типа ресурса.
available	Boolean	Флажок, указывающий, будет ли учетная запись пользователя иметь узел контейнера для ресурсов такого типа.

ФОТОГРАФИЯ ПОЛЬЗОВАТЕЛЯ

Фотография[482] пользователя.

Имя переменной: photo

Записи: 1

Права доступа: Доступно для чтения/записи на уровне[486] с правами доступа для *Менеджера*.

Формат[49] записи:

Имя поля	Тип поля	Примечания
photo	Data Block	

СТАТУС ПОЛЬЗОВАТЕЛЯ

Возвращает статус[484] учетной записи пользователя.

Имя переменной: status

Записи: 1

Права доступа: Доступно для чтения на уровне[486] с правами доступа для *Наблюдателя*.

Формат[49] записи:

Имя поля	Тип поля	Примечания
creationtime	Date	Время создания учетной записи.
updatetime	Date	Время последнего изменения учетной записи.
logintime	Date	Время последнего входа в систему, или NULL, если пользователь никогда не входил в систему.

Общие функции [?[70]]

УДАЛИТЬ АККАУНТ

Навсегда удаляет учетную запись пользователя и все связанные данные.

Имя функции: delete

Права доступа: Доступно на уровне[486] с правами доступа для *Администратора.*

Записи ввода: 0

Формат[49] ввода: нет

Записи вывода: 0

Формат[49] вывода : нет

Общие события [?[73]]

Общие события: info (информация)[77]

17.45 Утилиты

Это системный контекст, который предоставляет различные операции. Он не отображается в видимой части контекстного дерева.

Уникальные действия [?1450]

ПОКАЗАТЬ ДАННЫЕ

Это действие выбирает данные из системы, подсчитывая выражение 112, показывает его пользователю при помощи GUI процедуры редактировать данные 90 и активирует автоматическое получение и автообновление отображаемых данных.

Имя действия:	showData
Не интерактивный режим 99:	не поддерживается
Права доступа:	Доступно на уровне 486 без прав доступа
Параметры 102 выполнения:	1. Сбор данных: • **Выражение** для подсчета • **Период обновления** 2. Представление данных: • **Заголовок** окна данных • **Помощь**, текст, который будет отображен в окне данных • **Иконка,** строка ID иконки, которая принадлежит иконке данных 3. Месторасположение окна данных

ПОКАЗАТЬ ОТЧЕТ

Это действие показывает отчет, основанный на Device пользователя. Использует процедуру пользовательского интерфейса Показать отчет 97, позволяет контролировать стандартные свойства шаблона оформления 930.

Имя действия:	showReport
Неинтерактивный режим 99:	не поддерживается
Права доступа:	Доступно на уровне 486 с правами доступа для *Наблюдателя*
Параметры 102 выполнения:	1. **Данные**: Device для создания отчета 2. **Свойства оформления**: свойства шаблона оформления 930. Если параметр Свойства оформления не указаны, пользователю необходимо изменить свойства шаблона в подходящем диалоге. После этого открывается окно отчета.

Пример привязки для использования данного действия без взаимодействия с пользователем:

Цель: `utilities:showReport!`

Выражение:
```
table("<<data><T>><<designProperties><T>>",
{form/dataTableEditor1:dataTable}, table("<<title><S>>",
"Some title"))
```

Активатор: `form/button1:click@`

Действия, связанные с переменной [?102]

ПОКАЗАТЬ ИСТОРИЮ ИЗМЕНЕНИЙ

Это действие показывает исторические изменения значения переменной. Оно выводит историю изменения переменной в форме таблицы. Каждое поле формата 49 отлеживаемой переменной отображается в отдельном поле итоговой таблицы. Если значение переменной содержит несколько рядов, отображается лишь первый ряд.

ПОКАЗАТЬ ИНФОРМАЦИЮ О ПЕРЕМЕННОЙ

Это действие отображает 90 свойства определения 61 переменной и ее формат.

Действия, относящиеся к событию [? 102]

ПРОСМОТРЕТЬ ИНФОРМАЦИЮ О СОБЫТИИ

Это действие показывает 90 свойства определения 61 события и его формат.

Дополнительная информация

Информация о контексте

Тип контекста 43: утилиты

Имя контекста 42: утилиты

Описание контекста 43: utilities

Путь контекста 42: utilities

Контекстная маска 44: utilities

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Основные внутренние операции системы.
Наблюдатель	Доступ и удаление истории переменных/событий. Доступ и удаление статистики переменных. Доступ к топологии устройств.
Оператор	Отправка E-mail и SMS сообщений.
Менеджер	Те же, что у Оператора.
Инженер	Запросы на проверку входящей почты.
Администратор	Выполнение внешнего приложения.

Общие переменные (свойства) [? 61]

У этого контекста нет общих переменных (свойств).

Общие функции [? 70]

ПОЛУЧИТЬ ИСТОРИЮ ПЕРЕМЕННОЙ

Возвращает историю изменений указанной переменной. См. просмотреть историю переменной 1559.

Имя функции: variableHistory

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*.

Записи ввода: 1

Формат 49 ввода:

Имя	Тип	Описание
mask	Строка	Контексты, где определена переменная.
event	Строка	Имя переменной.

fromDate	Дата	Временная метка, используемая для выбора лишь тех событий изменения переменной, которые происходили после определенной даты. Если соответствует NULL, будут выбраны все события изменений.
toDate	Дата	Временная метка, используемая для выбора лишь тех событий изменения переменной, которые происходили до определенной даты. Если соответствует NULL, будут выбраны все события изменений.
dataAsTable	Булевое	Если выключено (по умолчанию), таблица результатов будет содержать одно дополнительное поле для каждого поля переменной, чья история извлекается. В этом случае только первые ряды исторических значений буду добавляться к таблице результатов, другие ряды будут исключаться. Таким образом, данный режим подходит только для переменной с одной строкой. Если включено, исторические значения будут содержаться в отдельном поле таблицы под названием `vValue`.
limit	Целое	Максимальное количество извлекаемых исторических значений.
sortAscending	Булевое	Флажок, переключающий между восходящей и нисходящей сортировкой исторических значений.

Записи вывода: 0...не ограничено

Формат вывода: динамический, как минимум имеет поле `vUpdateTime`, содержащее временную метку исторических значений.

ВЫПОЛНИТЬ ВНЕШНЕЕ ПРИЛОЖЕНИЕ

Эта функция выполняет внешнее приложение, которое относится к аргументам ввода, ожидает его завершения и возвращает его вывод.

Имя функции: execute

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*

Записи ввода: 1

Формат ввода:

Имя	Тип	Описание
command	Строка	Полное квалификационное имя команды с аргументами.
directory	Строка	Рабочая директория, может быть опущена.
charset	Строка	Каноническое имя кодирования, как обозначено в API java.nio, может быть опущено. Список поддерживаемых кодировок можно найти в Oracle-документации.

Записи вывода: 1

Формат вывода:

Имя	Тип	Описание

exitCode	Целое	Код выхода команды.
output	Строка	Захват вывода команды.
errors	Строка	Захват вывода ошибки команды.

```
{utilities:execute("admin/executer.bat mkdir admin\\\\ myPath\\\\NewPath")}
```
создаст папку с путем `admin\myPath\NewPath`

ПЕРЕЧИСЛИТЬ ПЕРЕМЕННЫЕ

Эта функция возвращает список значений всех переменных с одной ячейкой (т.е. нетабличную) во всех контекстах, соответствующих определенной маске и принадлежащих определенной группе.

Имя функции: listVariables

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*

Записи ввода: 1

Формат 49 ввода:

Имя	Тип	Описание
mask	Строка	Маска контекста, согласно которой перечисляются переменные.
group	Строка	Группа переменных.

Записи вывода: 0...не ограничено

Формат 49 вывода:

Имя	Тип	Описание
context	Строка	Описание контекста.
variable	Строка	Описание переменной.
value	Строка	Представление строки значения переменной.

СТАТИСТИКА

Эта функция возвращает последние значения для статистического канала 718 (например, среднее значение последнего месяца, минимальное последнего дня и пр.)

Имя функции: статистика

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*

Записи ввода: 1

Формат 49 ввода:

Имя	Тип	Описание
mask	Строка	Маска контекста для выбора статистических данных.
channel	Строка	Имя канала.
key	Строка	Ключ набора данных или NULL для использования набора данных по умолчанию.
period	Строка	Период времени для отображения последних значения. Если NULL, будут выбраны данные для

		всех периодов. Имена периодов: • **millisecond** или **ms** для миллисекунд • **second**, **sec** или **s** для секунд • **minute**, **min** или **m** для минут • **hour**, **hr** или **h** для часов • **day** или **d** для дней • **week** или **w** для недель • **month** для месяцев (обратите внимание, что месяц основан на нуле, т.е. значение для января -- 0.) • **year** или **y** для года
full	Булевое	Если верно (true), все статистические данные возвращаются согласно периоду, указанному ранее (т.е. среднечасовой). Если неверно (false), возвращаются только последние собранные данные (например, последнее среднее целого часа).
average	Булевое	Показать среднее значение для последнего периода выбранного типа. По умолчанию оно является true.
minimum	Булевое	Показать минимальное значение для последнего периода выбранного типа. По умолчанию оно является true.
maximum	Булевое	Показать максимальное значение для последнего периода выбранного типа. По умолчанию оно является true.
sum	Булевое	Показать итоговое значение для последнего периода выбранного типа. По умолчанию оно является true.
first	Булевое	Показать первое значение для последнего периода выбранного типа. По умолчанию оно является true.
last	Булевое	Показать последнее значение для последнего периода выбранного типа. По умолчанию оно является true.

Записи вывода: 0...не ограничено

Формат 49 **вывода:**

Имя	Тип	Описание
context	Строка	Имя контекста, который породил данные.
start	Дата	Начало промежутка времени.
end	Дата	Конец промежутка времени.
key	Строка	Ключ набора данных или NULL, если используется набор данных по умолчанию.
average	Плавающее	Среднее значение для промежутка времени.

minimum	Плавающее	Минимальное значение для промежутка времени.
maximum	Плавающее	Максимальное значение для периода времени.
sum	Плавающее	Итоговое значение для периода времени.
first	Плавающее	Первое значение для периода времени.
last	Плавающее	Последнее значение для периода времени.

СЫРЫЕ СТАТИСТИЧЕСКИЕ ДАННЫЕ

Эта функция возвращает сырые статистические данные для статистического канала 718.

Имя функции: rawStatistics

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*.

Записи ввода: 1

Формат 49 ввода:

Имя	Тип	Описание
context	Строка	Имя контекста для выбора статистических данных.
name	Строка	Имя канала для выбора статистических данных.

Записи вывода: 0...не ограничено

Формат 49 вывода: динамический

УДАЛИТЬ СТАТИСТИКУ

Эта функция используется для удаления всех данных, собранных каналом статистики 718.

Имя функции: deleteStatistics

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*.

Записи ввода: 1

Формат 50 ввода:

Имя	Тип	Описание
mask	Строка	Маска контекстов для поиска статистических каналов.
channel	Строка	Имя канала, откуда удаляются данные.

Записи вывода: 0

Формат 50 вывода: Отсутствует

ДОПУСТИМЫЕ ЗНАЧЕНИЯ

Эта функция возвращает таблицу данных, подходящую для использования в качестве допустимых значений 51 другого поля таблицы данных. Она чаще всего используется в выражении привязки таблицы данных 741, нацеленной на свойство выбора 741 поля.

Функция сперва выстраивает промежуточную таблицу данных, оценивая выражение. Затем она проходит через эту таблицу по строкам и оценивает два других выражения для каждого ряда. Первое возвращает само допустимое значение, а второе возвращает ее описание.

Имя функции: selectopnValues

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*.

Записи ввода: 1

Формат 50 ввода:

Имя	Тип	Описание
tableExpression	Строка	Выражение, которое должно вернуть таблицу, чьи записи будут использоваться для построение списка допустимых значений.
valueExpression	Строка	Выражение, которое рассчитывется поверх каждой записи вышеобозначенной таблицы и должно вернуть представление допустимых ззначений в виде строки.
descriptionExpresssion	Строка	Выражение, которое рассчитывается поверх той же записи и должно вернуть описание допустимого значения.

Записи вывода: 0... не ограничено

Формат 50 вывода:

Имя	Тип	Описание
value	Строка	Представление допустимых значений в виде строки.
description	Строка	Описание допустимых значений.

ОБОБЩЕНИЕ

Эта функция собирает данные, рассчитывая средние, максимальные, минимальные, общие значения и другие метрики по источнику и временному периоду. Числовые значения источника можно извлечь из:

- Исторических событий 73 контекста, хранящихся в базе данных сервера 692
- Исторических значений переменной 61 контекста (также загружаемых из базы данных сервера)
- Данные, собранные статистическим каналом 718

Этот ввод функции представляет из себя таблицу со многими строками. Каждая строка в таблице ввода опеделяет отчетливые *серии данных*. Каждая серия данных будет представлена отдельным столбцом в выводе функции, например, "средние значения для переменной A за период и на источник" или "количество образцов для события B за период и на источник".

Имя функции: summary

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*.

Записи ввода: 0... не ограничено

Формат 50 ввода:

Имя	Тип	Описание
seriesName	Строка	Имя серии данных, например, имя столбца серии в выводе функции. Должна содержать только буквы, цифры и нижние подчеркивания.
seriesDescription	Строка	Удобное описание серии, например, описание поля вывода.
startDate	Дата	Самая ранняя дата первого образца данных, который будет обрабатываться для серии.
endDate	Дата	Самая поздняя дата последнего образца данных, который будет обрабатываться для серии.

masks	Строка	Отделенный пробелами список масок контекста 44, соответствующих источникам, откуда загружаются серии. Примеры: • `users.*.devices.*` - все доступные устройства для вызова функции • `users.john.devices.dev1 users.john.devices.dev2` - два отдельных устройства
entity	Строка	Имя сущности контекста (т.е. переменной или функции), чьи исторические значения будут обрабатываться по сериям.
entityType	Целое	Имя сущности контекста, чьи исторические значения будут обрабатываться по сериям. Может быть Переменной или Функцией, см. ссылку на ее числовой код здесь 2160.
expression	Строка	Выражение 112 AtomMind, которое должно вернуть число. Выражение используется для: • Извлекать числовые образцы данных из таблицы данных 49, представляющей хронологические события и значения переменной • Обрабатывать ранее собранные образцы данных, содержащиеся в статистическом канале (если серия относится к статистике) Если выражение возвращает NULL, обработанный образей данных удаляется. Среда вычисления 114 выражения серии: Контекст по умолчанию 119 — Контекст, чья история/статистика обрабатывается. Таблица данных по умолчанию 120 — Таблица данных, представляющая обрабатываемое в данный момент историческое событие или значение переменной. Строка по умолчанию 119 — 0 Переменные среды 123 — Только стандартные 123 переменные.
dateExpression	Строка	Выражение 112 AtomMind, которое должно возвращать дату. Оно используется для извлечения дат сбора образцов из таблицы данных 49, представляющей исторические события и значения переменных. Выражение NULL (значение по умолчанию, подходящее для большинства случаев) заставляет функцию использовать временные

		метки, указывающие, когда исторические события и значения переменных записывались в базу данных сервера. Среда вычисления 114 выражения даты серии: Контекст по умолчанию 119 — Контекст, чья история/статистика обрабатывается Таблица данных по умолчанию 120 — Таблица данных, представляющая обрабатываемое в данный момент историческое событие или значение переменной. Строка по умолчанию 119 — 0 Переменные среды 123 — Только стандартные 123 переменные.
grouping	Целое	Тип временного периода, используемый для группировки данных. См. доступные типы здесь 2160.
aggregation	Целое	Тип сбора числовых значений. См. доступные типы здесь 2160.
changeType	Целое	Типы серий. См. описание доступных типов здесь 1062. Коды числовых типов: • 0 - датчик • 1 - счетчик • 2 - счетчик без переполнения • 3 - абсолютный
outOfRangeValuesHandling	Целое	Определяет, как обрабатываются выходящие за пределы диапазона значения (т.е. значения типа датчика или посекундные значения, рассчитываемые в соответствии с вышеуказанными типами серий): • 0 - **Игнорировать**, т.е. не руководствоваться приведенными ниже минимальными и максимальными значениями • 1 - **Удалить**, т.е. не обрабатывать данные, выходящие за пределы приведенного ниже диапазона • 2 - **Нормализовать**, т.е. перевести низкие и высокие значения, выходящие за пределы диапазона, в нижние и верхние границы
minValue	Двойное	Минимально разрешенное значение типа датчика или посекундные данные.

maxValue	Двойное	Максимально разрешенное значение типа датчика или посекундные данные.
timeZone	Строка	Временная зона, используемая для выравнивания временных периодов в таблице вывода. Если она равна null, используется временная зона AtomMind Server по умолчанию. Временные зоны могут определяться в форме строки, например: • GMT-8 • GMT-08:00 • America/Los_Angeles  Указание различных временных зон для различных серий данных не поддерживается и может выдавать непредсказуемые результаты.
showDetails	Булевое	Определяет, будут ли добавляться данные по временному периоду к выводу функции. Использоваться будут только значения, определенные для первой серии (например, в первой строке вывода функции), другие значения будут игнорироваться.
showTotals	Булевое	Определяет, могут ли все общие суммы быть добавлены к выводу функции. Использоваться будет лишь значение, определенное первой серией (например, в первом ряду ввода функции), другие значения будут игнорироваться.
sortPriority	Целое	Определяет метод сортировки вывода функции: • 0 - сначала группирует по временным периодам, затем по источникам • 1 - сначала группирует по источникам, затем по временным периодам

Записи вывода: 0... не ограничено

Формат 50 вывода: Всегда доступные поля статического вывода:

Имя	Тип	Описание
periodName	Строка	Читаемое имя временного периода, чьи собранные данные представлены в виде записи вывода.
periodMiddle	Дата	Первая миллисекунда временного периода, чьи собранные данные представлены в виде записи вывода.
periodEnd	Дата	Средняя миллисекунда временного периода, чьи собранные данные представлены в виде записи вывода.
periodStart	Дата	Последняя миллисекунда временного периода, чьи собранные данные представлены в виде записи вывода.
context	Строка	Путь контекста 41 источника, чьи собранные данные представлены в виде записи вывода.

Дополнительные столбцы вывода функции определяются серией данных, указанной в функциональном вводе:

Имя	Тип	Описание
имя, определяемое полем ввода `seriesName`	• Дата для серий с использованием типов объединения **Первая дата** и **Последняя дата**. • Целое для серий с использованием типа объединения **Количество**. • Двойной для других серий.	Каждый динамический столбец содержит объединения по отдельному источнику, отдельному временному периоду, рассчитанные согласно параметрам серии.

СГЕНЕРИРОВАТЬ СОБЫТИЕ ЗАДНИМ ЧИСЛОМ

Эта функция используется для формирования событий, время создания которых находится в прошлом. Она подходит для обновления истории событий, полученных от сторонних источников с задержкой.

Имя функции: fireBackdatedEvent

Права доступа: Доступно на уровне 486 с правами доступа для *Администратора*

Записи ввода: 0...unlimited

Формат 50 ввода:

Имя	Тип	Описание
context	Строка	Путь контекста, в котором генерируется событие.
event	Строка	Имя генерируемого события.
level	Целое	Уровень события или ноль для использования уровня по умолчанию.
creationTime	Дата	Дата события в прошлом. Будет сохранена в базе данных сервера.
data	Таблица данных	Специфичные данные по событию. Должны соответствовать формату, заданному в определении события.

Записи вывода: 0

Формат 50 вывода: Отсутствует

Общие события [? 73]

Общие события: info (информация) 77

17.46 Виджеты

Данный контекст[41], является контейнером, который вмещает все виджеты[943] для определенного пользователя.

Уникальные действия [?1450]

СОЗДАТЬ НОВЫЙ ВИДЖЕТ (действие по умолчанию[88])

Это действие по перетаскиванию мышью[101] используется для создания нового виджета. Оно описано здесь[945].

Тип значения:	перетаскивание мышью[101]
Иконка действия:	
Имя действия:	create
Не интерактивный режим[99]:	не поддерживается
Права доступа:	Доступно на уровне[486] с правами доступа для *Менеджера*

СОЗДАТЬ КАРТУ УСТРОЙСТВ

Это действие используется для создания виджета Карты устройств.

Поток действия:

1. Задайте[90] свойства карты (имя, описание, фоновое изображение, размер изображения устройства и пр.).
2. Выберите[94] устройства для отображения на карте.
3. Отредактируйте новую карту в редакторе виджетов[423].

Имя действия:	createMap
Не интерактивный режим[99]:	не поддерживается
Права доступа:	Доступно на уровне[486] с правами доступа для *Менеджера*

Общие действия [?1450]

Создать на основе шаблона[105], копировать в дочерние контексты[111], импорт[108], экспорт[108], редактировать права доступа[106], просмотр событий[109], поиск/фильтрация[110], различные групповые действия[101] согласно контекстам потомков.

Действия, связанные с переменной [?102]

СОЗДАТЬ ДИАГРАММУ

Это действие создает новый виджет с компонентом диаграммы[1051] и несколькими другими компонентами для осуществления контроля за параметрами диаграммы в реальном времени. Эти компоненты формируют инструментальную панель.

Диаграмма настраивается для отображения исторических изменений значений переменной.

На первом этапе необходимо указать тип диаграммы:

- Последние значения и/или диаграмма изменений в реальном времени
- Диаграмма хронологических событий
- Диаграмма использования для значений типа счетчик

На втором этапе должны быть заданы следующие параметры:

- Тип диаграммы (линейная, с областями или столбчатая)

- Инструментальная панель (включена/отключена)
- Одно из двух заранее определенных выражений пригодности 946, которые позволяют использовать диаграмму **только с текущим контекстом** или **со всеми контекстами этого же типа.**

Затем следует создать один или более трендов диаграммы с именами. Каждый тренд определяется выражением, которое может относится к одному и более полям переменной.

Имя действия: chartForVariable

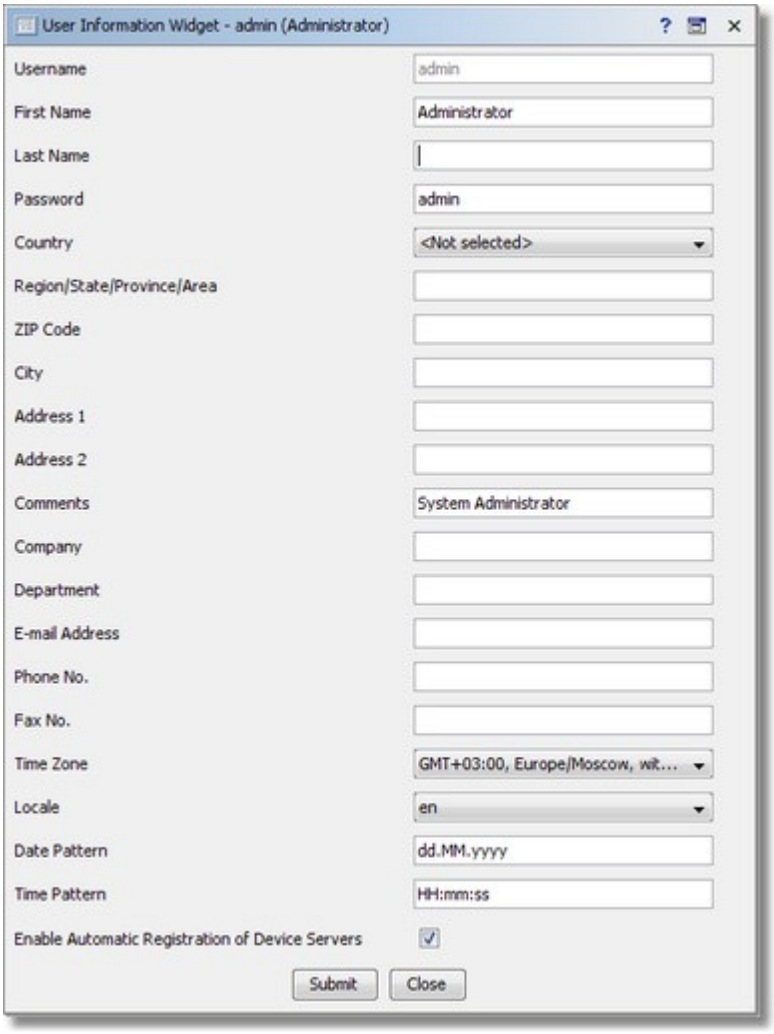
Не интерактивный режим 99**:** не поддерживается

Права доступа: Доступно на уровне 486 с правами доступа для *Менеджера*

СОЗДАТЬ ВИДЖЕТ

Это действие генерирует новый виджет, который можно использовать для редактирования значения переменной. Только что созданный виджет можно настроить непосредственно сразу после создания так, чтобы он имел удобный для пользователя вид.

Шаблон 946 виджета, произведенный данным действием лучше всего подходит для редактирования значений сложных переменных, формат которых определяет множество полей, но один ряд. Шаблон состоит из главной панели с двумя колонками и панелью кнопок, расположенной внизу. Левая колонка главной панели отображает имена полей, использующих ярлыки 956 виджетов. Правая колонка содержит различные компоненты, используемые для просмотра/редактирования значений поля. Внизу расположены две кнопки: **Submit** (кнопка подтверждения), которая заставляет виджет подтвердить изменения и выйти, и **Close** (кнопка закрытия), которая останавливает виджет без сохранения изменений. Вот как выглядит автоматически созданный виджет для переменной пользовательского контекста Свойства учетной записи пользователя 481:



Имя действия: widgetForVariable

Не интерактивный режим: Не поддерживается

Права доступа: Доступно на уровне с правами доступа для *Менеджера*

Действия, связанные с событием [?]

СОЗДАТЬ ДИАГРАММУ

Это действие создает новый виджет с компонентом диаграммы и несколькими другими компонентами для осуществления контроля за параметрами диаграммы в режиме реального времени. Эти компоненты формируют инструментальную панель.

Диаграмма настраивается для отображения значения данных, взятых из истории текущего события.

Сначала следует задать следующие параметры:

- тип диаграммы (линейная, с областями или столбчатая)
- инструментальная панель (включено/отключено)
- одно из двух заранее определенных выражений пригодности, которые позволяют использовать диаграмму **только с текущим контекстом** или со **всеми контекстами этого же типа.**

Затем следует создать один или более трендов диаграммы с именами. Каждый тренд опредяется выражением, которое может относится к одному и более полям переменной.

Имя действия: chartForEvent

Не интерактивный режим: не поддерживается

Права доступа: Доступно на уровне с правами доступа для *Менеджера*

Состояния и иконки контекста

У этого контекста нет состояний. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста: виджеты

Имя контекста: widgets

Описание контекста: widgets

Путь контекста: users.USER_NAME.widgets

Контекстная маска: users.*.widgets

Права доступа к контексту [?]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт виджетов.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [?]

У этого контекста нет общих переменных (свойств).

Общие функции [?70]

Общие функции: makeCopy (скопировать)71, delete (удалить)72

СОЗДАТЬ

Создает новый виджет.

Имя функции:	create
Права доступа:	Доступно на уровне486 с правами доступа для *Менеджера*
Записи ввода:	1
Формат49 ввода:	Аналогичный формату переменной childInfo1628 в контексте виджета1627.
Записи вывода:	0
Формат49 вывода:	нет

Общие события [?73]

Общие события: info (информация)77

17.47 Виджет

Этот контекст41 предоставляет Вам доступ к одному виджету943 и позволяет им управлять.

Уникальные действия [?1450]

НАСТРОИТЬ

Действие настроить105 используется для редактирования свойств947 виджета.

Изменение поля **Имя** во время этой операции меняет название данного контекста. Это может привести к сбою в работе других компонентов системы, использующих имя/путь контекста в качестве основного идентификатора.

Тип действия:	настроить105

РЕДАКТИРОВАТЬ ШАБЛОН (действие по умолчанию88 или относительные946 виджеты)

Это действие поддерживается только в AtomMind Client359. Оно запускает редактор виджетов423 и позволяет редактировать шаблон946 виджетов.

Имя действия:	editTemplate
Неинтерактивный режим99:	не поддерживается
Права доступа:	Доступно на уровне486 с правами доступа для *Менеджера*.

ЗАПУСТИТЬ ВИДЖЕТ (действие по умолчанию88 для абсолютных946 виджетов)

Это действие запускает виджет. См. Запуск виджетов948.

Имя действия:	launch
Не интерактивный Mode99:	не поддерживается
Права доступа:	Доступно на уровне486 с правами доступа для *Наблюдателя*.

Параметры выполнения: Месторасположение окна виджета

Свойства инструментальной панели виджета

Общие действия [?]

Удалить, создать копию, реплицировать, редактировать права доступа, просмотр событий, показать статус

Состояния и иконки контекста

Иконка	Код	Состояние
	0	Этот виджет относительный.
	1	Этот виджет абсолютный.

Дополнительная информация

Информация о контексте

Тип контекста: виджет

Имя контекста: предоставляется пользователем

Описание контекста: предоставляется пользователем

Путь контекста: users.USER_NAME.widgets.WIDGET_NAME

Контекстная маска: users.*.widgets.*

Права доступа к контексту [?]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Запуск виджета. Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Конфигурация и удаление виджета.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [?]

Общие переменные: groupMembership (членство группы), validity (пригодность), activeAlerts (активные тревоги)

СВОЙСТВА ВИДЖЕТА

См. описание переменной и ее полей здесь.

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне с правами доступа для *Наблюдателя*, доступно для записи с правами доступа *Менеджера*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
name	Строка	1 - 50 символов
description	Строка	1 - 50 символов
template	Строка	
type	Строка	
validityExpression	Строка	
validityListeners	Таблица данных	
defaultContext	Строка	

Общие функции [? 70]

У этого контекста нет общих функций.

Общие события [? 73]

Общие события: info (информация) 77

17.48 Процессы

Этот контекст 41 является контейнером, содержащим все процессы 895 для определенного пользователя.

Уникальные действия [? 1450]

СОЗДАТЬ (действие по умолчанию 88)

Это действие используется для создания нового процесса. Оно позволяет пользователю определить основные свойства 815 для нового процесса и настроить их сразу после создания.

Тип действия: создать 105

Права доступа: Доступно на уровне 486 прав доступа для *Менеджера*

Общие действия [? 1450]

Создать на основе шаблона 105, копировать в дочерние контексты 111, импорт 108, экспорт 108, редактировать права доступа 106, просмотр событий 109, поиск/фильтрация 110, различные групповые действия 101 согласно контекстам потомков.

Состояния и иконки контекста

У этого контекста нет состояний 44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста 43: workflows

Имя контекста 42: workflows

Описание контекста[43]: процессы

Путь контекста[42]: users.USER_NAME.workflows

Маска контекста[44]: users.*.workflows

Права доступа к контексту [?[44]]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт процесса.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [?[61]]

У этого контекста нет общих переменных (свойств).

Общие функции [?[70]]

Общие функции: makeCopy (сделать копию)[71], delete (удалить)[72]

СОЗДАТЬ

Создает новый процесс.

Имя функции: create

Права доступа: Доступно на уровне[486] прав доступа для *Менеджера*

Записи ввода: 1

Формат[50] ввода: Тот же, что формат переменной childInfo[1540] в контексте Процесс[1630].

Записи вывода: 0

Формат[50] вывода: нет

Общие события [?[73]]

Общие события: info (информация)[77]

17.49 Процесс

Этот контекст[41] позволяет получить доступ и управлять одним процессом[895].

Уникальные действия [?[1450]]

НАСТРОИТЬ

Данное действие настроить[105] используется для редактирования свойств[815] процесса.

Изменение поля **Имя** во время этой операции меняет название данного контекста. Это может привести к сбою в работе других компонентов системы, использующих имя/путь контекста в качестве основного идентификатора.

Тип действия: настроить

РЕДАКТИРОВАТЬ ШАБЛОН (действие по умолчанию)

Это действие поддерживается только в AtomMind Client. Оно запускает редактор виджетов и позволяет редактировать шаблон процесса.

Имя действия:	editTemplate
Неинтерактивный режим:	не поддерживается
Права доступа:	Доступно на уровне с правами доступа для *Менеджера*.

ЗАПУСТИТЬ

Это действие запускает процесс. См. Запуск процессов.

Имя действия:	launch
Не интерактивный Mode:	поддерживается
Права доступа:	Доступно на уровне с правами доступа для *Наблюдателя*.
Параметры выполнения:	Месторасположение окна виджета Свойства инструментальной панели виджета

Общие действия [?]

Удалить, сделать копию, реплицировать, редактировать права доступа контекста, просмотреть события, просмотреть статус

Состояния и иконки контекста

У этого контекста нет состояний. Он представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста: workflows

Имя контекста: предоставляется пользователем

Описание контекста: предоставляется пользователем

Путь контекста: users.USER_NAME.workflows.WORKFLOW_NAME

Маска контекста: users.*.workflows.*

Права доступа к контексту [?]

Уровень	Описание
Отсутствует	Нет доступа.
Наблюдатель	Просмотр конфигурации модели. Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Удаление модели.
Инженер	Те же, что у Менеджера.

Администратор	Конфигурация модели.

Общие переменные (свойства) [?61]

Абсолютные811 процессы добавляют заявленные переменные в свой контекст. Поэтому создается одна переменная на каждую запись в таблице модели Переменных817.

Общие переменные: groupMembership (членство группы)67, validity (пригодность)68, activeAlerts (активные тревоги)

СВОЙСТВА

См. описание переменной и ее поле здесь815.

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне486 с правами доступа для *Наблюдателя*, доступно для редактирования на уровне с правами доступа для *Менеджера*

Формат50 записи:

Имя поля	Тип поля	Примечания
name	Строка	1 - 50 символов
description	Строка	1 - 50 символов
template	Строка	
normalConcurrentBindings	Целое	
maximumConcurrentBindings	Целое	
maximumBindingQueueLength	Целое	
multiUserWorkflowExecution	Булевое	
logWorkflowExecution	Булевое	

СТАТИСТИКА ПУЛА ПОТОКОВ

Возвращает статистическую информацию о пуле потоков процесса.

Имя переменной: threadPoolStatus

Записи: 0... не ограничено

Права доступа: Доступно для чтения на уровне486 с правами доступа для *Наблюдателя*

Формат50 записи:

Имя поля	Тип поля	Примечания

activeCount	Целое	Количество активных задач.
completedCount	Длинное	Количество завершенных задач.
totalCount	Длинное	Общее количество задач.
coreSize	Целое	Размер ядра пула.
largestSize	Целое	Самый большой (пиковый) размер пула.
maximumSize	Целое	Максимально разрешенный размер пула.
queueLength	Целое	Длина очереди задач.

Общие функции [?70]

АВТОНОМНОЕ ВЫПОЛНЕНИЕ ПРОЦЕССА

Запускает процесс. В качестве входного параметра передается таблица, которая будет использоваться как таблица по умолчанию 898.

Имя функции: execute

Права доступа: Доступно на уровне 486 с правами доступа для *Инженера*.

Записи ввода: 1

Формат 49 ввода:

Имя	Тип	Описание
defaultDataTable	Таблица данных	Имя таблицы

Записи вывода: 0...не ограничено

Формат 49 вывода: нет

Общие события [?73]

ВЫПОЛНЕНИЕ ПРИВЯЗОК

Это событие генерируется каждый раз при выполнении привязки модели.

Имя события: bindingExecution

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*

Период действия: Непостоянный

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
context	Строка	Контекст процесса
target	Строка	Цель привязки.
expression	Строка	Выражение привязки.

value	Строка	Представление результата выражения привязки в виде строки.
activator	Строка	Активатор привязки.
condition	Строка	Состояние привязки.
execution	Строка	Режим выполнения привязки (при запуске, при событии или периодически).
cause	Строка	Причина выполнения привязки (ссылка на событие или измененную переменную, которая вызвала выполнение).

ОШИБКА ПРИВЯЗКИ

Это событие генерируется каждый раз, когда привязка модели не выполняется и выдает ошибку.

Имя события: bindingError

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*

Период действия: Непостоянный

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
context	Строка	Контекст процесса.
target	Строка	Цель привязки.
expression	Строка	Выражение привязки.
activator	Строка	Активатор привязки.
execution	Строка	Режим выполнения привязки (при запуске, при событии или периодически).
cause	Строка	Причина выполнения привязки (ссылка на событие или измененную переменную, которая вызвала выполнение).
error	Строка	Текст сообщения об ошибке.
stack	Таблица данных	Таблица, которая содержит трассировку стека ошибки привязки.

ВЫПОЛНЕНИЕ БЛОКОВ

Это событие генерируется каждый раз при выполнении блоков процесса.

Имя события: workflowExecution

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*

Период действия: Непостоянный

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания

workflowExecutionElement	Строка	Имя текущего исполняемого блока
workflowExecutionInputValue	Строка	Значение переменной среды 123 с именем value

ОШИБКА БЛОКА

Это событие генерируется каждый раз, когда блок задачи процесса не выполняется и выдает ошибку.

Имя события: workflowError

Права доступа: Доступно на уровне 486 с правами доступа для *Наблюдателя*

Период действия: Непостоянный

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
target	Строка	Имя блока в котором возникла ошибка
error	Строка	Текст сообщения об ошибке.
stack	Таблица данных	Таблица, которая содержит трассировку стека ошибки привязки.

Общие события: info (информация) 77

18 Практические примеры

Порой не хочется изучать теорию и внутренние механизмы системы, разбираться, как AtomMind Server использует алгоритмы для выполнения той или иной операции; иногда просто хочется сразу приступить к практике и *сделать что-то*. Соответственно Вы хотите, чтобы мы Вам рассказали, как это сделать.

Поэтому эти практические уроки для Вас. Здесь Вы не найдете подробного описания или теоретических основ - лишь простые, выполняемые в несколько этапов операции, которые выполняются в AtomMind Client 359 или ином пользовательском интерфейсе AtomMind Servera.

Время от времени мы будем добавлять новые статьи в этот раздел. Возможно, причиной для этого будут поступающие запросы о технической поддержке относительно работы системы или какого-то ее компонента.

Важное примечание

Учитывая специфику данного раздела, следует принять во внимание следующие важные моменты:

- Тема, которую вы ищете, возможно еще не существует в этом разделе. Если это так, прочитайте полностью тот раздел руководства, который описывает искомую тему. Возможно, там содержится ответ на Ваш вопрос, и может быть, даже с примерами.
- Практические уроки написаны специально для AtomMind Client. Аналогичные процедуры можно выполнить, используя и другие пользовательские интерфейсы AtomMind Servera.
- В разделе не содержатся теоретические выкладки. Не следует ожидать, что после прочтения практических уроков, Вы сразу все поймете. Для более глубокого понимания требуется прочтение всей темы руководства. Мы постарались включить как можно больше ссылок на информативные страницы.
- И самое важное: изучение лишь практических уроков **может не сработать**, поскольку AtomMind Server - это сложная система; при написании практических уроков мы стремились охватить основные рабочие моменты. Возможно, некоторые настройки на Вашей системе или в сети отличаются, и это может привести к неожиданным результатам. Если это происходит, пожалуйства, прочитайте соответствующий раздел руководства по той проблеме, которую Вы пытаетесь решить.
- Если все попытки решить проблему оказались неуспешными, обратитесь в службу технической поддержки. См информацию на сайте ТВЭЛ.

18.1 Действия с множеством устройств

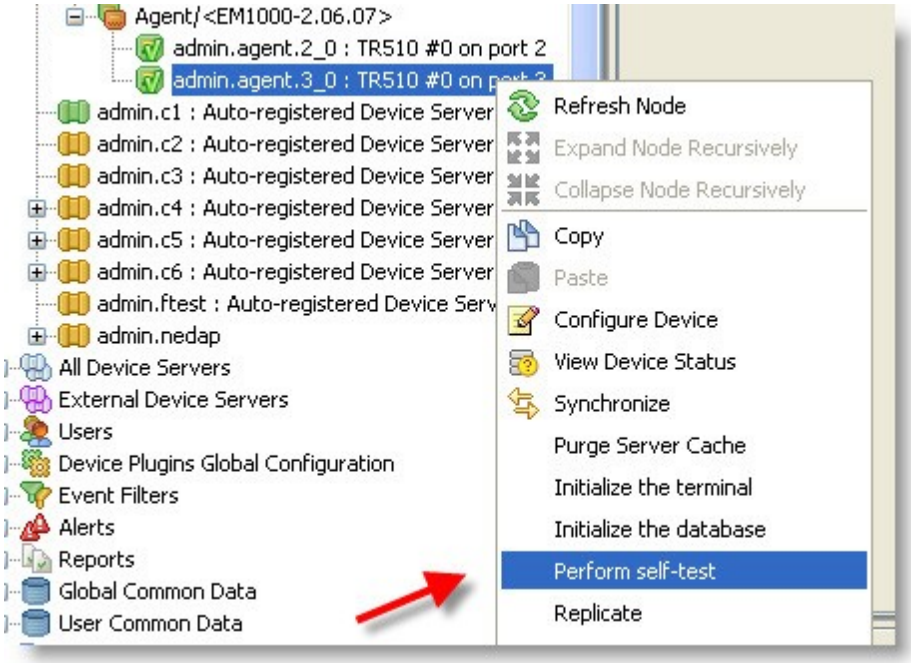
В этом уроке объясняется, как одно и то же действие с множеством устройств можно выполнять при помощи мыши, и используя Избранное 2186.

Давайте предположим, что у Вас может быть много устройств, которые поддерживают операцию самотестирования. Эта операция может заставить каждое устройство проводить диагностику его внутреннего состояния. Каждое устройство может выводить результаты самотестирования на свой дисплей или отправлять отчет AtomMind.

Поскольку операторы часто запускают самотестирование устройства, его выполнение должно быть простым - посредством пары кликов мышью. Более того, очень удобно проводить самотестирование сразу у нескольких устройств. Свойство Избранное 2186 идеально подходит для выполнения этой задачи.

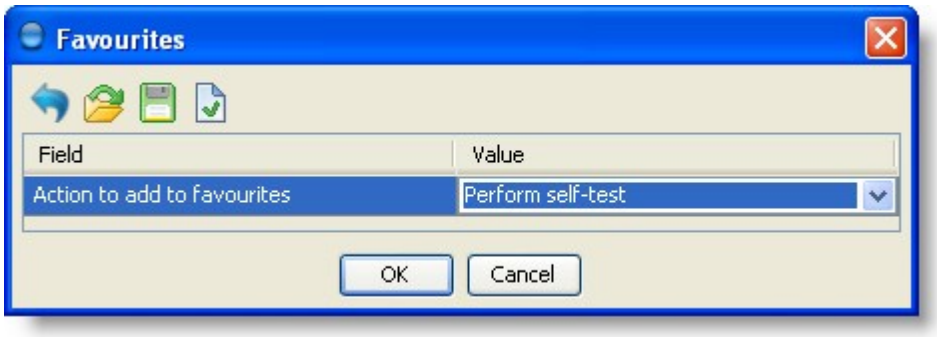
В этом разделе показано, как настроить операцию самотестирования так, чтобы она вызывалась из списка Избранное в AtomMind Client 359. Аналогичный метод можно использовать для выполнения любой другой операции с Вашим устройством.

Вот как может выглядеть действие Самотестирование в контекстном меню устройства:

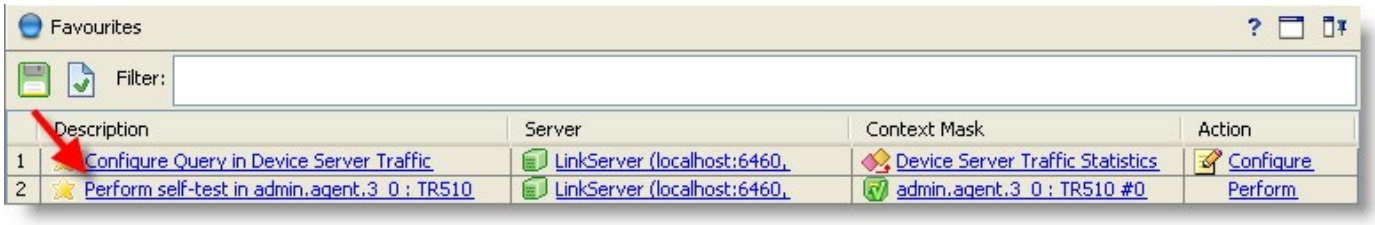


1. Помещаем действие Самотестирование в Избранное

Перетащите узел ( ) Системного дерева устройства в узел ( ) Избранное. Выберите действие Самотестирование из выпадающего списка и кликните OK:

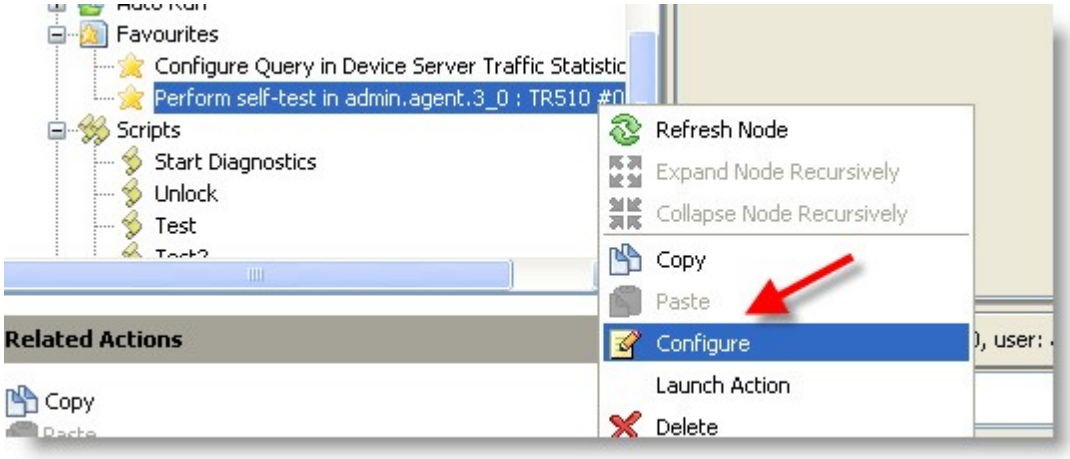


Только что созданный избранный элемент появится в таблице Избранное:



2. Редактирование действия Избранное для запуска самотестирования множества устройств

Расширьте узел Системного дерева Избранное и щелкните правой кнопкой мыши по новому элементу "Выполнить самотестирование". Выберите Настроить из контекстного меню:

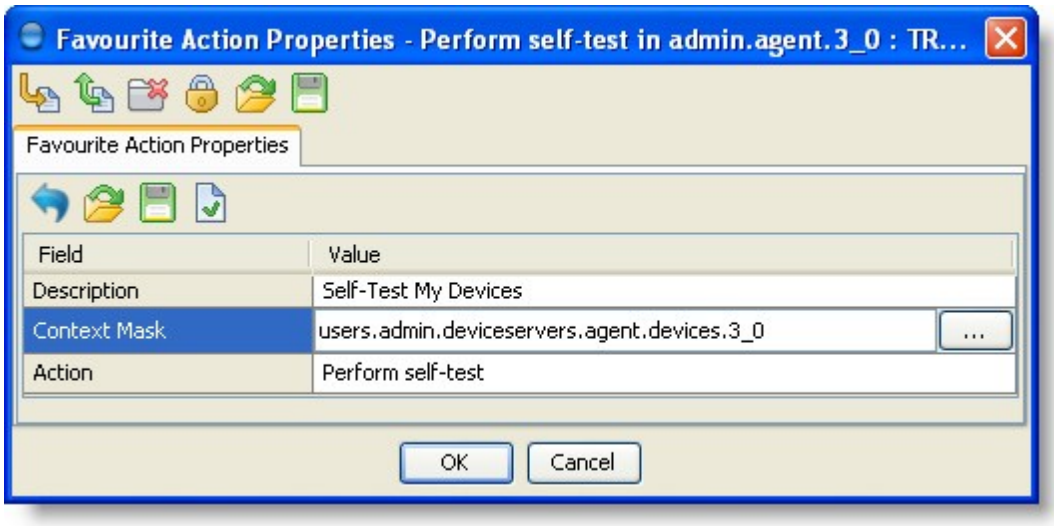


Смените **Описание** на новое значимое имя, например, **Провести самотестирование моих устройств**.

Затем измените **контекстную маску** так, чтобы она включала ряд контекстов[41] устройства, из которых будет вызываться действие Самотестирования. Смените **контекстную маску** на **users.*.deviceservers.*.devices.*** , чтобы заставить Избранное инициировать самотестирование у всех устройств системы.

Выполнять самотестирование будут лишь те устройства, которые доступны с Вашими пользовательскими правами доступа[477].

Чтобы провести самотестирование лишь Ваших собственных устройств, например, устройств, которые принадлежат Вашей пользовательской учетной записи[478], смените Контекстную маску на **users.*YOUR_USER_NAME*.deviceservers.*.devices.***.



3. Запустить самотестирование

Чтобы выполнить самотестирование, просто кликните в Избранном **Провести самотестирование моих устройств** в таблице Избранное. Это заставит AtomMind Server немедленно выполнить запрошенную процедуру. В зависимости от типа устройства, результат тестирования и результаты о неполадках будут отправлены на дисплей устройства, в журнал событий[398] или показаны пользователю при помощи процедуры Показать сообщение[97] /показать ошибку[95].

18.2 Копирование объектов между пользовательскими

В этом разделе объясняется, как администраторы AtomMind могут копировать объекты между различными пользовательскими учетными записями[478], используя AtomMind Client[359]. Такое копирование объектов может быть удобным следующих случаях:

- Когда Вы создали полезные объекты (например, тревоги[779], отчеты[928], виджеты[943] и пр.) под своей учетной записью и хотите поделиться ими с другими пользователями, не позволяя им модифицировать исходные объекты;

- Когда пользователь хочет использовать объект, созданный другим пользователем, но Вы не хотите дать ему права доступа к данному объекту.

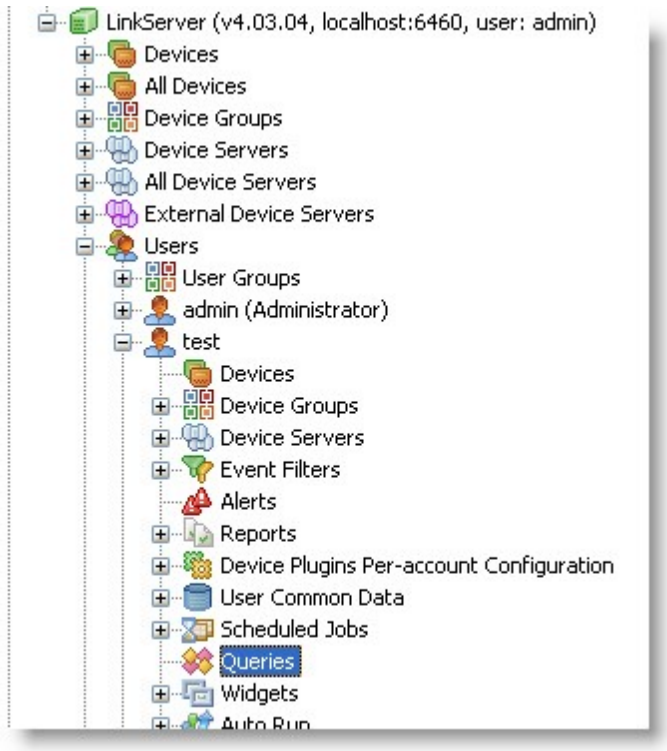
Если Вы хотите разрешить одному пользователю полный доступ, возможность изменять и использовать объект, который принадлежит другому пользователю, и не делать при этом копию этого объекта, см раздел Как разрешить одному пользователю иметь доступ к объектам другого пользователя 1640.

1. Находим узел контейнера

В системном дереве 370 разверните узел для пользовательской учетной записи ( ), под которой будет создана копия. Узлы пользовательской учетной записи находятся под узлом Пользователи **(Users)** ( ). Узел Пользователи доступен лишь в том случае, если у вас есть права доступа 477 администратора.

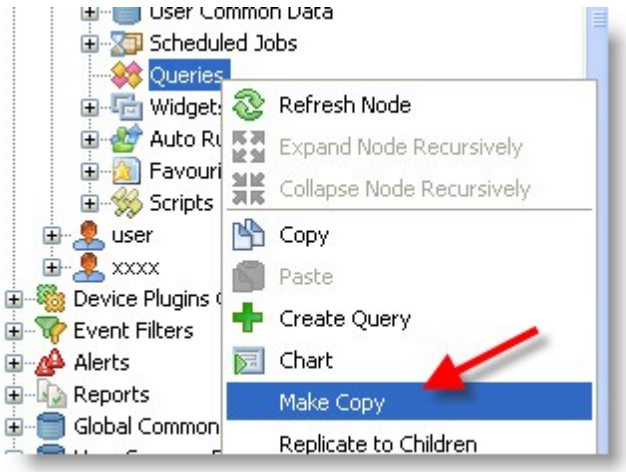
В рамках узла пользователя найдите узел контейнера, который содержит объекты такого же типа, что и скопированный объект. Например, если Вы собираетесь сделать копию скрипта 879, найдите узел **Скрипты** ( ).

Ниже приводится скриншот, который показывает, как узел **Запросы** для пользователя вызвал **тестирование**:



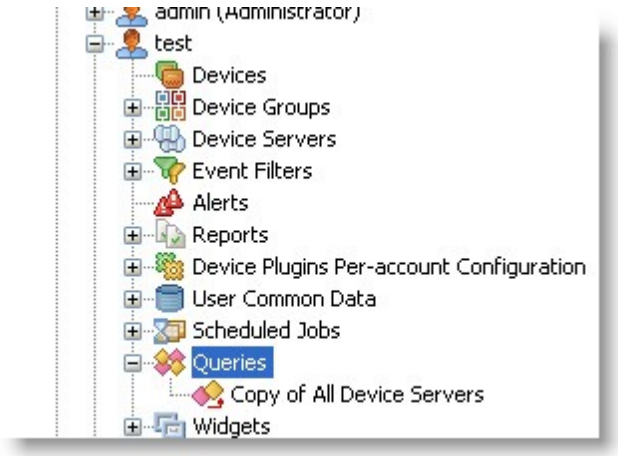
2. Запуск действия Сделать копию

Правой кнопкой мыши щелкните по узлу, найденному во время выполнения прошлого этапа и выберите **Сделать копию**:



В окне Селектора объектов 402 найдите объект, который нужно скопировать. Кликните по нему, а затем кликните OK. Например, если вы выполняете копию запроса ( ), раскройте узел **Пользователи**, а затем узел пользователя, которому принадлежит запрос, необходимый для копирования. А затем разверните узел **Запросов** данного пользователя ( ):

Копия выбранного запроса будет создана под учетной записью пользователя, выбранного на этапе 1:



Дополнительную информацию о копировании объектов см в разделе Выполнить действие Копировать 109.

Использование операции по перетаскиванию мышью для клонирования объектов

Еще один метод копирования объекта - это перетаскивание его мышью в соответствующий узел контейнера под учетной записью другого пользователя. Это быстрее, но может оказаться неудобным в некоторых случаях потому, что узлы могут располагаться далеко друг от друга в Системном дереве, и Вам придется одновременно удерживать мышью объект и прокручивать дерево до нужного места.

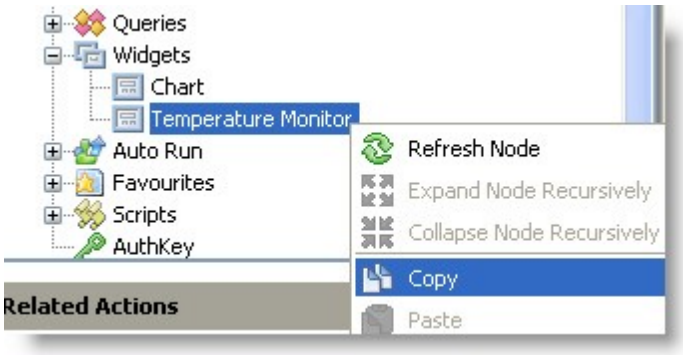
18.3 Предоставление разрешения пользователю к объ

В этом уроке объясняется, как предоставить пользователю 478 AtomMind права доступа для просмотра, изменения и использования объекта, который принадлежит другой пользовательской учетной записи. Этот объект может быть Отчетом 928, Фильтром событий 762, устройством, или любым другим системным ресурсом. Информацию о системе безопасности AtomMind см. в разделе Безопасность и права доступа 477.

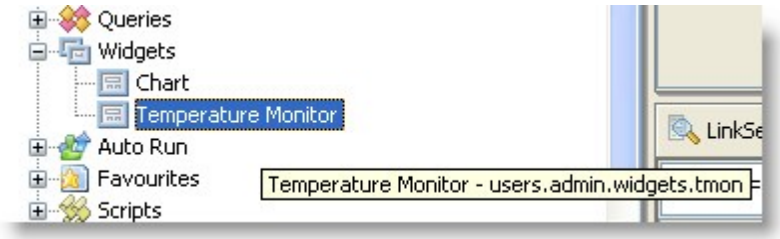
Давайте посмотрим, каким образом виджет 943 **Мониторинга температуры**, принадлежащий пользователю - **администратору**, может стать доступным и для пользователя **тест**.

1. Определение пути контекста совместно используемого объекта

Найдите объект, который будет совместно использоваться в системном дереве 370 AtomMind Clienta 359. Кликните по нему правой кнопкой мыши и выберите Копировать **(Copy)** в контекстном меню, чтобы скопировать его контекстный путь в область обмена данными:

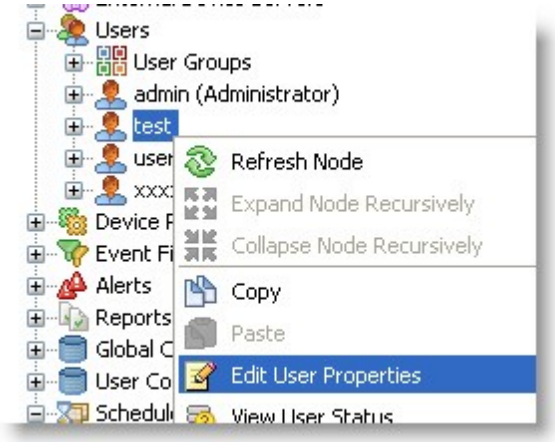


Можно также ввести путь вручную позднее, если это необходимо. Чтобы увидеть путь контекста любого объекта, наведите на него мышь, и Вы увидите подсказку. В нашем примере путь контекста виджета 1627 - **users.admin.widgets.tmon**:

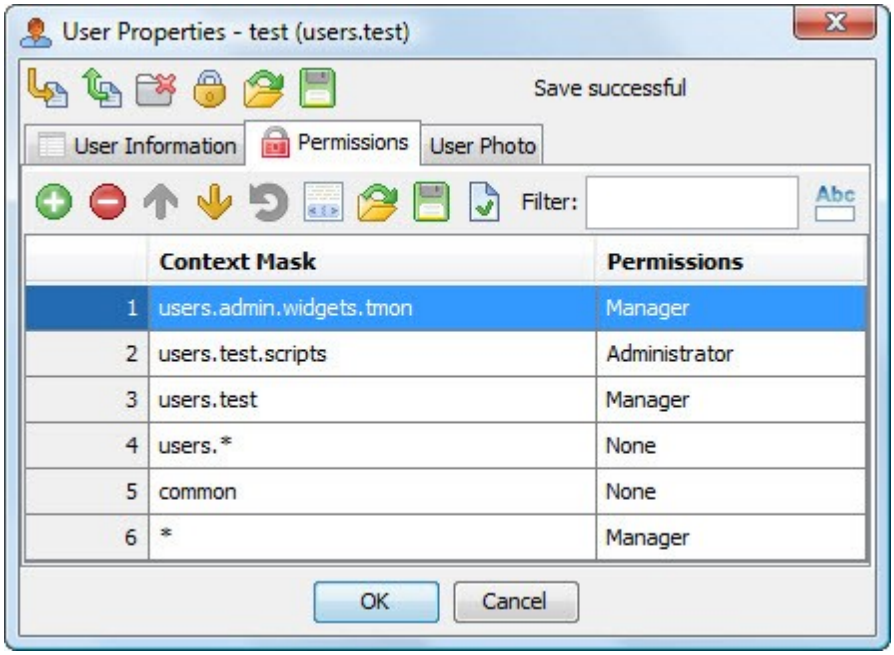


2. Изменение таблицы прав доступа

А теперь отредактируйте таблицу с правами доступа для пользователя, который должен иметь доступ к совместо используемому ресурсу. Найдите узел этого пользователя ( ) в Системном дереве (она находится под узлом Пользователи) и щелкните по нему правой кнопкой мыши. Выберите из контекстного меню **Редактировать свойства пользователя**:



Во вкладке **Права доступа** выберите первую строку в таблице, кликнув по ее номеру. Затем кликните по элементу **Добавить ряд** ( ), чтобы вставить новый ряд в начале таблицы. Кликните правой кнопкой мыши по полю **Контекстной маски** для новой записи и выберите из контекстного меню элемент **Вставить**. Можно также нажать Ctrl-V или вручную вставить путь контекста, обозначенный на этапе 1. А затем выберите элемент **Менеджер** в поле **Права доступа**:



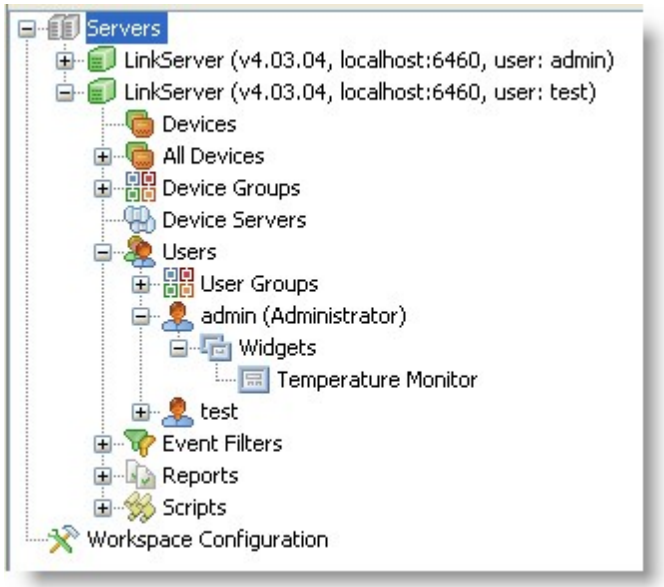
В начале таблицы мы поместили запись о новых правах доступа. Убедитесь, что она не будет перезаписана другими записями.

Для пользователей контекстными масками 44 возможно совместное использование ряда ресурсов. Например, чтобы разрешить доступ ко **всем** виджетам, используйте контекстную маску **users.admin.widgets.*** вместо **users.admin.widgets.tmon.**

Чтобы предоставить **все только что созданные пользовательские права** на определенные ресурсы, добавьте запись прав доступа, аналогичную описанной ранее, когда речь шла о таблице глобальной конфигурации - Дополнительные права доступа для новых пользователей 202.

3. Доступ к совместно используемым ресурсам

Зарегистрируйтесь как пользователь, который уполномочен иметь доступ к совместно используемому ресурсу. Теперь у Вас должна быть возможность просматривать ресурс через системное дерево. Оно находится под узлом его владельца:



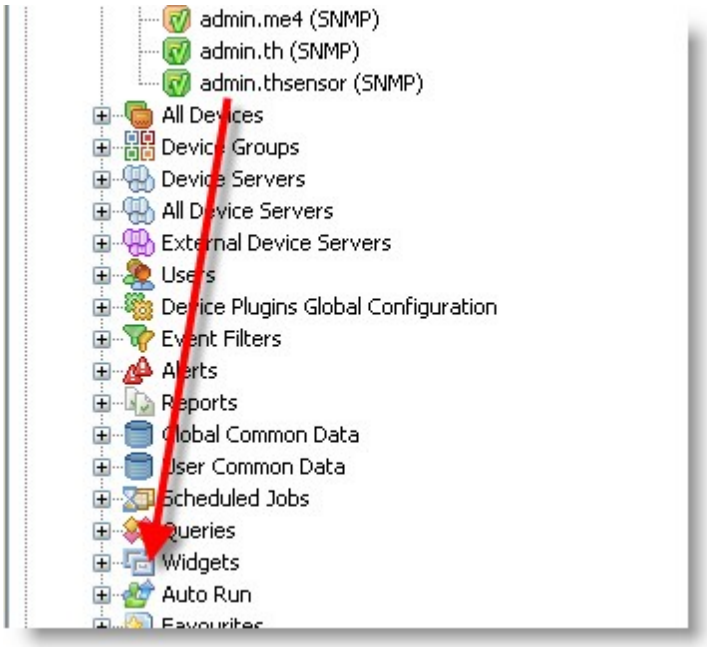
18.4 Создание виджета для мониторинга устройства

В этом уроке рассказывается о процессе создания простого виджета[943] с целью осуществления мониторинга устройства. В данном случае мы предполагаем, что устройство - это цифровой датчик температуры/влажности, который связан с AtomMind Server по протоколу SNMP. Наш виджет будет показывать текущую температуру или влажность в режиме реального времени.

Этот раздел не содержит информацию о связи сенсора с сервером и его конфигурации, мы исходим из того, что сенсор уже находится в режиме online, а система распознает/видит его как устройство.

1. Запустите клиент

Перед тем как приступить к созданию виджета, найдите узел устройства в Системном дереве[370] ( ). Перетащите его на узел Виджеты ( ):

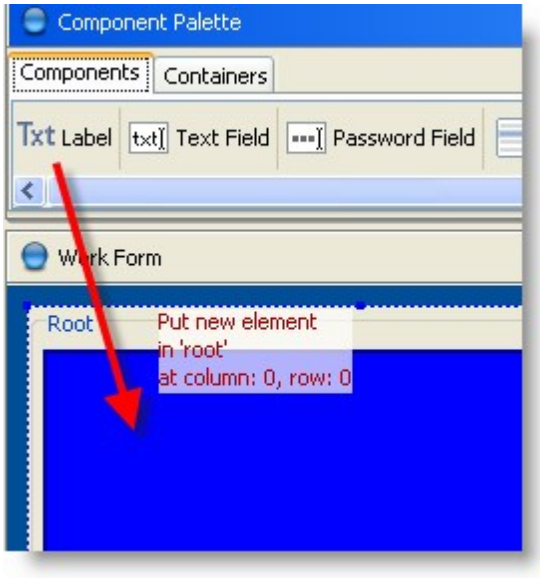


Эта операция отобразит **Свойства**: окно **Виджета**. Установите **имя Виджета** как `thmonitor`, а **Описание виджета** как `Temperature/Humidity Monitor`. Если кликнуть OK, откроется редактор виджетов[423] с пустым шаблоном виджета[946]. Если перетащить устройство на узел Виджеты, только что созданный виджет будет доступен для каждого контекста устройства.

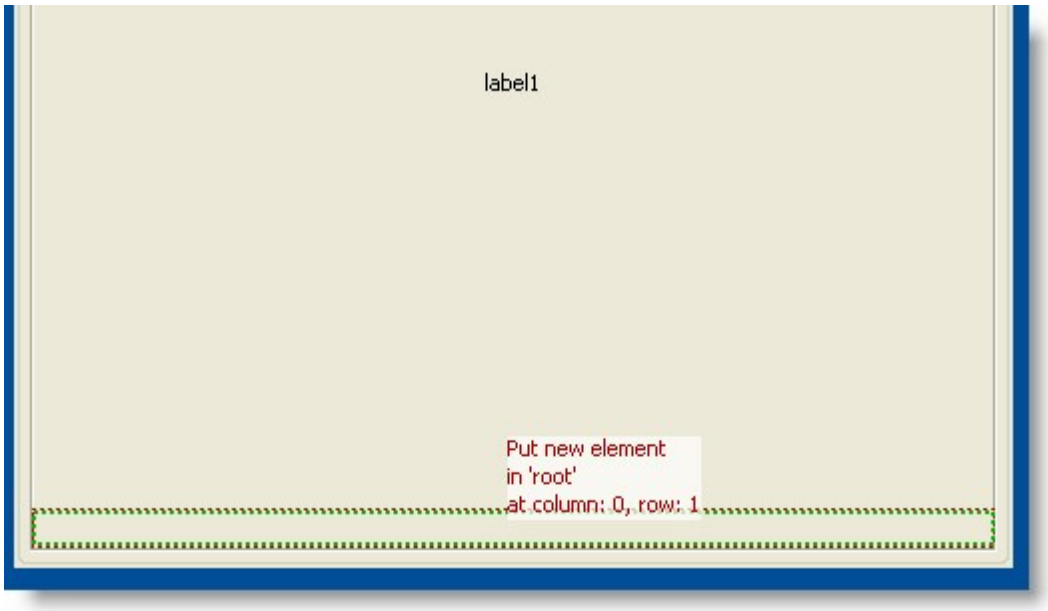
2. Добавление компонентов виджета

Давайте добавим некоторые компоненты в виджет. О том, как обращаться с компонентами виджета, можно узнать из раздела Редактирование разметки виджета[432].

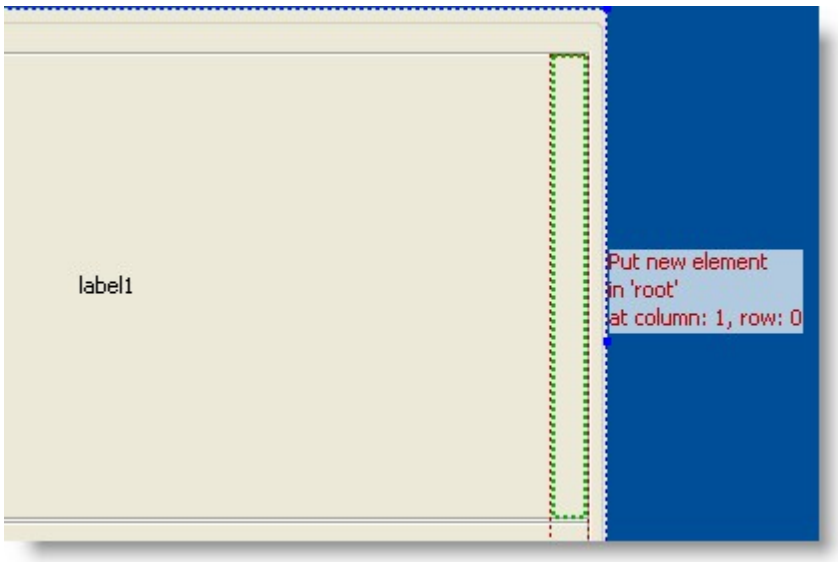
Перетащите компонент **Надпись** ( ) из палитры компонентов[430] в корневую панель виджета на рабочей форме[426]:



Добавьте еще одну надпись, перетащив его мышью под текущий в колонке 0, ряду 1 корневой панели:



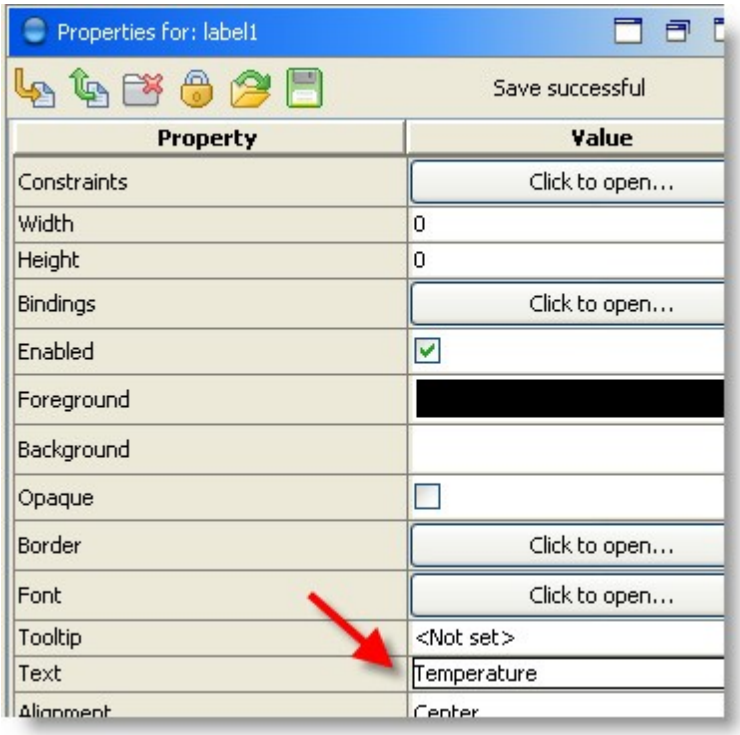
Добавьте еще одну надпись в колонку 1, ряд 0:



Затем добавьте компонент **Регулятор** ( ) в колонку 1, ряд 1:

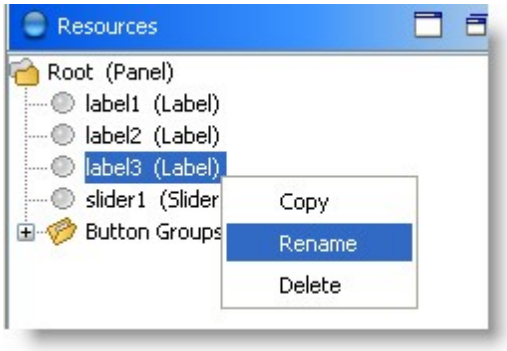
3. Редактирование свойств компонента

Кликните по **надписи 1** в рабочей форме и измените его свойство **Текст** в окне свойств на "`Temperature: `".



Измените текст **Надписи 2** на "`Humidity: `". Затем измените текст **Надписи 3** на `0`.

Затем правой кнопкой мыши кликните по **Надписи 3** в Окне ресурсов 428 и выберите **Переименовать** из контекстного меню. Задайте имя `temperature`.



Переименуйте **регулятор 1** на `humidity` аналогичным образом.

А теперь откройте свойство **Шрифт ярлыка 1** в окне Свойства. Проверьте **Использовать особый шрифт**, установите **имя** на **Arial**, **размер** на **20** отметьте **Полужирный шрифт**.

Установите тот же шрифт для **надписи2 (label2) и температуру** (бывшая **label3**).

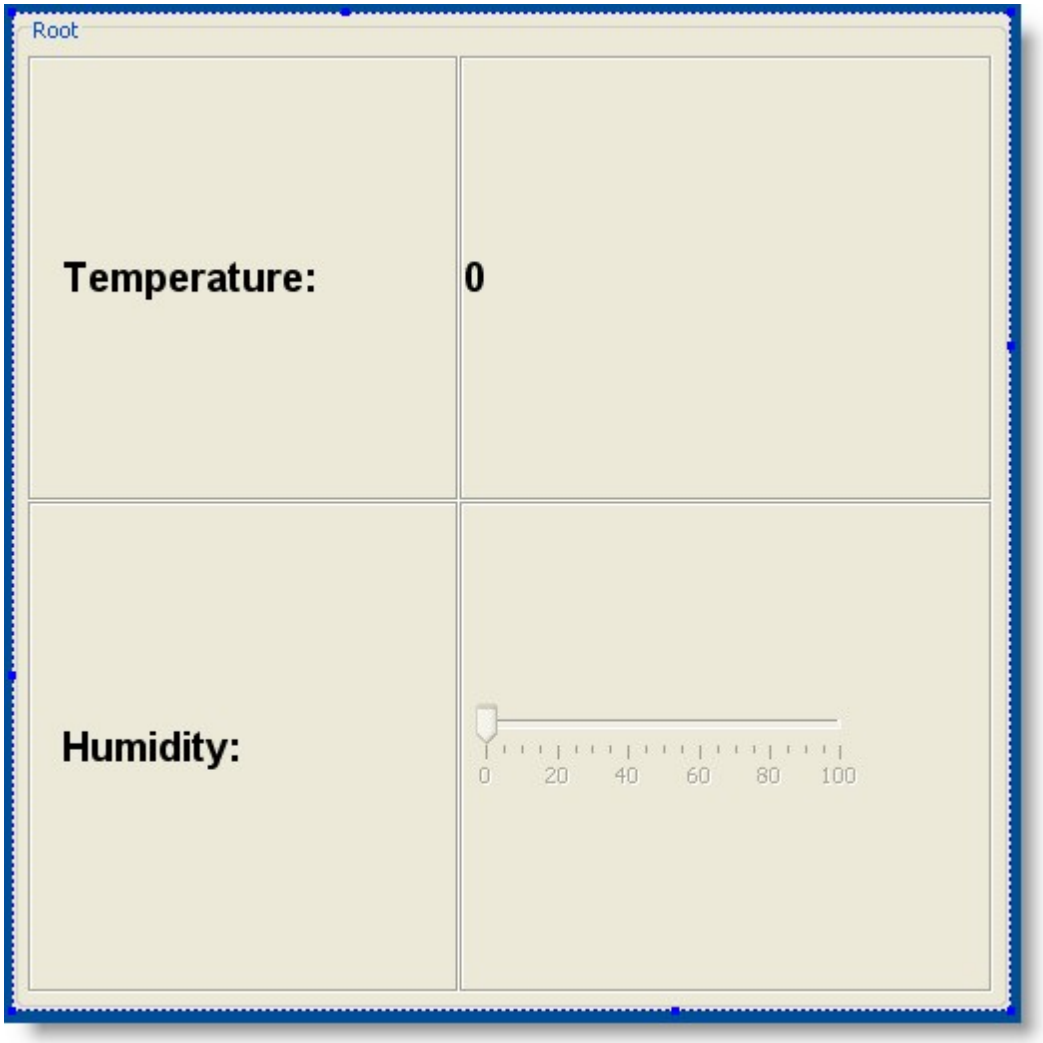
Установите **Включенное** свойство регулятора на **false** (неправильно), чтобы не допустить его перемещения.

Выберите все четыре компонента, один за другим, и установите их точку привязки 951 слева, щелкнув в инструментальной панели по кнопке .

Затем отредактируйте ограничения 950 надписей **температуры** и **влажности** и установите Левое поле на 15.

Вы также можете редактировать поля при помощи мышки, см. раздел Редактирование разметки виджета 432.

Теперь Ваш виджет должен выглядеть следующим образом:

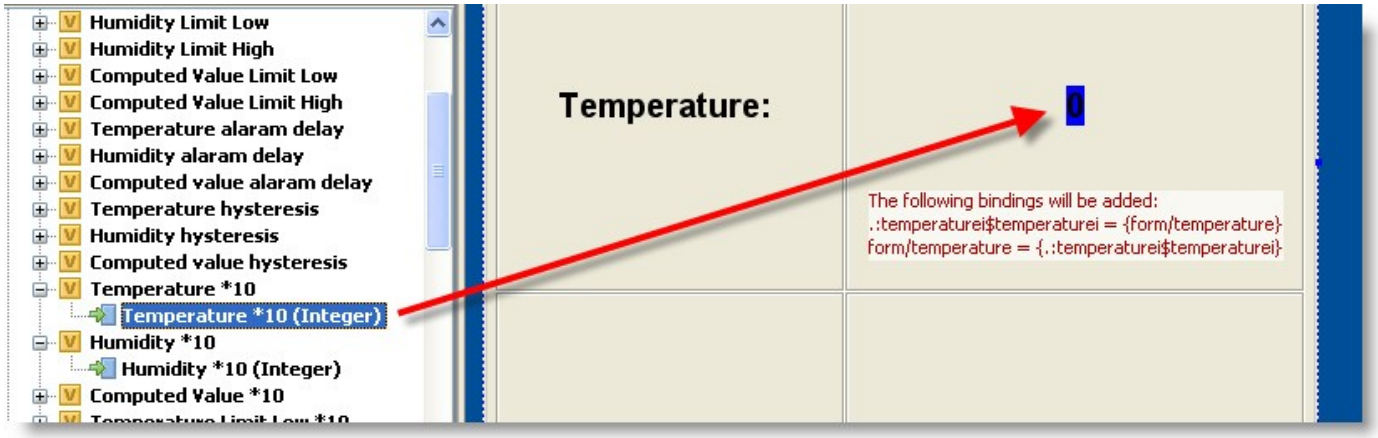


4. Создание привязок

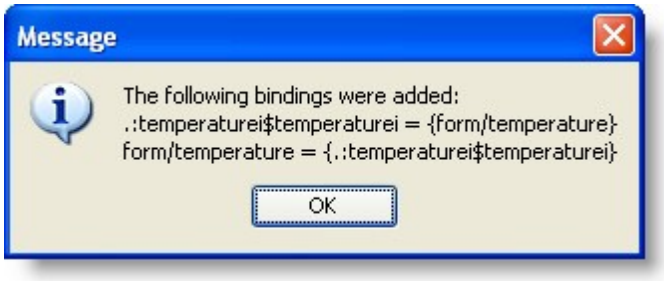
На этом этапе вы будете заниматься привязкой 1295 компонентов виджета к реальным данным. Эти данные представлены в форме переменных 61 контекста. У нашего сенсора есть две доступные переменные:

- **Temperature*10** , которая содержит значение температуры, выраженное в градусах Цельсия и умноженное на 10 до целого числа (т.е. значение 234 соответствует 23.4 градусам Цельсия);
- **Humidity*10** , которая содержит значение относительной влажности в процентах, умноженное на 10 до целого числа (т.е. значение 456 соответствует относительной влажности 45.6%).

Давайте сначала привяжем поле переменной **Temperature*10** к надписи температуры. Найдите эту переменную в окне селектора объектов 430 под контекстом устройства по умолчанию (должно быть выделено жирным шрифтом). Разверните его, чтобы увидеть поле **Temperature*10**. Перетащите это поле на надпись температуры:



Вы увидите сообщение о том, какие привязки были созданы:



Аналогичным образом привяжите поле **Humidity*10** переменной **Humidity*10** к полосе прокрутки при помощи мыши.

РЕДАКТИРОВАНИЕ ПРИВЯЗОК ВРУЧНУЮ

А теперь нам необходимо вручную отредактировать созданные привязки. Выберите **Корневую панель**, кликнув по Корню в верхнем левом углу рабочей формы и откройте его свойство **Все привязки** в окне свойств.

В таблице должно быть четыре привязки:

	Target	Expression	Activator	On Startup	On Event	Periodically	Evaluation Period (seconds)
1	.:temperaturei$temperaturei	{form/temperature}	form/root#submit	☐	☑	☐	0
2	form/temperature	{.:temperaturei$temperaturei}	form/root#reset	☑	☑	☐	0
3	.:humidityi$humidityi	{form/humidity}	form/root#submit	☐	☑	☐	0
4	form/humidityi	{.:humidityi$humidityi}	form/root#reset	☑	☑	☐	0

Value: .:humidityi$humidityi

Привязки 1 и 3 следует удалить, поскольку они будут пытаться записывать новые значения температуры и влажности в устройство. Эта операция не разрешена, и сервер будет ее блокировать. Удалите эти привязки, кликнув по их числам, а затем кликните **Удалить ряд** ( ) в инструментальной панели Редактора таблиц данных 382.

А теперь мы должны исправить значения температуры и влажности, которые были умножены на 10. Отредактируйте поле выражений в оставшихся привязках, добавив в них **/ 10**:

- `{.:temperaturei$temperaturei}` => `{.:temperaturei$temperaturei} / 10`
- `{.:humidityi$humidityi}` => `{.:humidityi$humidityi} / 10`

Кроме того, необходимо удалить активаторы 1296 оставшихся привязок так, чтобы они обрабатывались каждый раз, когда происходит изменение показателя температуры или влажности. Установите для активаторов пустые строки.

Таблица с получившимися в итоге привязками:

	Target	Expression	Activator	On Startup	On Event	Periodically	Evaluation Period (seconds)
1	form/temperature	{.:temperaturei$temperaturei} / 10		☑	☑	☐	0
2	form/humidity	{.:humidityi$humidityi} / 10		☑	☑	☐	0

Кликните OK, чтобы сохранить таблицу.

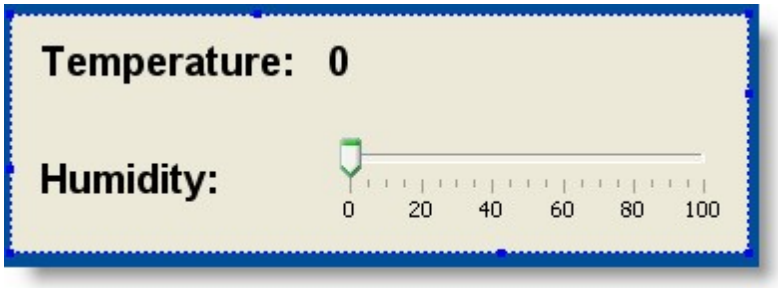
5. Изменение размера шаблона виджета

Кликните по элементу **Переключить оформление** ( ) в инструментальной панели 424. Виджет в рабочей форме станет похож на то, как он выглядел во время обычной операции.

Выберите корневую панель, кликнув по **Корню** в верхнем левом углу рабочей формы. Установите **ширину Окна** на 370 , а **высоту Окна** приложения на 120.

Можно также изменить размер виджета и его компонентов при помощи мыши. См. Редактировать разметку виджета 432.

В итоге, шаблон будет выглядеть следующим образом:



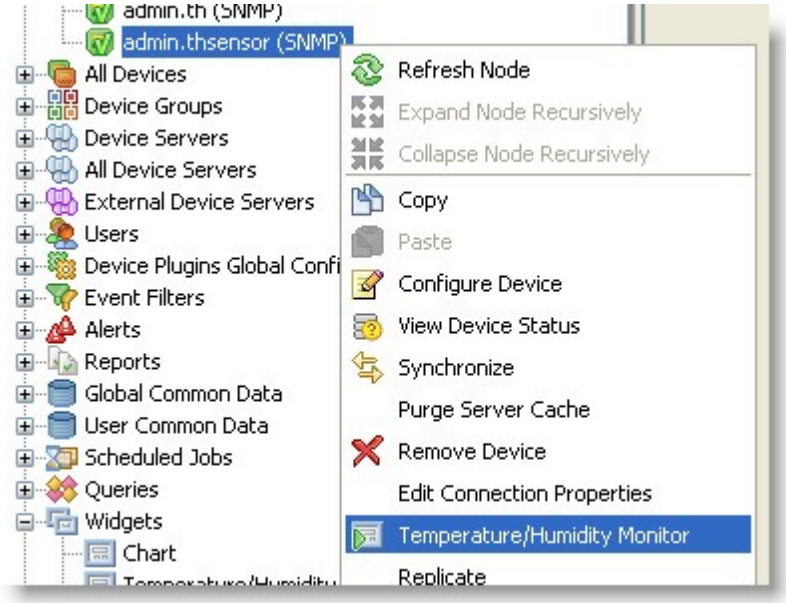
6. Сохранение шаблона виджета

Шаблон готов. Кликните **Done (**готово**)** ( ).

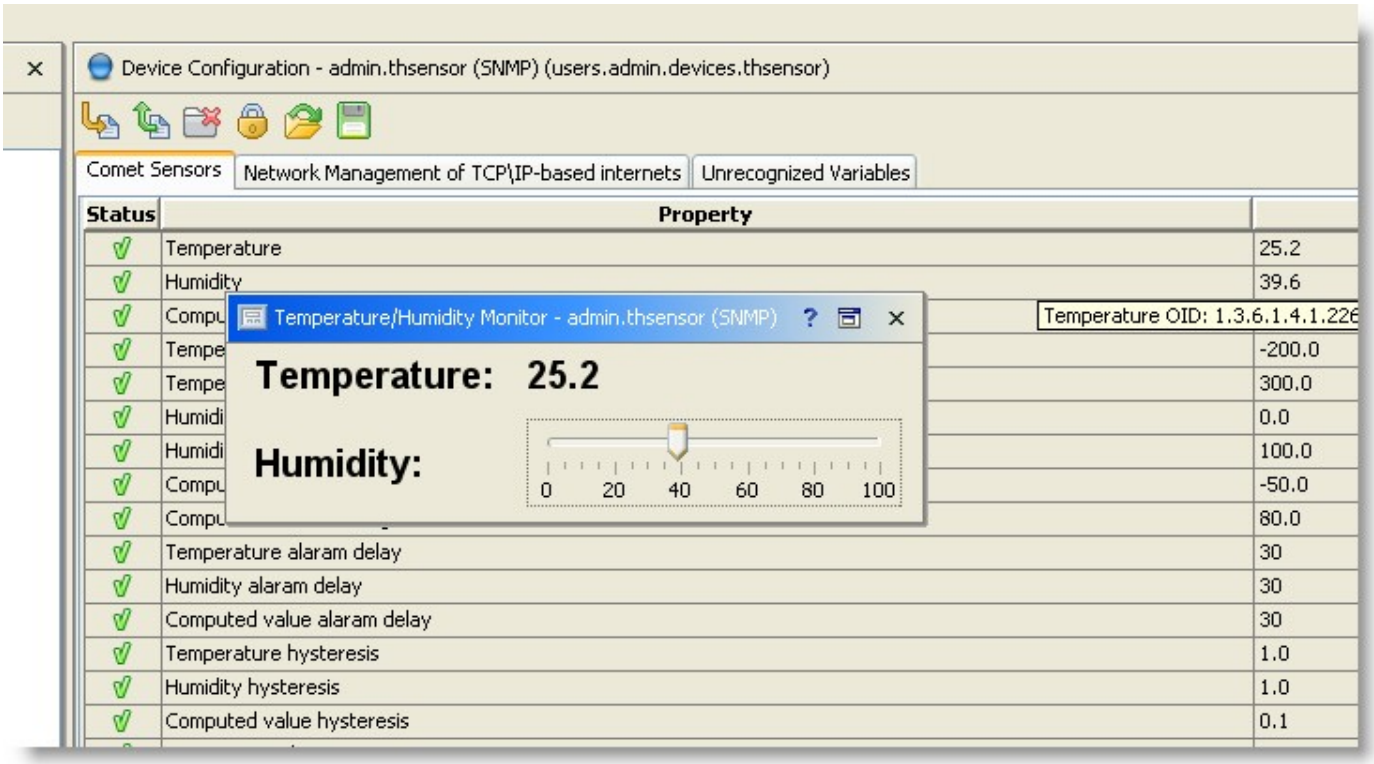
Теперь должен быть создан новый контекст виджета 1627 ( ), который появится в Системном дереве 370:

7. Запуск виджета

Пора увидеть наш новый виджет в действии. Щелкните правой кнопкой мыши по любому узлу Системного дерева для устройства ( ). Вы должны увидеть в контекстом меню новое действие Запустить виджет 948:



Выберите действие Запустить виджет для определенного сенсора температуры/влажности. В AtomMind Client появится новое содержащее виджет плавающее окно 366:



Вот и все! Новый виджет действует, а Вы теперь можете отслеживать текущую температуру и влажность. Информацию о том, как автоматически запустить виджет при запуске AtomMind Client см в разделе Как создать информационную панель для мониторинга устройства в реальном времени 1660.

18.5 Управление SVG-рисунками

AtomMind предлагает универсальную поддержку для SVG-рисунков (Scalable Vector Graphics - масштабируемая векторная графика). Она включает такие свойства, как прозрачность, произвольные геометрические тела, эффекты фильтров (тени, подсветка и пр.) и анимация. Любой SVG-рисунок можно масштабировать, повернуть, отразить зеркально без утраты его качества. Специальная библиотека 1974 включает в себя огромную коллекцию SVG-рисунков с дополнительной встроенной функциональностью. Эти специальные рисунки имеют динамические свойства для изменения различных визуальных аспектов: цвета, толщины линии, анимации и пр.

В этом уроке показано, как управлять SVG-рисунками и использовать их для создания интерактивных пользовательских интерфейсов. Здесь Вы научитесь создавать виджет с двумя SVG-компонентами: вентилятором, контролируемым кнопкой.

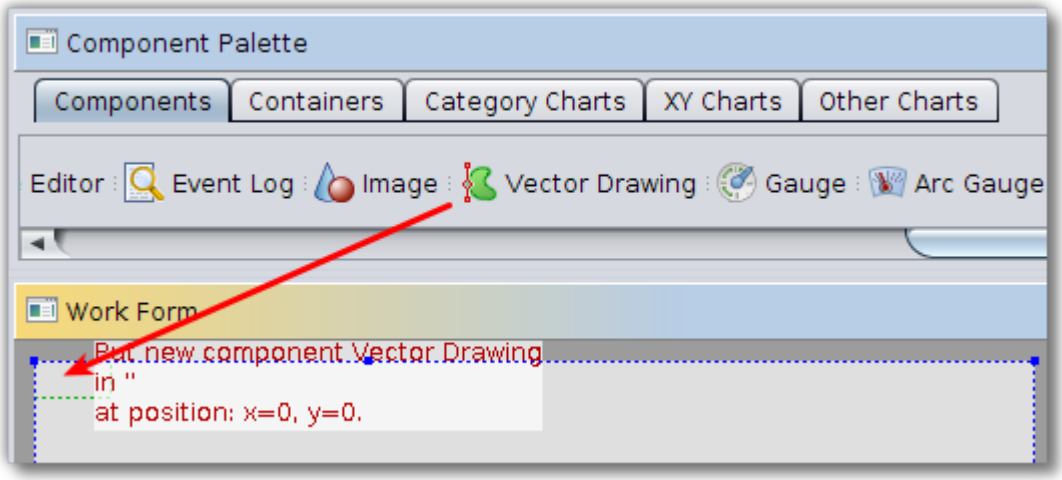
1.Создание виджета

Чтобы создать пустой виджет, дважды кликните по узлу **Виджеты** ( ) в системном дереве 370 или выберите во всплывающем меню действие Создать **виджет** ( ). В открывшемся окне установите свойство **Имя виджета** - `svgManipulation` и **Описание виджета** - `SVG Manipulation`, кликните OK. Это переместит Вас к редактору виджетов 423 с пустым шаблоном виджета 946.

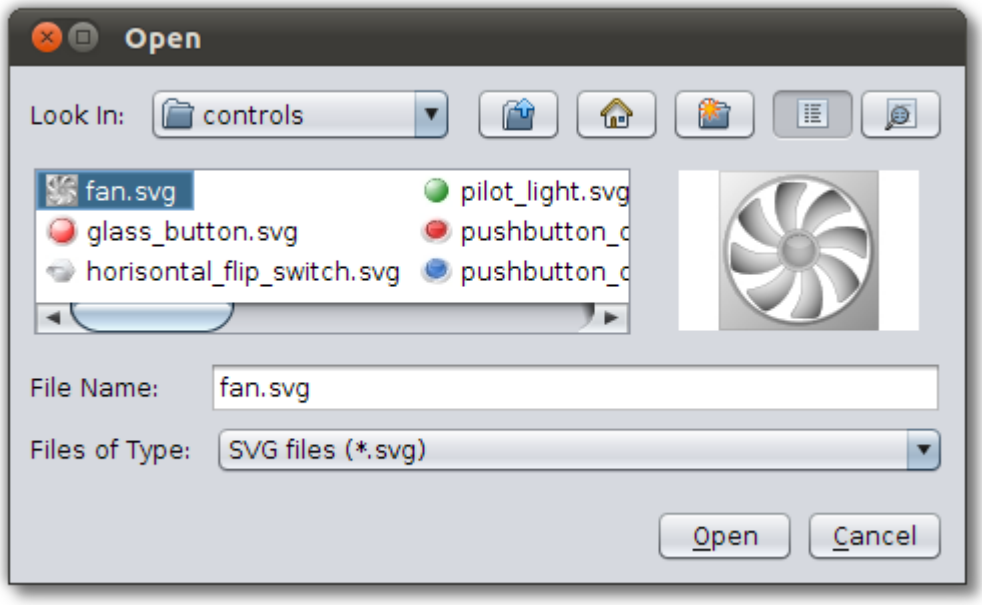
2. Добавление SVG-рисунков

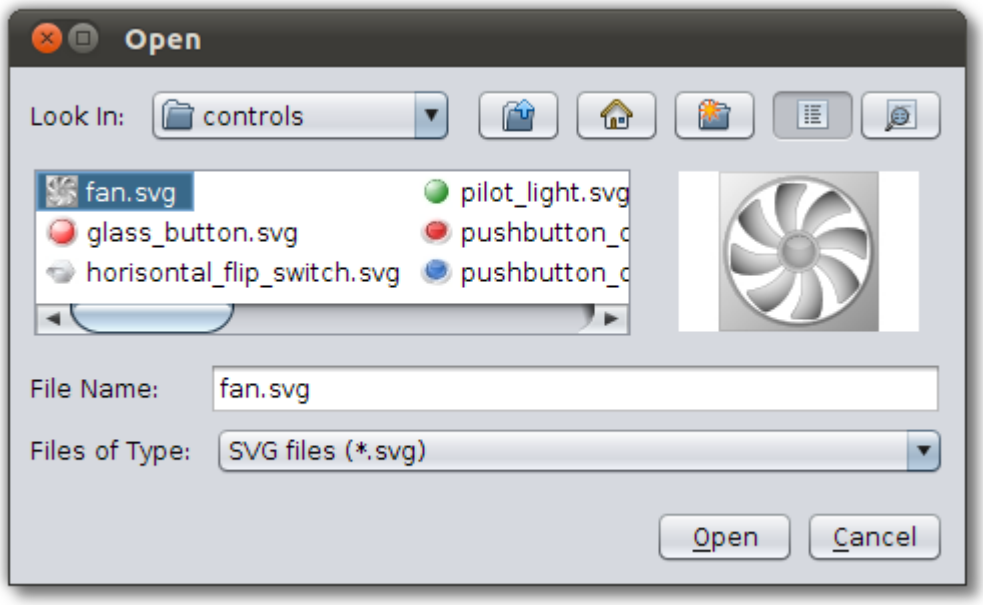
Во-первых, установите свойство **компоновки** Корневой панели 1042 на абсолютную 954 - так Вы не будете ограничены в разметке компонентов. Мы используем корневую панель для простоты, но Вы можете также использовать в качестве контейнера и отдельную панель.

Перетащите компонент **Векторный рисунок** ( )из палитры компонентов 430 в корневую панель рабочей формы 426 виджета:

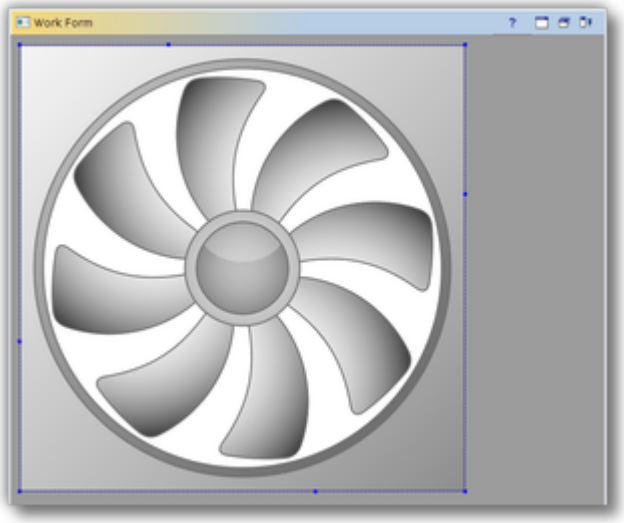


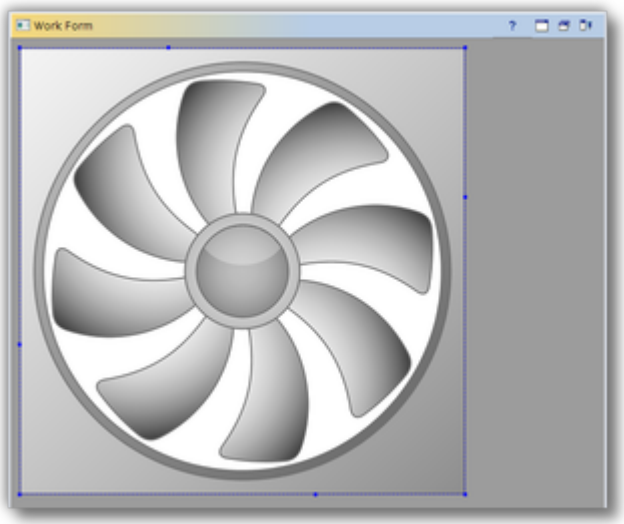
Теперь загрузите файл с изображением вентилятора из библиотеки 1974: нажмите кнопку **Выбрать (Choose)** в свойстве файла **SVG**, а затем найдите и откройте нужный файл в появившемся диалоговом окне.



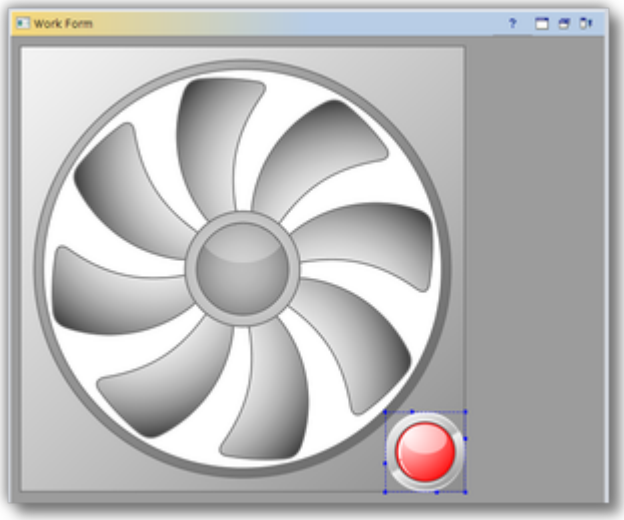


Установите положение компонента и его размер: **координата X** и **координата Y** равны 0, **ширина** и **высота** - 500. Таким образом, вентилятор будет занимать все пространство виджета.



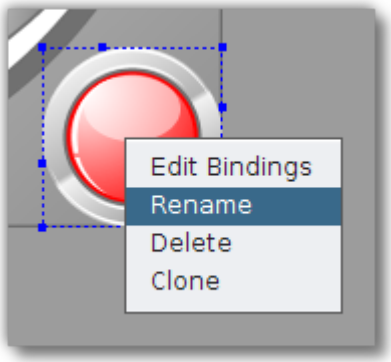


Перетащите еще один компонент **Векторного рисунка** ( )из палитры компонентов 430 в рабочую форму 426. Затем загрузите фал изображения кнопки из библиотеки 1974, как описано ранее. Измените размер и поместите его в левый нижний угол: установите значение **координаты X** и **координаты Y** равным 410, свойства **ширины** и **высоты** - 90. А теперь перед нами вентилятор с кнопкой:



3. Переименование компонентов

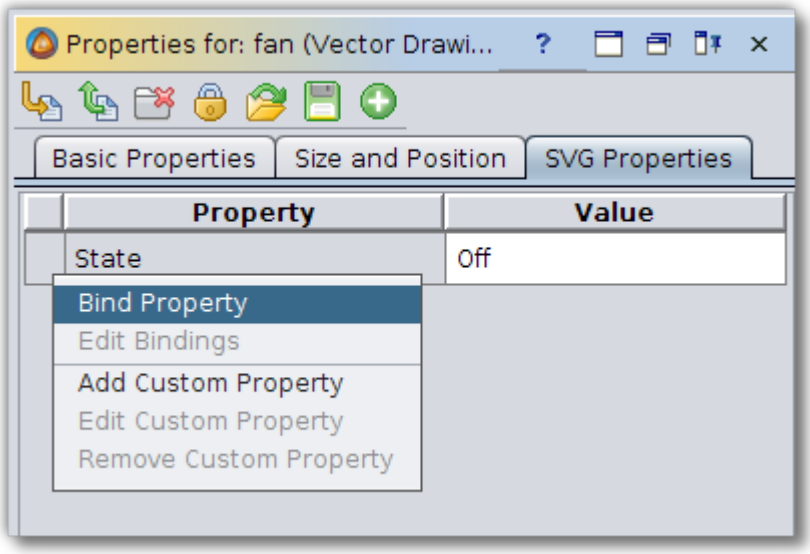
Давайте для простоты идентификации переименуем наши компоненты. Правой кнопкой мыши кликните по компоненту (или по имени компонента в Дереве ресурсов 428) и выберите в появившемся меню элемент **Переименовать**.



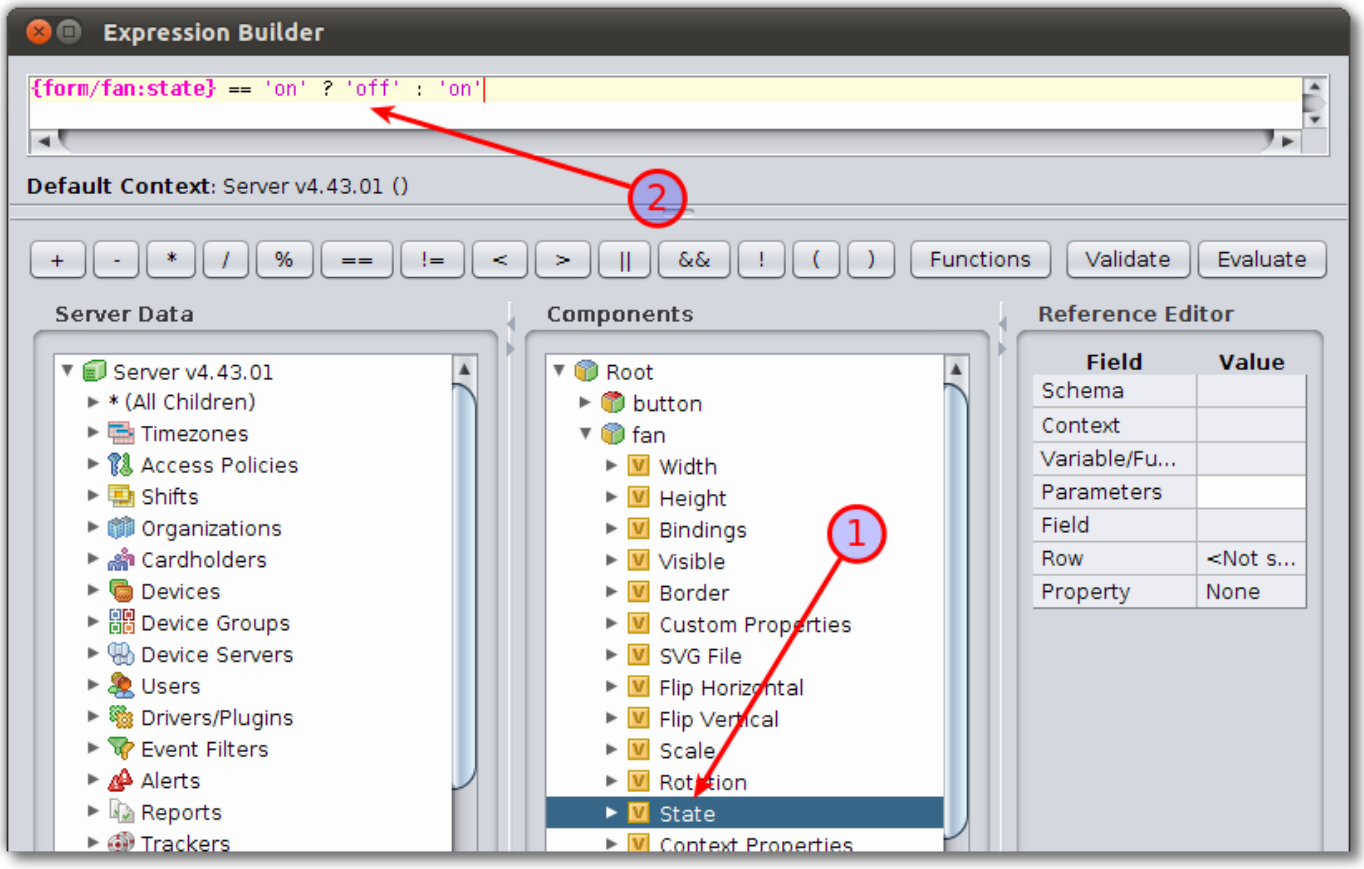
Присвойте имена компонентам `fan` и `button`.

4. Анимация вентилятора

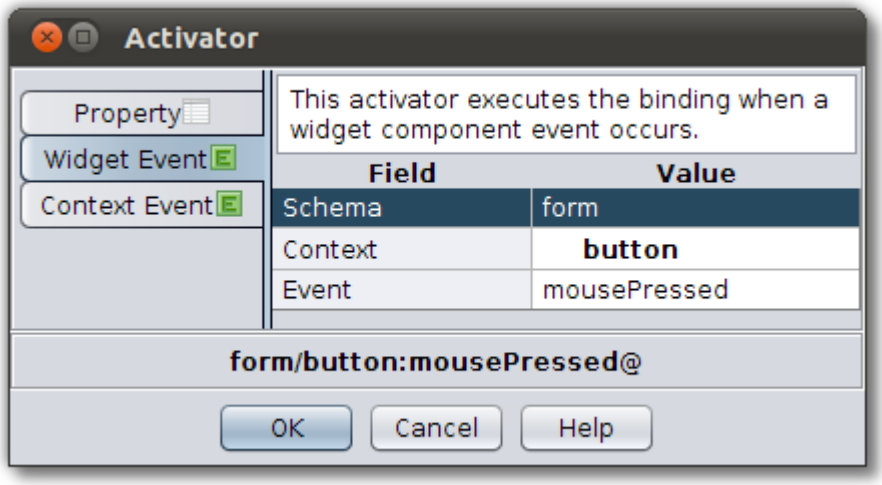
Сначала заставьте кнопку включать вентилятор. SVG-изображение вентилятора имеет свойство **Состояние**, которое активирует и деактивирует анимацию вращения. Давайте создадим привязку 1295 для этого свойства так, чтобы щелчок по кнопке отвечал за переключение состояния анимации вентилятора. Чтобы это выполнить, щелкните правой кнопкой мыши в свойстве вентилятора **Состояние** в редакторе свойств 431 и выберите элемент во всплывающем меню **Привязать свойство**. Должно появиться диалоговое окно Привязать свойство.



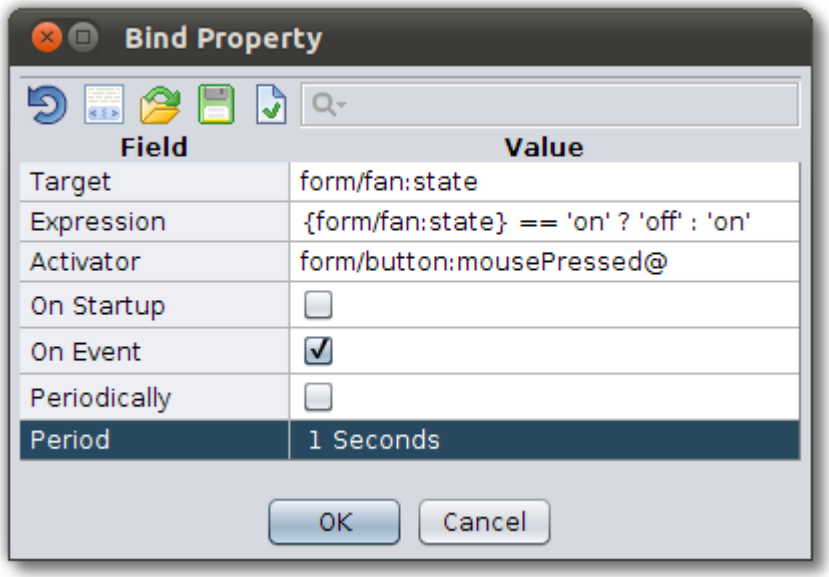
Свойство Цель[1295] уже установлено на `form/fan:state`. Кликните по кнопке **[...]** внутри поля Выражение[1296], чтобы отредактировать текст выражения интерактивно в Конструкторе выражений[404]. Во-первых, найдите свойство **Состояние** вентилятора в дереве **Компонентов** редактора выражений, дважды кликните по нему, чтобы вставить ссылку. Во-вторых, добавьте `== 'on' ? 'off' : 'on'` к выражению, кликните OK, чтобы разрешить диалоговое окно **Конструктора выражений**. Итоговое выражение вернет инверсию значения свойства **Состояния** вентилятора после его вычисления.



Затем переключитесь в поле Активатор[1296] и кликните **[...],** чтобы открыть диалоговое окно **Активатор**. Выберите вкладку **Событие виджета**, а затем установите поле **Контекст** как `button` , а **Событие** как `mousePressed`. Кликните OK для подтверждения. Этот активатор будет запускать привязку при каждом клике по кнопке.

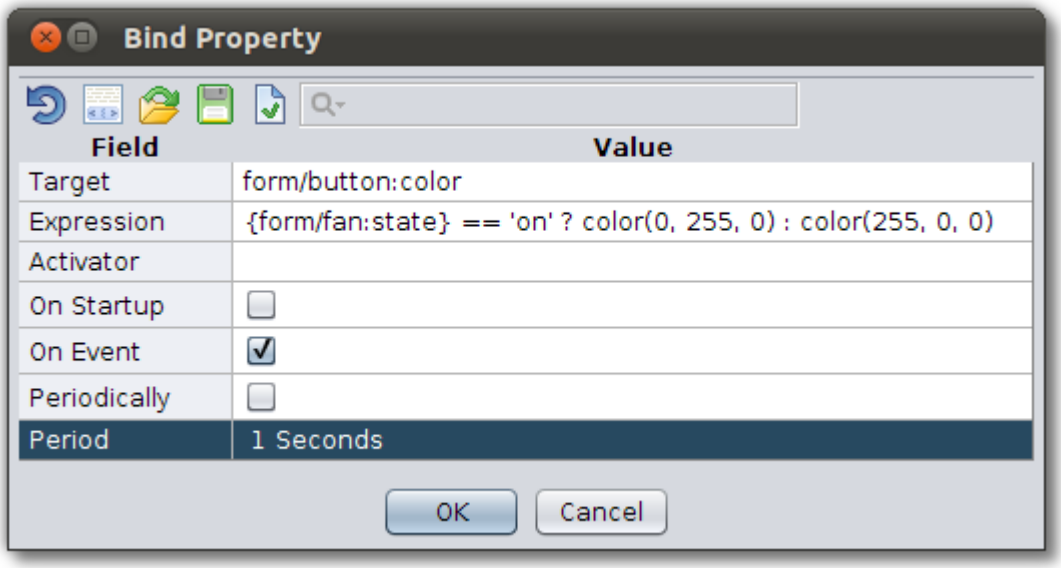


Убедитесь, что свойство **Событие** отмечено в диалоговом окне **Привязать свойство**. Удостоверьтесь, что все свойства имеют такие же значения, что и на скриншоте, и кликните OK:

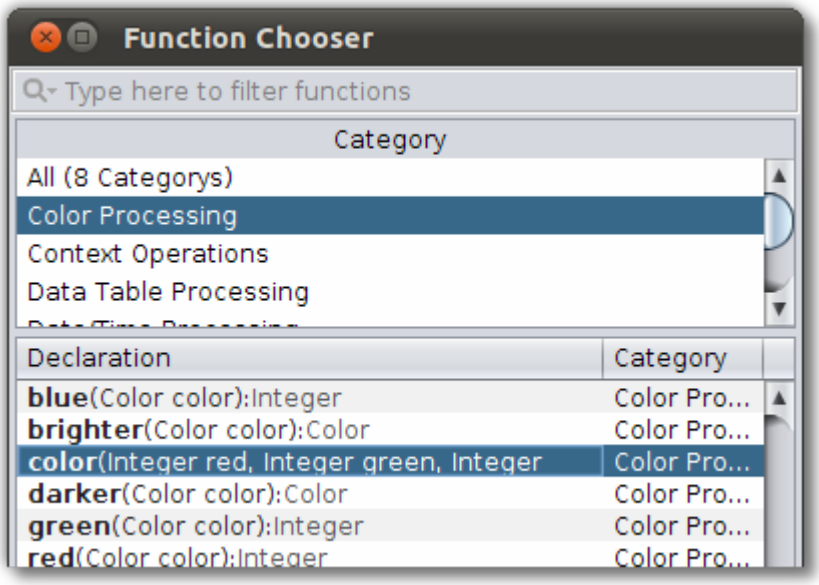


5. Анимация кнопки

А теперь давайте разрешим кнопке загораться зеленым светом, когда вентилятор включен. Правой кнопкой мыши щелкните по свойству кнопки **Цвет** в Редакторе свойств[431] и выберите во всплывающем меню элемент **Привязать свойство**. Установите свойства привязки аналогичным образом, как описано ранее, так, чтобы все значения соответствовали значениям на скриншоте (см ниже) и кликните OK:

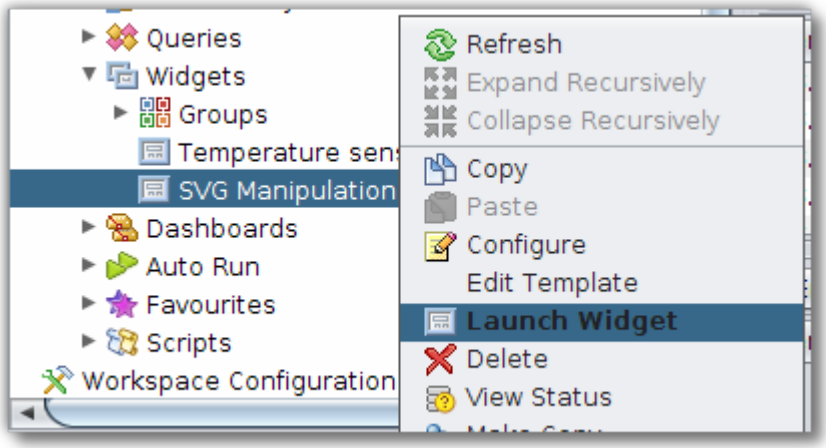


Выражение включает функцию **цвета**, которая возвращает цвет с назначенными значениями красного, зеленого и синего цветов. Чтобы вставить функции в выражение, можно использовать диалоговое окно **Выборщик функций**, которое открывается кнопкой **Функции** в диалоговом окне **Конструктора выражений**:



6. Сохранение и запуск виджета

Кликните **Готово** ( ), чтобы завершить редактирование и сохранить виджет. В Системном дереве 370 должен появиться контекст виджета 162 ( ) по **управлению SVG**. Чтобы увидеть виджет в действии, выберите во всплывающем меню действие **Запустить виджет** ( ).



А теперь можно нажать на кнопку и увидеть, как работает вентилятор.

В этом уроке мы показали, как создавать интерактивные компоненты. А теперь Вы можете сочетать элементы управления и использовать их для построения собственных комплексных интерфейсов.

18.6 Управление событиями компонентов

Когда происходит событие, у него появляются данные, специфичные для этого типа события. Например, данные о событии **Клик мышкой** содержат координаты X и Y позиции мыши во время клика. Этот раздел объясняет, как использовать эти данные в привязках виджета. Мы создадим виджет с изображением, которое передвигается на последнюю позицию клика мыши.

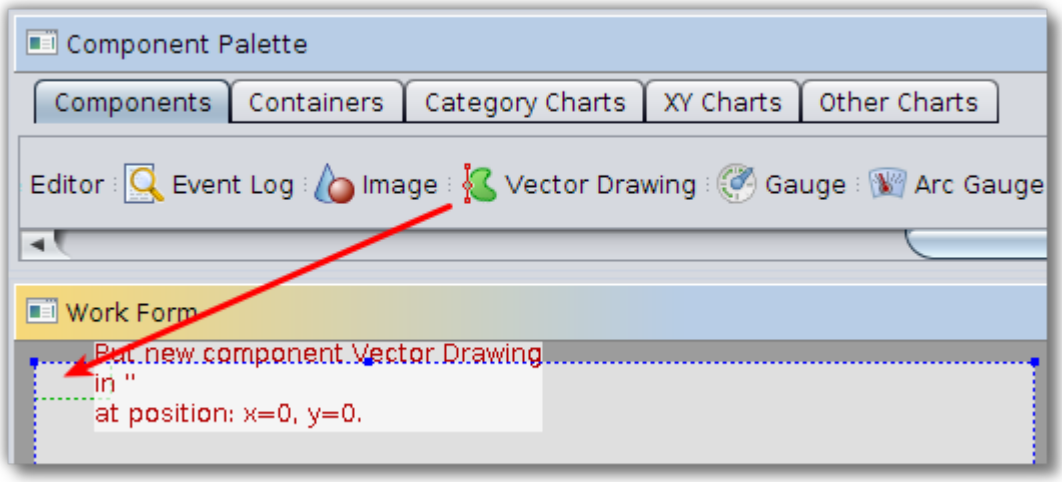
1. Создание виджета

Для создания простого виджета дважды щелкните на узле **Виджеты** ( ) системного дерева 370 или выберите его действие **Создать виджет** ( ) во всплывающем меню. В открытом окне задайте свойство **Имя виджета** на `componentEvents`, а **Описание виджета** на `Handling Component Events`, нажмите OK. Вы попадете в Редактор виджетов 423 с пустым шаблоном виджета 946.

2. Добавление изображения

Во-первых, установите свойство **Разметка** корневой панели 1042 на абсолютную разметку 954, чтобы можно было свободно позиционировать компоненты. Можно использовать любой другой контейнер вместо Корневой панели.

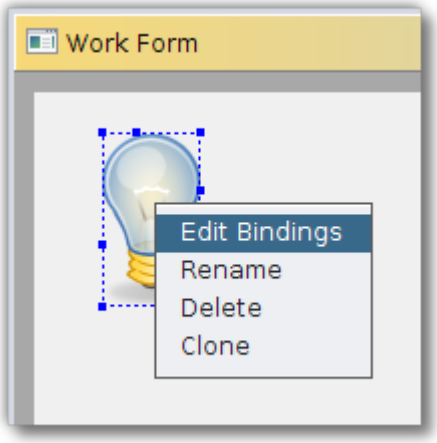
В качестве изображения будем использовать компонент **Векторное рисование** 995 ( ). Перетащите его из палитры компонентов 430 и поместите в корневой панели рабочей формы 426 виджета:



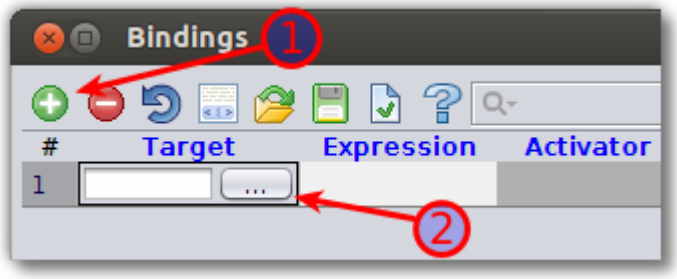
Теперь вы можете загрузить любое SVG-изображение в компонент: нажмите кнопку **Выбрать** в свойстве **SVG-файл**, затем найдите и откройте подходящий файл в появившемся диалоговом окне. Измените размер компонента по своему желанию.

3. Добавление привязок

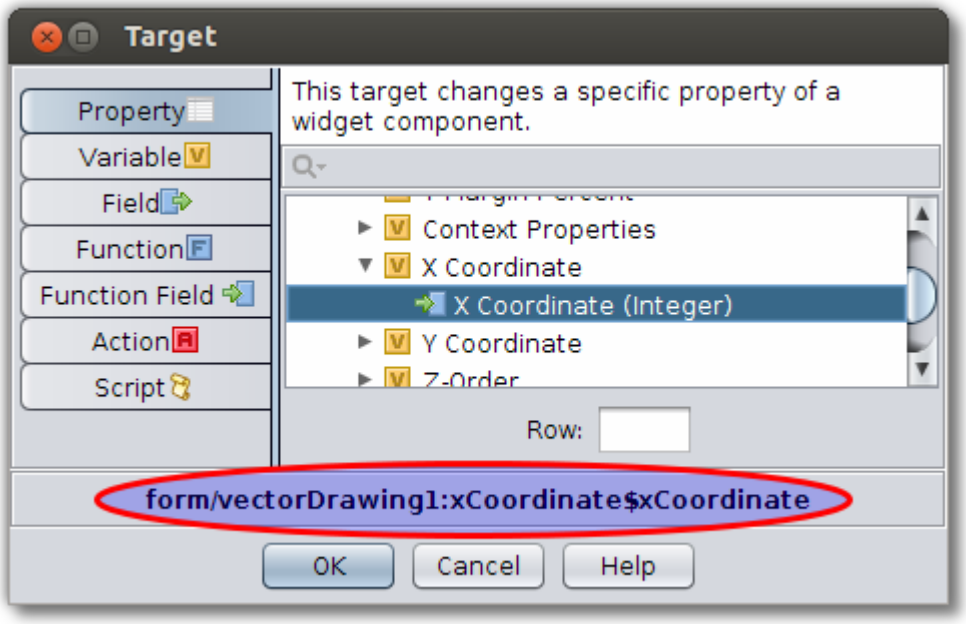
Выберите компонент и нажмите по нему правой кнопкой мыши. В открывшемся всплывающем меню выберите пункт **Редактировать привязки**:



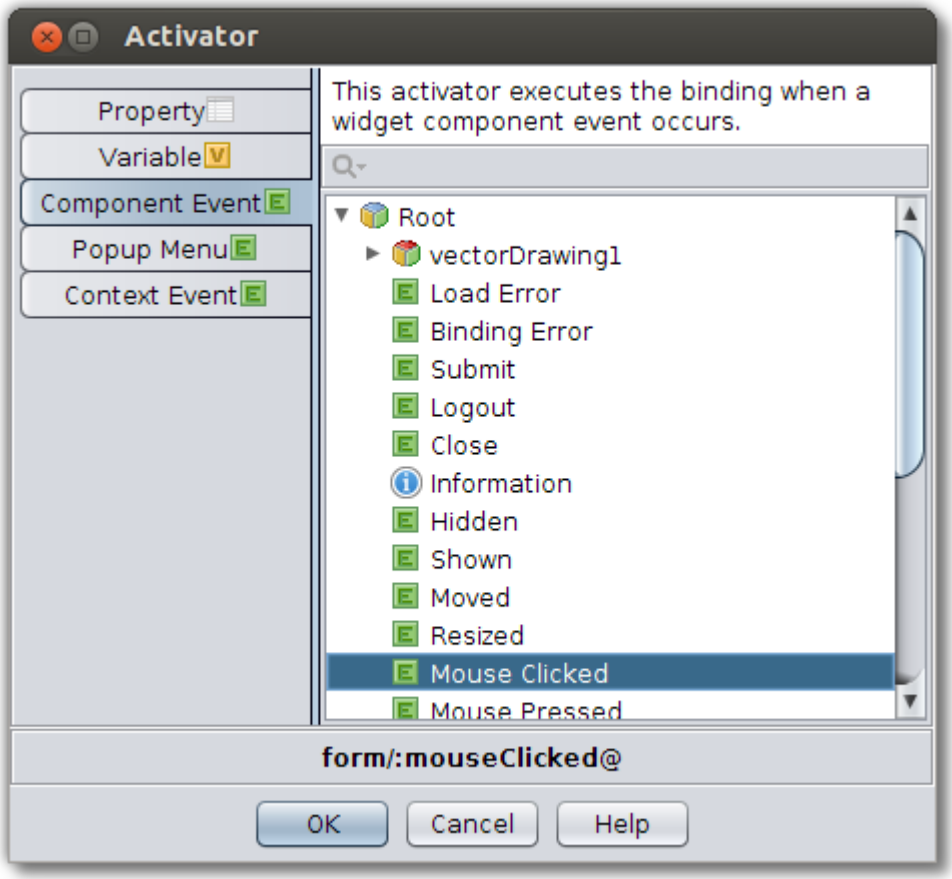
Должен появиться диалог Привязки. Затем нажмите кнопку **Добавить** ( ) для добавления новой пустой привязки. После этого кликните на кнопке многоточие **[...]** у **Цели**:



Это действие откроет мастер-программу **Цели**. Здесь необходимо выбрать, какое свойство будет меняться при помощи выполнения привязки. В нижней части диалогового окна отобразится ссылка на выбранное свойство. Здесь выберите свойство **xCoordinate** нашего компонента **Векторное рисование** (так что цель будет указывать на `form/vectorDrawing1:xCoordinate$xCoordinate`) и кликните OK.



Затем сделайте то же самое с полем **Активатор** 1296 и выберите событие **Мышка нажата** компонента **Корневая панель** во вкладке **Событие компонента**. Это означает, что привязка будет вызываться при помощи клика по корневой панели.



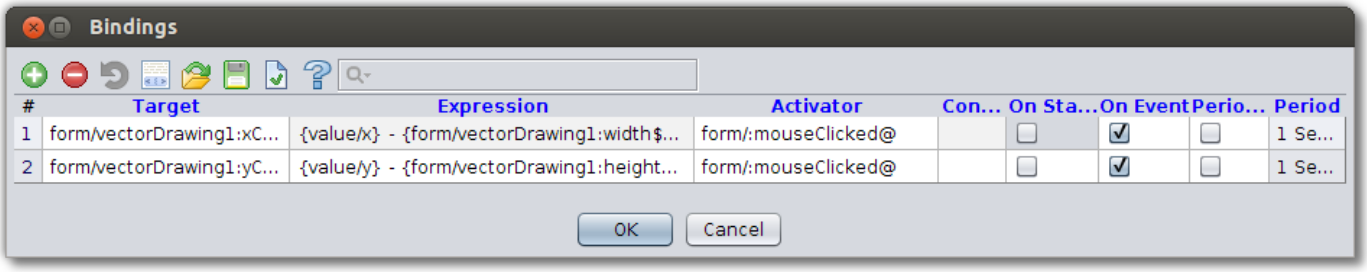
Теперь неоходимо выбрать **Выражение** 1296, которое определяет каждый раз вызова привязки и записи ее результата в цель. Используется специальная схема 117 **'value'** для ссылки на параметры событий. Чтобы получить событие клика мыши x-координаты, нужно использовать следующую ссылку: `{value/x}`.

Полный список параметров событий компонентов описан в разделе События компонентов 1285.

Чтобы центрировать изображение на позиции клика, нужно переместить x-координату нового компонента на половину ширины компонента. Итоговое выражение должно быть следующим: `{value/x} - {form/vectorDrawing1:width$width} / 2.` В написании выражений поможет Редактор выражений 404.

Итак, тепепрь, когда у нас есть привязка для x-координаты, теперь нужно то же самое для y-координаты. Для создания привязки мы можем использовать функцию дублирования Редактора таблиц данных 382: выберите привязку x-координаты в редакторе **Привязок**, нажмите кнопку **Добавить** (), удерживая клавишу **Shift**. Это действие создаст копию оригинальной строки в таблице привязок. Останется лишь исправить созданную привязку. Установите **Цель** на `form/vectorDrawing1:yCoordinate$yCoordinate`, а **Выражение** на `{value/y} - {form/vectorDrawing1:height$height} / 2.`

Наконец, нужно снять галочки с поля **При запуске**, чтобы привязка работала толко по данному событию. Итоговая таблица **Привязок** должна быть следующей:



4. Сохранение и запуск виджета

Кликните на **Готово** () в инструментальной панели Редактора виджетов, чтобы завершить редактирование и запись виджета.В системном дереве 370 должен появиться Контекст виджета 1627 () **Обработка событий компонентов**. Выберите действие **Запуск виджета** () во всплывающем меню виджета и откройте наш интерактивный виджет.

18.7 Создание инструментальной панели для монитори

Когда AtomMind используется для мониторинга различных электронных устройств, важно иметь возможность просматривать их рабочий статус и важные параметры в режиме реального времени. Более того, может оказаться необходимым оперативно выполнить определенные операции в ответ на изменения в отслеживаемых данных. Обычные средства AtomMind, такие как мониторинг событий устройства в журнале событий 398 и ответы на них с использованием избранного 2186 могут быть неудобны в некоторых случаях. Эти виджеты могут также содержать кнопки и прочие контролирующие средства ввода для разрешения интерактивной связи с устройством.

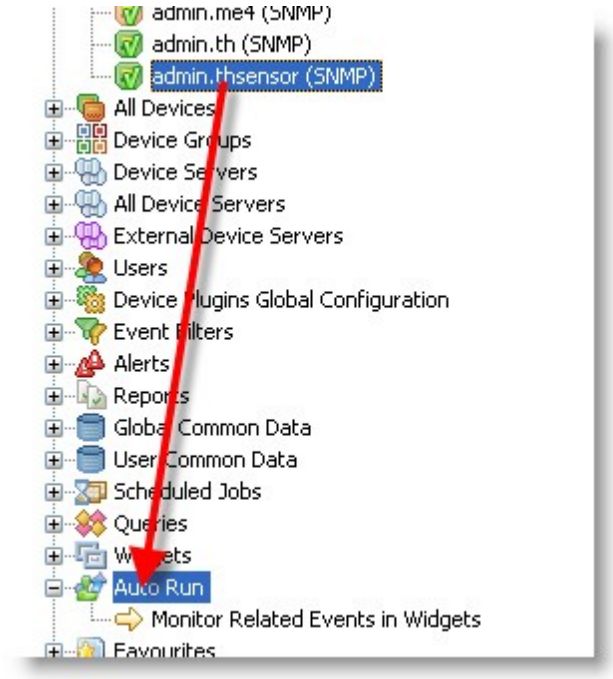
Эти виджеты должны быть постоянно активными, когда системный оператор активен, а AtomMind Client 359 запущен. Они должны автоматически запускаться при запуске AtomMind Client. В этом уроке мы покажем, как сочетать виджеты со свойством автозапуска 941 для формирования так называемой ***инструментальной панели***. Приборная панель - это группа виджетов, которые запускаются автоматически и размещаются в предварительно определенных положениях для обеспечения быстрого доступа к важным для системы параметрам и функциям.

Например, мы будем использовать виджет ***Мониторинг температуры/влажности***, созданный в уроке Как создать виджет для мониторинга вашего устройства 1643. Конечно, любой другой виджет (или виджеты) могут быть использованы для создания инструментальной панели.

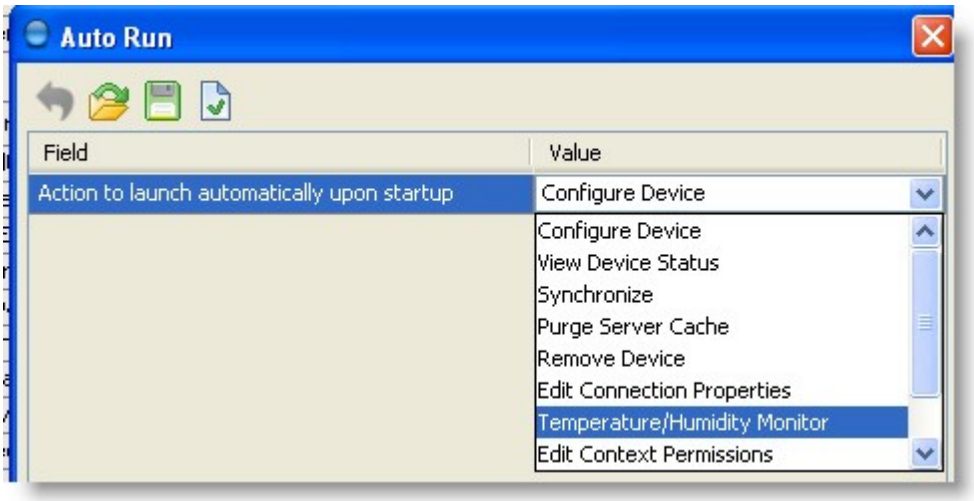
1. Добавление действия Запустить виджет к автозапуску

Во-первых, мы открываем узел для устройства, который необходимо отслеживать. Контекстное меню данного узла должно содержать действие Запустить виджет 948. Когда такое действие существует, это означает, что определенный виджет был создан 946 для работы с этим устройством.

Перетащите узел Системного дерева устройства () в узел ***Автозапуск*** ():



Выберите действие Запустить виджет (в этом случае ***Мониторинга температуры/влажности***) из раскрывающегося списка.



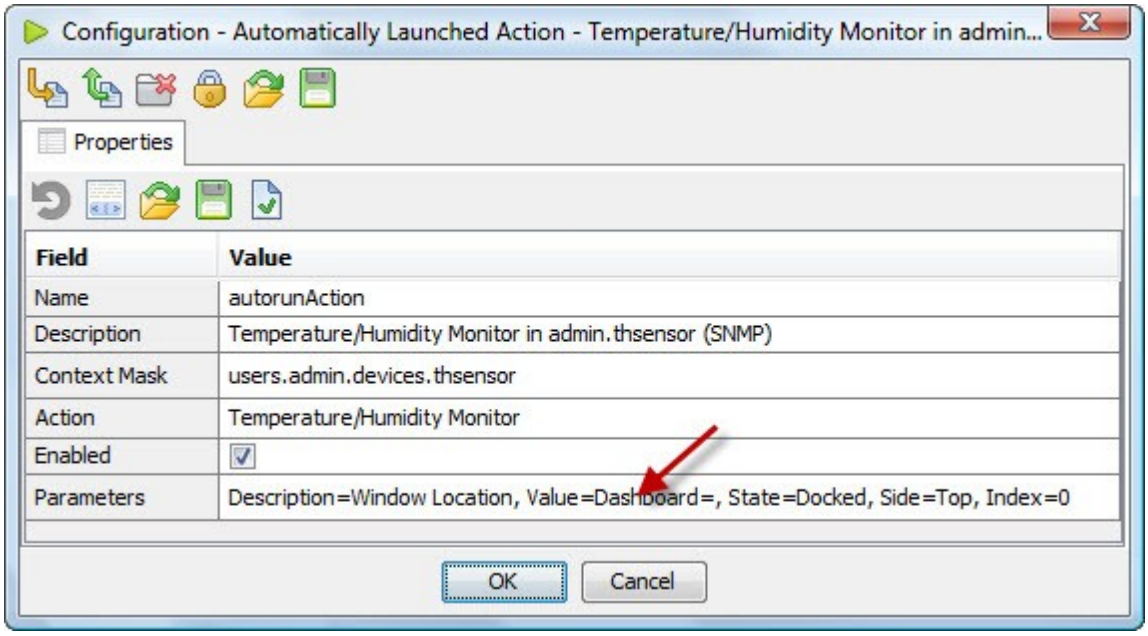
Будет создано новое действие Автозапуск ( ):

Создайте действия Автозапуск для всех других пар виджета/устройства, которые должны появляться в инструментальной панели аналогичным образом.

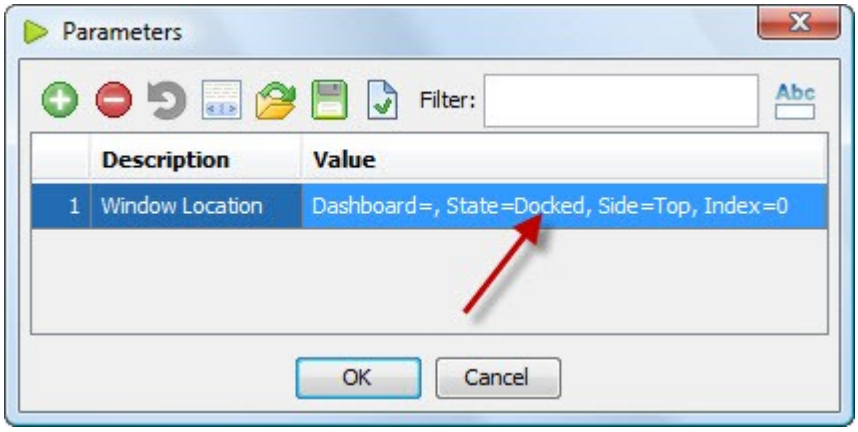
2. Создание раскладки инструментальной панели

Правой кнопкой мыши кликните по действию Автозапуск и выберите из контекстного меню ***Настроить*** ( ).

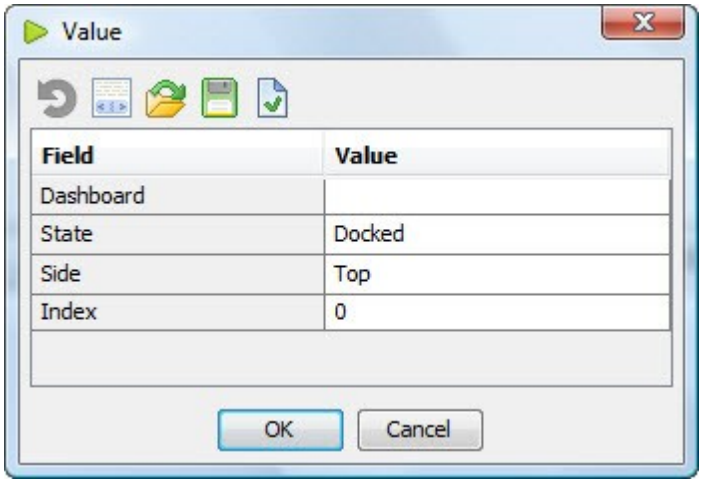
Кликните по полю **Параметры** в диалоговом окне настройки:



В диалоговом окне Параметры кликните по элементу **Расположение окна**:



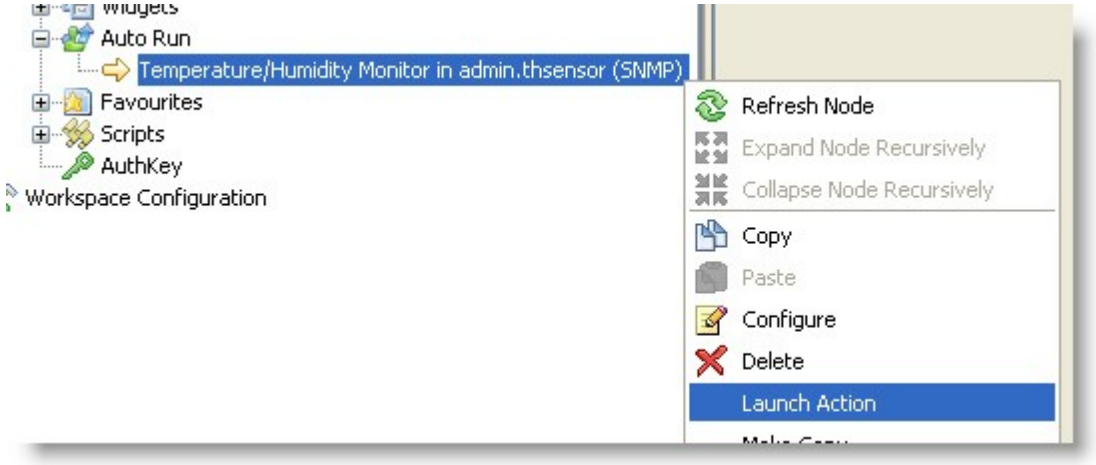
А теперь Вы можете редактировать имя инструментальной панели, чтобы разместить Ваш виджет и его расположение внутри панели:



Дополнительная информация о свойстве расположение окна содержится здесь.

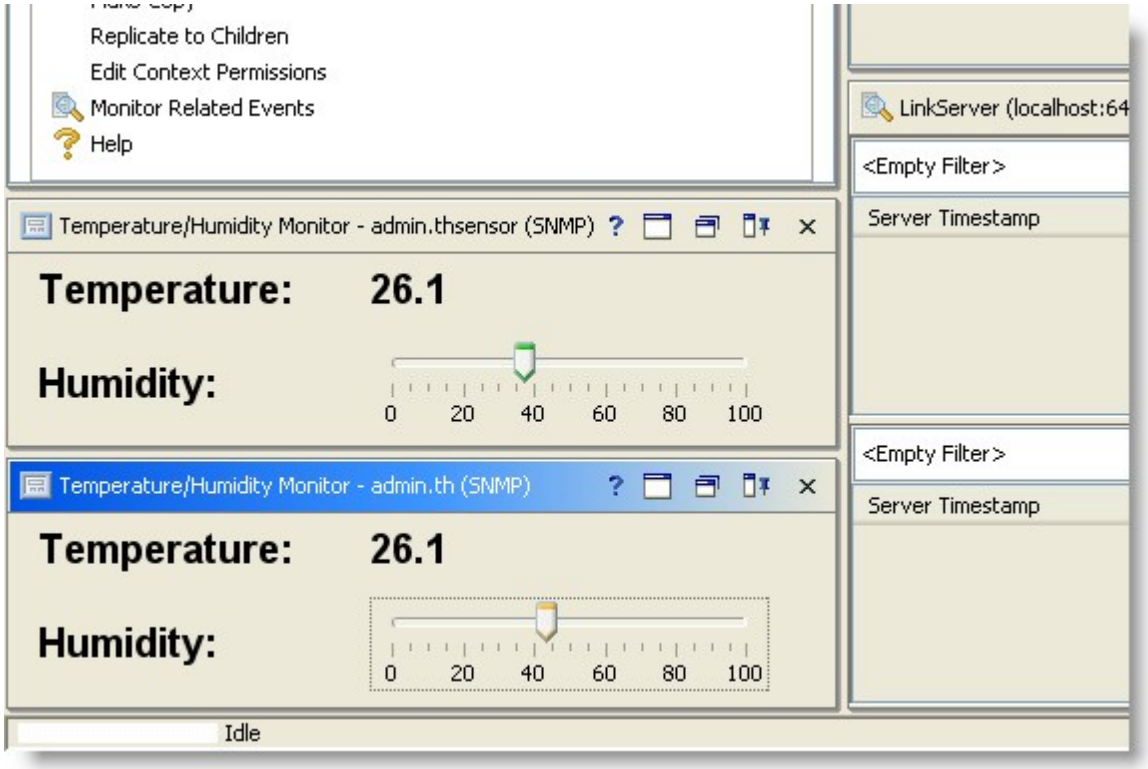
3. Отладка раскладки инструментальной панели

Правой кнопкой мыши кликните по узлу Действие автозапуск и выберите из контекстного меню **Запустить действие**:

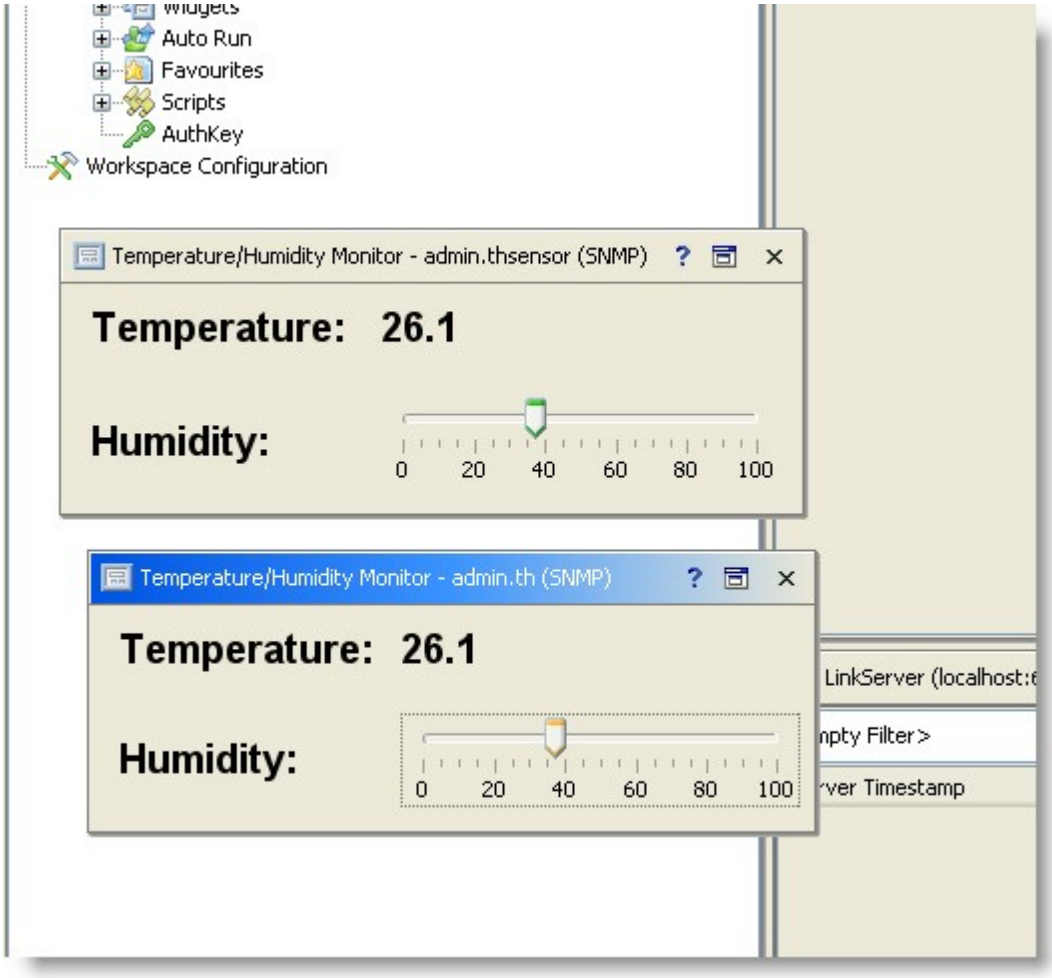


Запустится виджет, и появится окно нового виджета. Переместите его в нужное место. Оно может быть плавающим или пристыкованным к другим окнам. Дополнительную информацию о раскладке окна см. в разделе Настройка закрепляемых панелей[366].

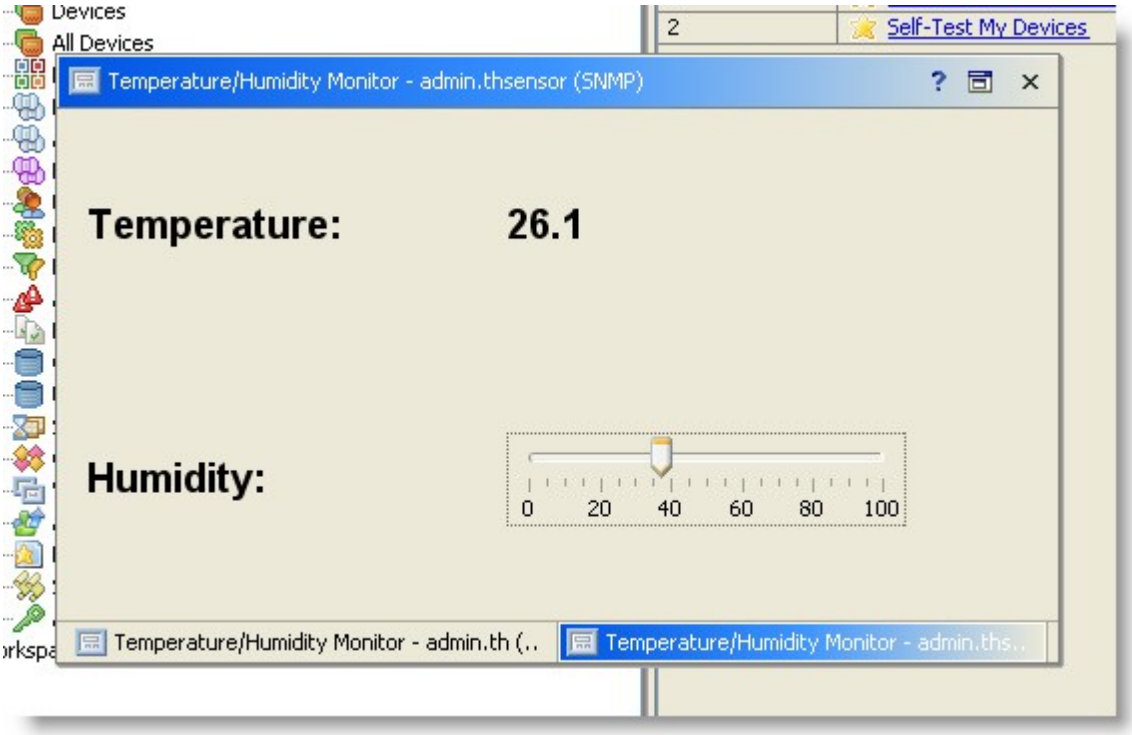
Повторите аналогичные этапы для каждого созданного действия Автозапуска. Вот как может выглядеть инструментальная панель со всеми закрепленными окнами виджета:



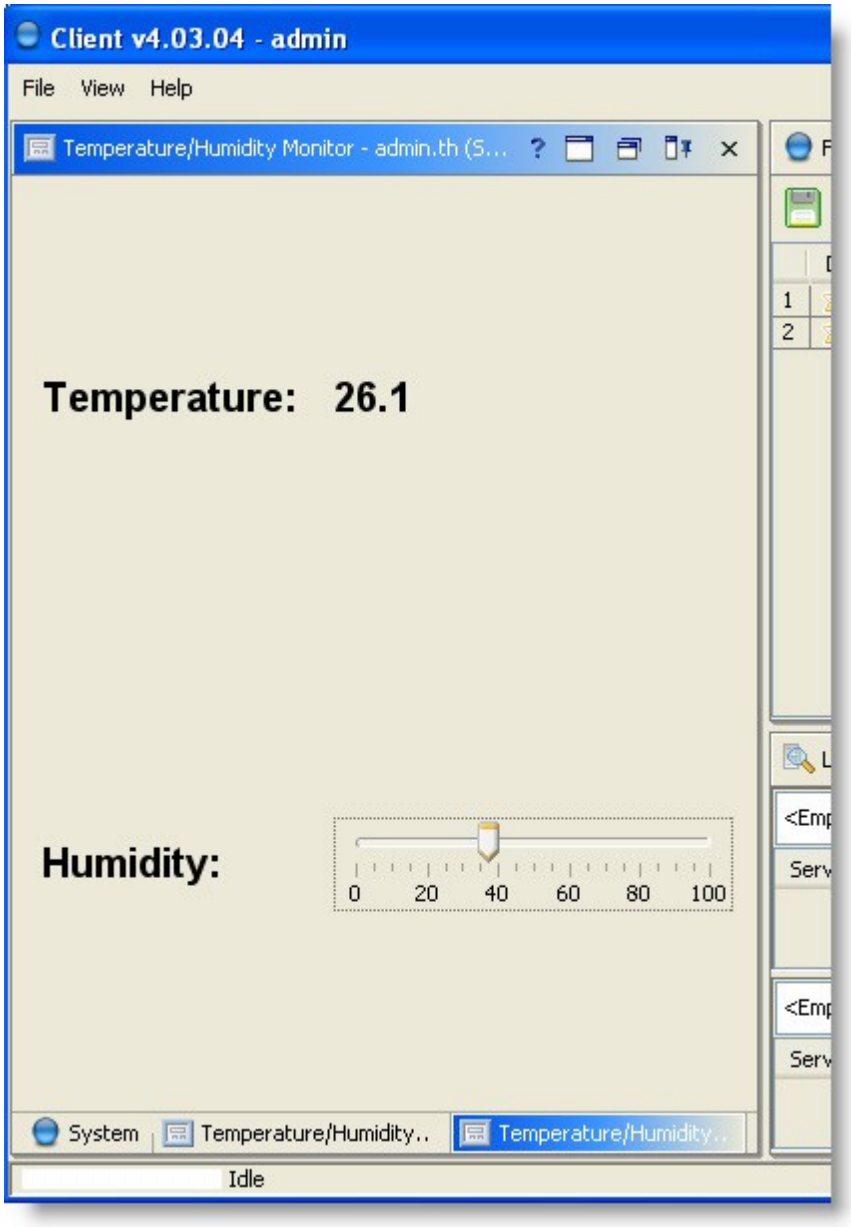
Инструментальная панель с отдельными плавающими окнами:



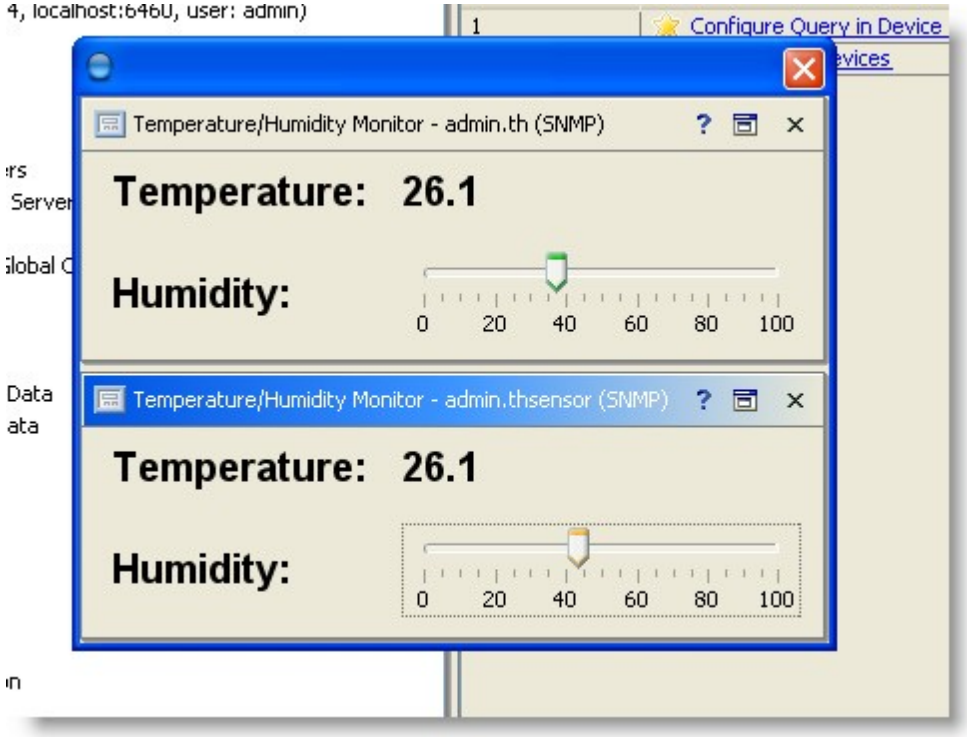
Плавающая инструментальная панель с окнами, расположенными, как вкладки:



Инструментальная панель с окнами-вкладками, сгруппированными с Системным деревом:



Инструментальная панель с несколькими виджетами, сгруппированными в одном плавающем окне:



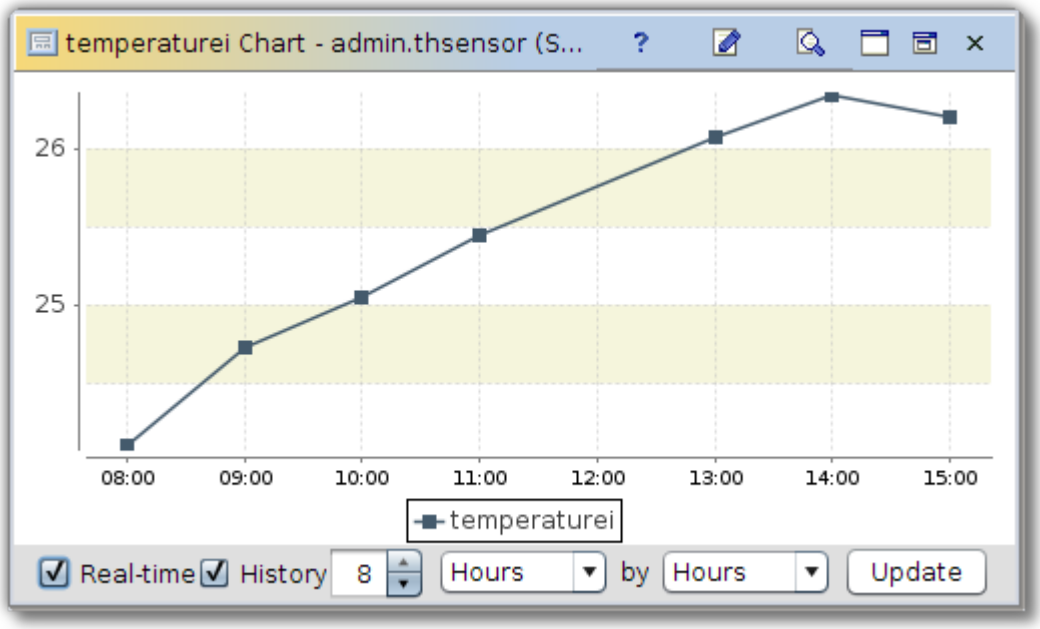
Теперь Вы можете выйти из AtomMind Client. Ваша инструментальная панель появится автоматически в следующий раз, когда вы зарегистрируетесь. Вы можете добавлять, удалять и перемещать его компоненты в любое время. Все изменения будут автоматически сохранены и применены вновь при следующем запуске AtomMind Client.

18.8 Мониторинг параметров устройства при помощи д

Диаграммы очень удобны для отображения изменений параметров устройства. В этом уроке мы Вам покажем, как отслеживать данные о температурном режиме, поступающие от сенсора, при помощи диаграммы. Мы не рассматриваем здесь подключение сенсора к серверу, полагая, что он уже находится в режиме on-line, и система видит его как устройство.

В AtomMind диаграммы являются частью виджетов 943. Мы собираемся создать виджет, содержащий простой компонент диаграммы 1051.

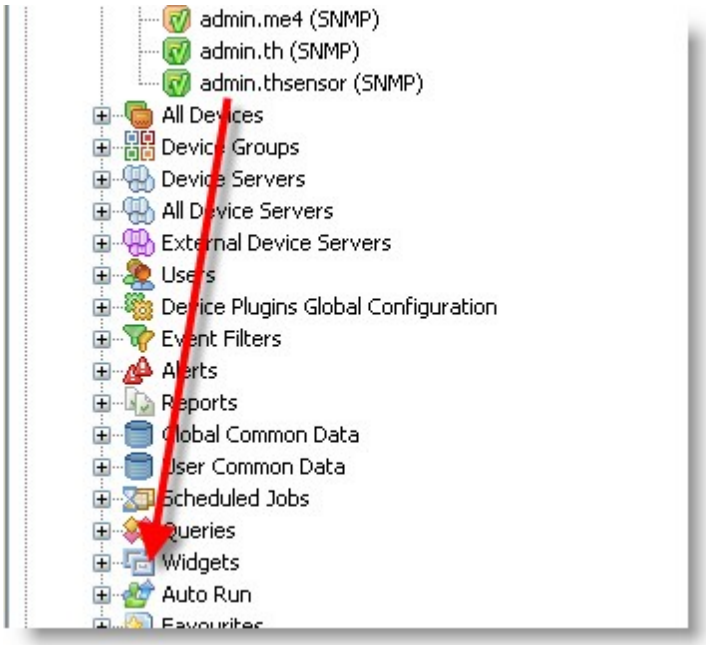
Наиболее простой способ создать диаграмму для переменной AtomMind - запустить действие настроить устройство 1495 ( ), правой кнопкой мыши щелкнуть по нужной переменной и выбрать элемент создать диаграмму 1624 ( ). Это действие создаст новый виджет с компонентом диаграммы и несколькими другими компонентами для осуществления оперативного контроля за параметрами диаграммы. Созданный виджет будет выглядеть следующим образом:



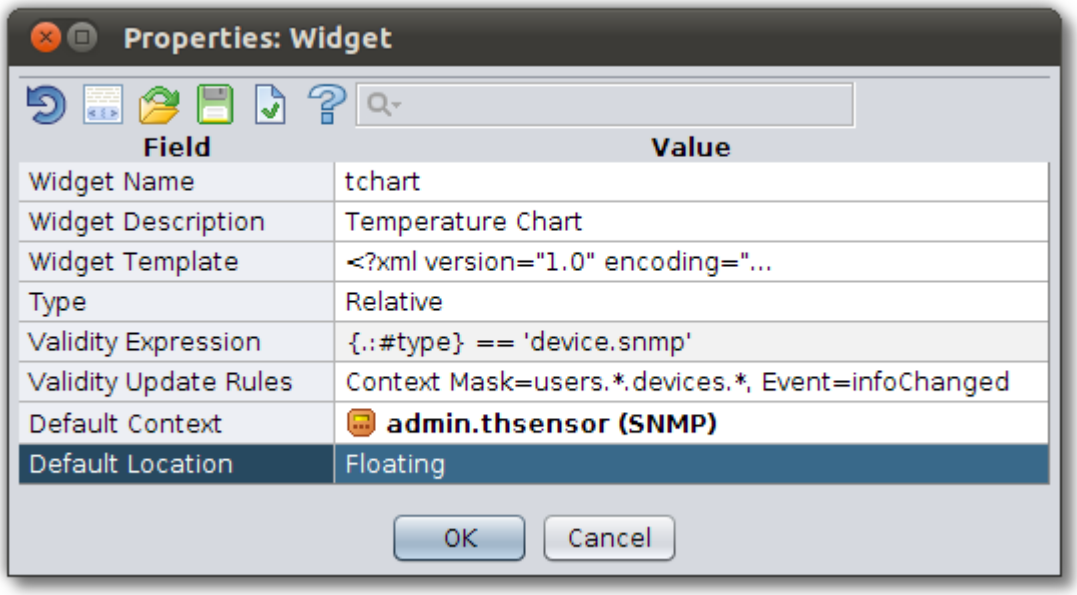
Еще один способ получить диаграмму с изменениями переменной - создать виджет вручную, добавить в нее компонент диаграммы и настроить диаграмму.

1. Создание виджета

Перед началом создания виджета, найдите узел устройства, которое необходимо отслеживать в системном дереве 370 ( ). Перетащите его на узел **Виджеты** ( ):



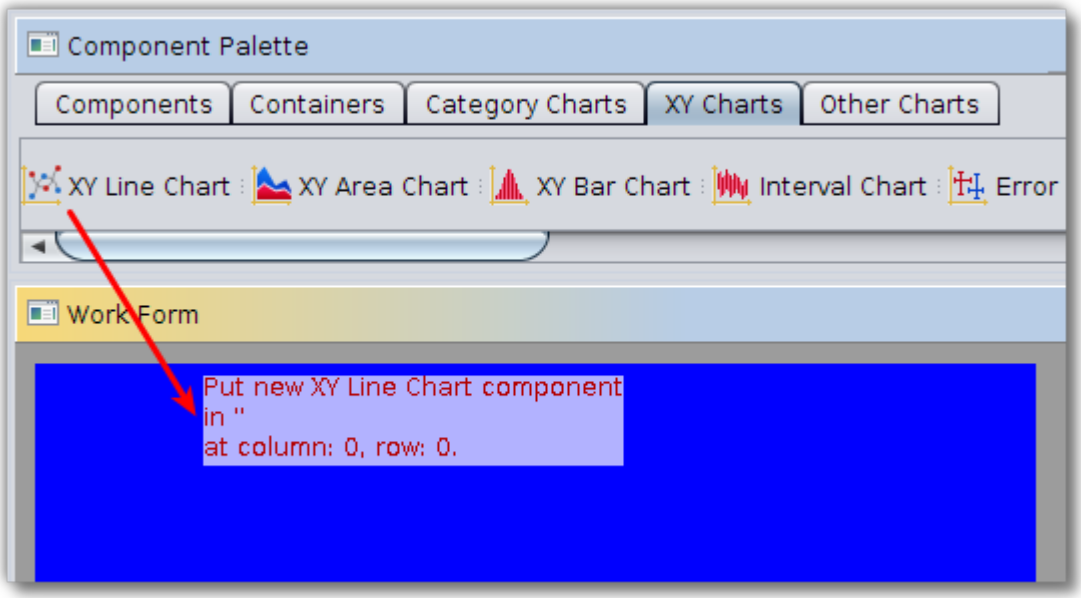
Во всплывающем меню выберите действие **Создать виджет** ( ). Откроется диалоговое окно свойства виджета 947. Установите для **Имя виджета** `tchart`, **Описание виджета -** `Temperature Chart` и кликните OK.



Это запустит редактор виджетов 423 с пустым шаблоном виджета 946. Если перетащить устройство на узел виджета, диаграмма будет доступна для контекста каждого терминала устройства 1494.

2. Добавление компонента диаграммы

Перетащите компонент **Диаграмма строки XY** (XY Line Chart) из палитры компонентов 430 в Корневую панель виджета:

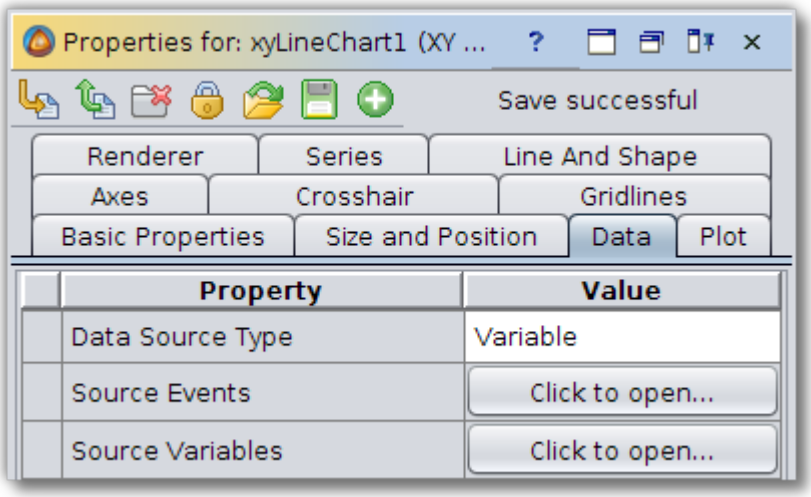


Это добавит новый компонент диаграммы в шаблон виджета 946.

3. Настройка источника данных

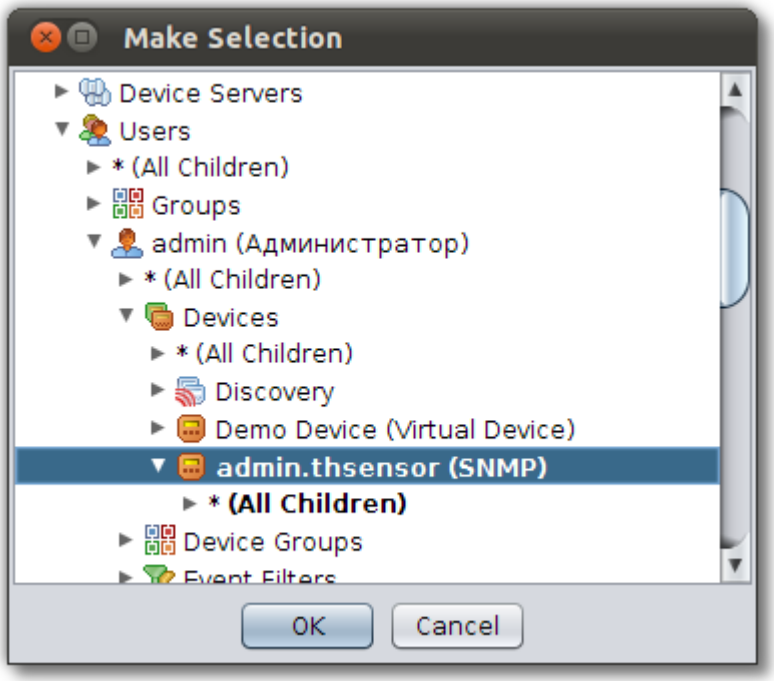
Наша диаграмма покажет изменения в значении переменных 61 контекста. У сенсора температуры есть одна переменная, которую можно легко представить графически. Это **temperaturei**, содержащая целое значение температуры, выраженное в градусах Цельсия, умноженное на 10 (т.е. значение 234 - это 23.4 градуса Цельсия). Значение температуры, которое содержится в одном поле этой переменной, также называется **temperaturei**.

Таким образом, наша диаграмма будет основана на данных переменной. Кликните по диаграмме в рабочей форме и измените свойство *Тип источника данных* на *Переменную* во вкладке *данных* окно свойств 431:



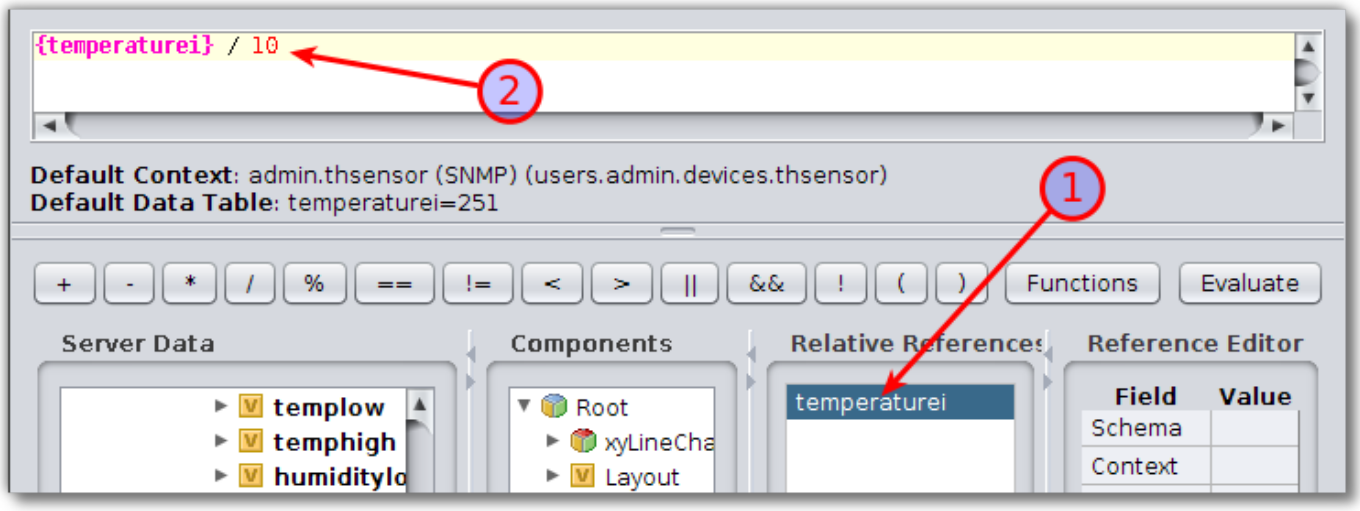
Затем кликните Переменные источника[1055], чтобы установить серию новых данных. Откроется окно Переменные источника. Кликните **Добавить ряд** (⊕), чтобы создать серию новых данных, а затем отредактируйте ее:

- Установите **Имя** на `Temperature`;
- Переключитесь в поле **Контекст** и кликните **[...],** чтобы отредактировать путь контекста интерактивно в селекторе объектов[402]. Выберите устройство из дерева (его название должно быть выделено жирным шрифтом, поскольку это контекст по умолчанию) и кликните OK:

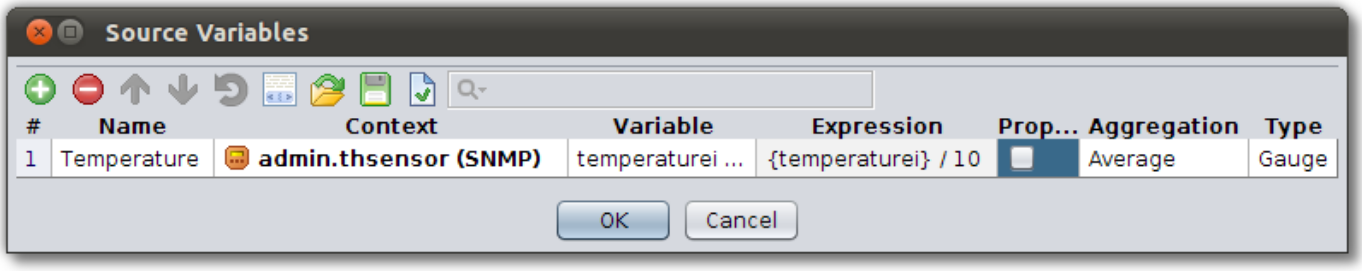


Действительное значение для поля **Контекст** будет точка (".") вместо полного пути к контексту устройства[1494]. Это относительный путь, который позволит виджету диаграммы работать с другими аналогичными устройствами, не только с тем, для которого диаграмма была создана.

- Выберите **temperaturei** из списка **Переменные**.
- Переключитесь в поле **Выражение** и кликните **[...],** чтобы отредактировать выражение данных интерактивно в редакторе выражений[404]. Во-первых, в окне Редактор выражений дважды кликните по полю **temperaturei** в разделе **Относительные ссылки**, чтобы добавить одну относительную ссылку. Во-вторых, добавьте / 10 к **Выражению**, чтобы разделить значение и, таким образом, конвертировать его в градусы Цельсия. Это необходимо, поскольку поле переменной изначально содержит значение температуры, умноженной на 10, позволяющее хранить его как целое число:



Другие колонки не изменяйте. Серия данных переменной должна выглядеть следующим образом:



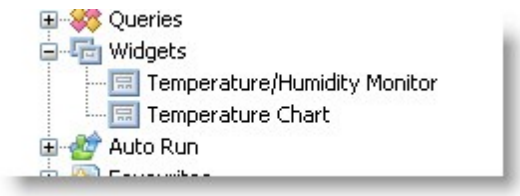
4. Настройка свойств диаграммы

На этом этапе Вы можете пожелать настроить свою диаграмму. Информацию о доступных свойствах и их описание см в разделе диаграммы. Например, Вы можете изменить флажки ***Включить исторические данные*** и ***Включить данные реального времени***, чтобы диаграмма включала текущее значение температуры, историю температуры и сразу два этих показателя. Другой возможный вариант - изменить ***Тип оси***. Если установить для него ***Ось данных*** **(Date Axis)** или **Ось периода (Period Axis),** это заставит горизонтальную ось отображать данные вместо целых числел**.** Еще одна полезная опция - **Отрисовщик**, который влияет на внешний вид диаграммы.

Можно также ограничить временной диапазон диаграммы, активировав опцию **Ограничить Диапазон времени** и изменив **Диапазон времени** на требуемое значение.

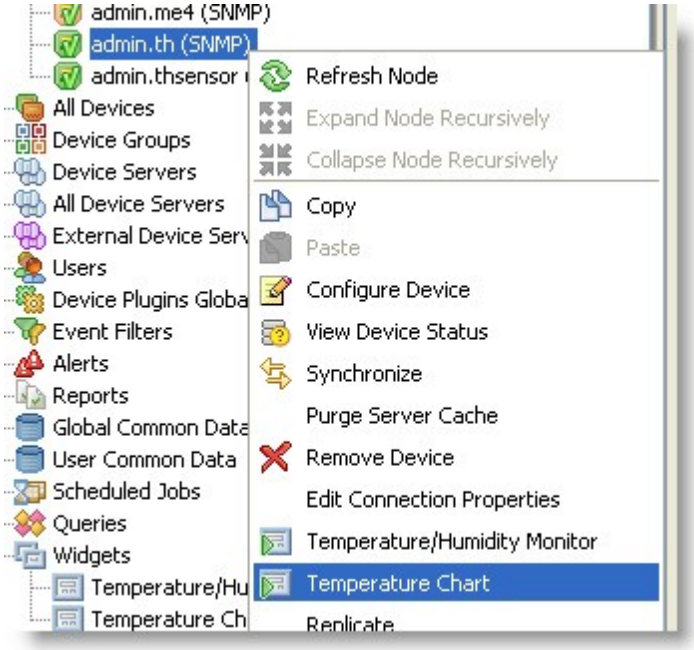
5. Сохранение виджета диаграммы

Кликните **Готово** ( ), чтобы завершить редактирование и сохранить виджет. Он должен появиться в Системном дереве:



6. Отображение диаграммы

Правой кнопкой мыши кликните по узлу системного дерева устройства ( ) Вашего датчика температуры. В контекстном меню Вы должны видеть новое действие запустить виджет, именуемое **Диаграммой температуры (Temperature Chart)** ( ):



Выберите это действие, чтобы запустить виджет для определенного датчика температуры. В AtomMind Client появится новое плавающее окно[366], содержащее виджет с нашей диаграммой:

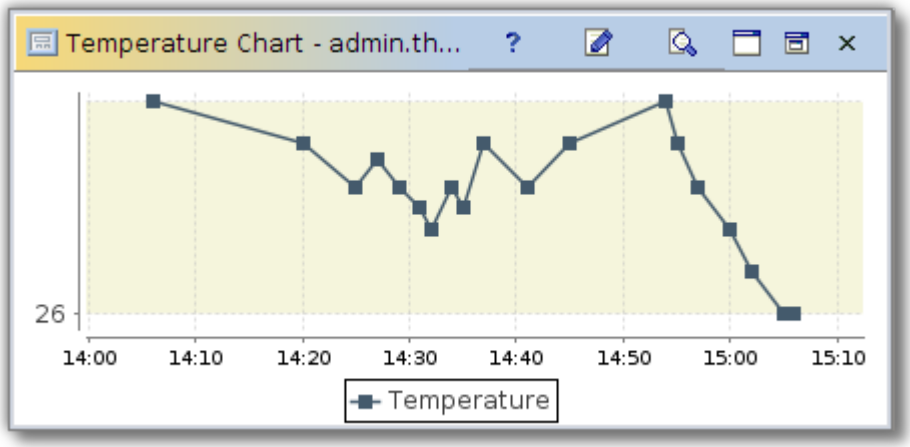


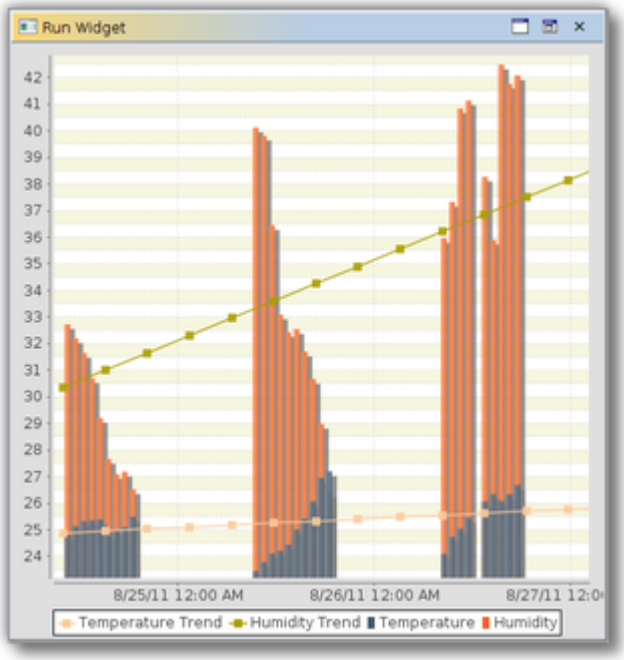
Диаграмма будет динамически обновляться, если в настройках диаграммы активировано **Включить данные реального времени**.

Если Вы хотите отслеживать температуру постоянно, когда запущен AtomMind Client, включите виджет диаграммы в инструментальную панель. См. как создать инструментальную панель для мониторинга Вашего устройства в режиме реального времени[1660].

18.9 Построение сложной диаграммы

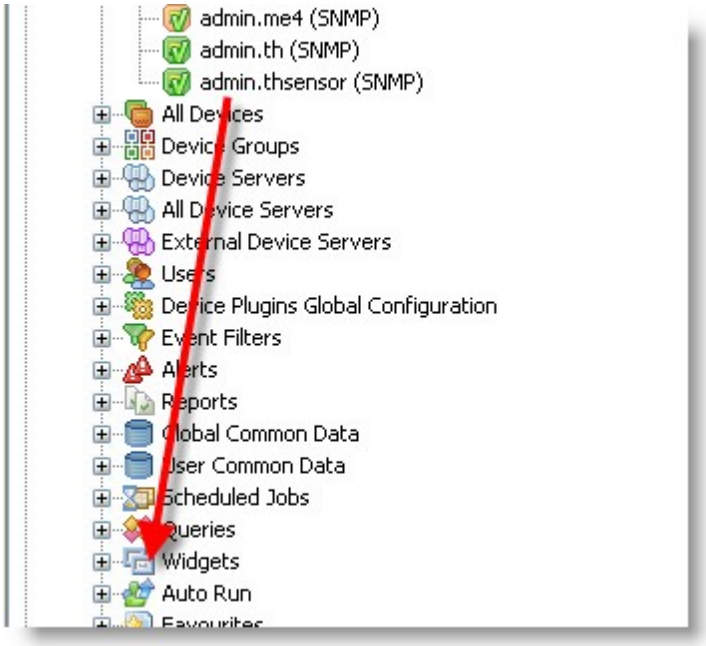
В этом уроке демонстрируется, как использовать смешанную диаграмму[1139] для сочетания двух диаграмм: одну для отображения данных и другую для отображения трендовых строк для этих данных. Мы будем использовать данные о влажности и температуре, получаемые от сенсора. Вопрос специфики связи сенсора с сервером не рассматривается в этом уроке, мы предполагаем, что оно уже находится в режиме online, а система видит его как устройство.

В конце этого урока мы получим виджет с диаграммой температуры и влажности с трендовыми строками:

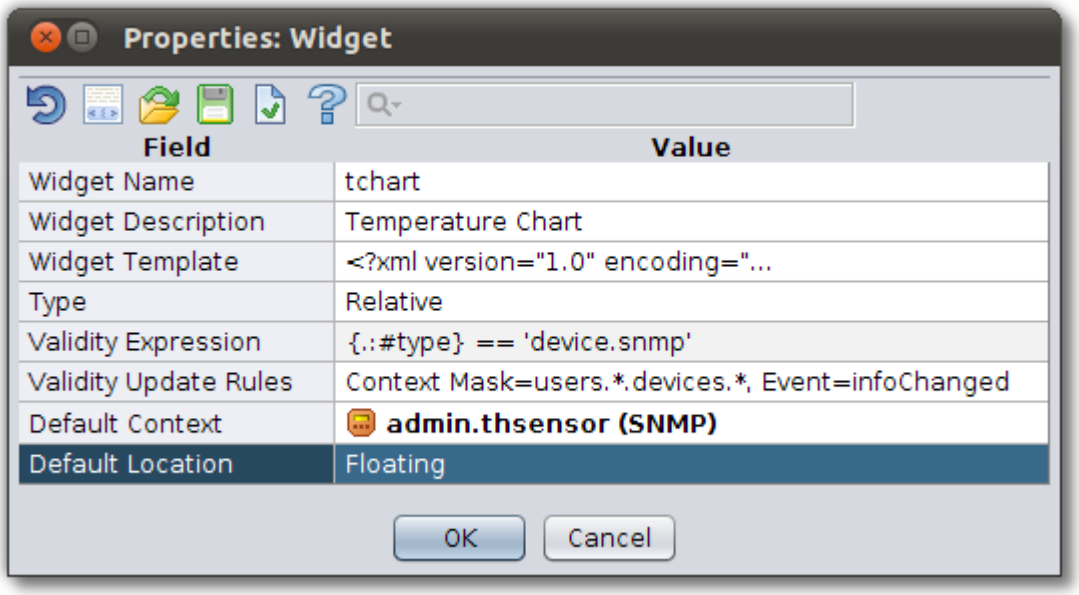


1. Начинаем создавать виджет

Перед тем, как приступить к созданию виджета, найдите устройство, которое необходимо для мониторинга в системном дереве 370 ( ). Перетащите узел устройства на узел **Виджеты** ( ):



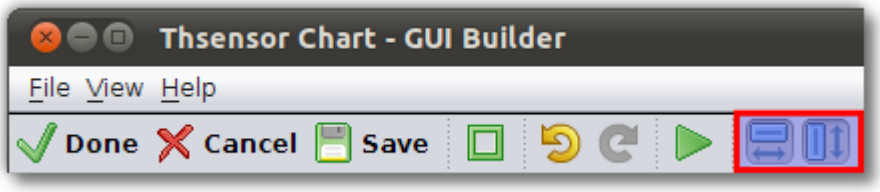
Во всплывающем меню выберите действие **Создать виджет** ( ). Откроется диалоговое окно свойства виджета 947. Установите **Имя виджета** как `tchart`, а **Описание виджета** как `Temperature Chart` и кликните OK.



Это запустит редактор виджетов 423 в пустом шаблоне виджета 946. Если перетащить устройство в узел Виджеты, наша диаграмма будет доступна для каждого контекста устройств 1494.

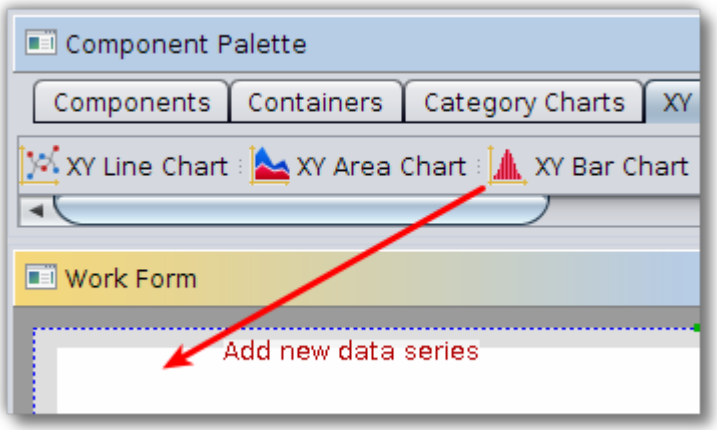
2. Добавление диаграмм

Перетащите компонент **Смешанная XY диаграмма** из палитры компонентов 430 в Корневую панель виджета. Это добавит новый компонент диаграммы в шаблон виджета 946. Включите **Горизонтальное** и **Вертикальное** заполнение для диаграммы, нажав кнопки в инструментальной панели 424.



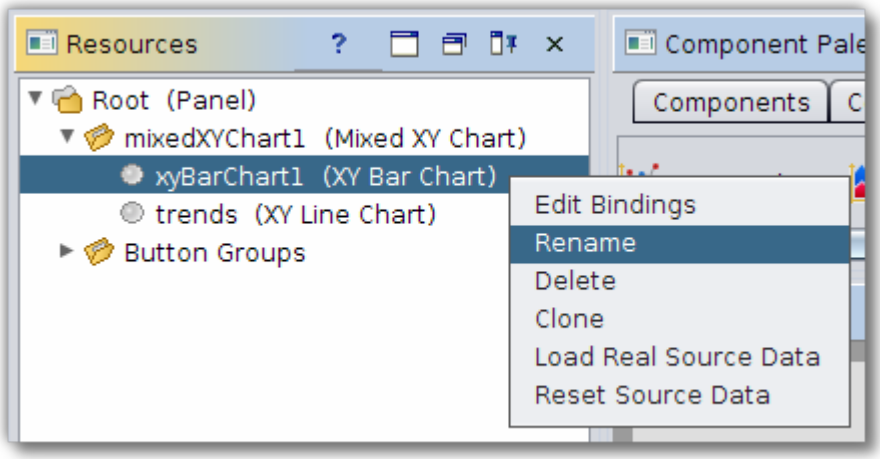
Смешанная XY диаграмма 1139 - это составная диаграмма 1160, которая может содержать другие XY диаграммы 1099, имеющие одни и те же (совместные) оси и графики. В частности, она позволяет одним диаграммам изображать тренды для других.

Перетащите **Столбчатую диаграмму XY** (XY Bar Chart) (для отображения температуры и влажности) и **Линейную диаграмму XY** (XY Line Chart) (для отображения трендов) в **Смешанную диаграмму XY** (Mixed XY Chart).



3. Переименование диаграмм

А теперь переименуйте **xyLineChart1** в `trends`, **xyBarChart1** в `thsensor`. Чтобы переименовать компонент, щелкните правой кнопкой мыши в Дереве ресурсов 428, выберите элемент **Переименовать** во всплывающем меню и введите новое имя в появившемся окне для редактирования.



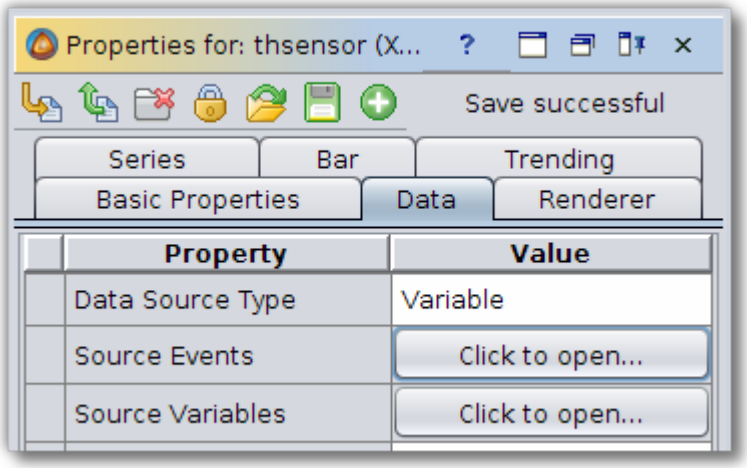
Компонент можно также переименовать в его рабочей форме во всплывающем меню.

4. Настройка диаграммы сенсора

Давайте установим столбчатую диаграмму **thsensor** на заднем плане, а линейную диаграмму на переднем. Для этого выберите диаграмму **thsensor** в дереве ресурсов и установите свойство **Z-Order**, принадлежащее **thsensor,** на 2 в окне свойств.

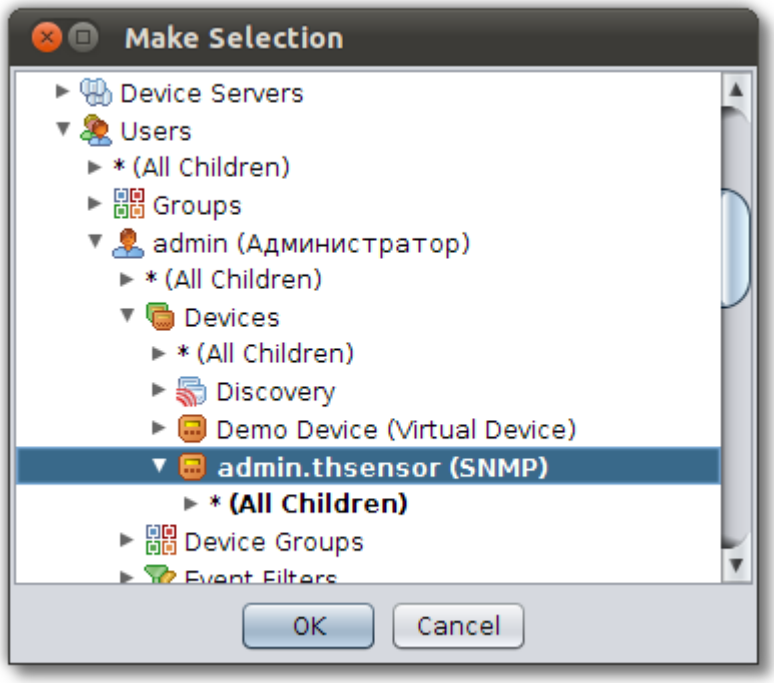
Теперь сделайте так, чтобы диаграмма отображала температуру и влажность устройства **thsensor**. У сенсора температуры есть переменная **temperaturei**, содержащая целочисленное значение температуры по Цельсию, умноженное на 10 (т.е. значение 234 - это 23.4 градуса Цельсия) и переменную **humidityi** , которая является значением влажности в процентах и умноженное на 10.

Диаграмма **thsensor** будет основана на данных переменной. Измените значение **Типа источника данных** на **Переменную** во вкладке **данных** Окно свойств:



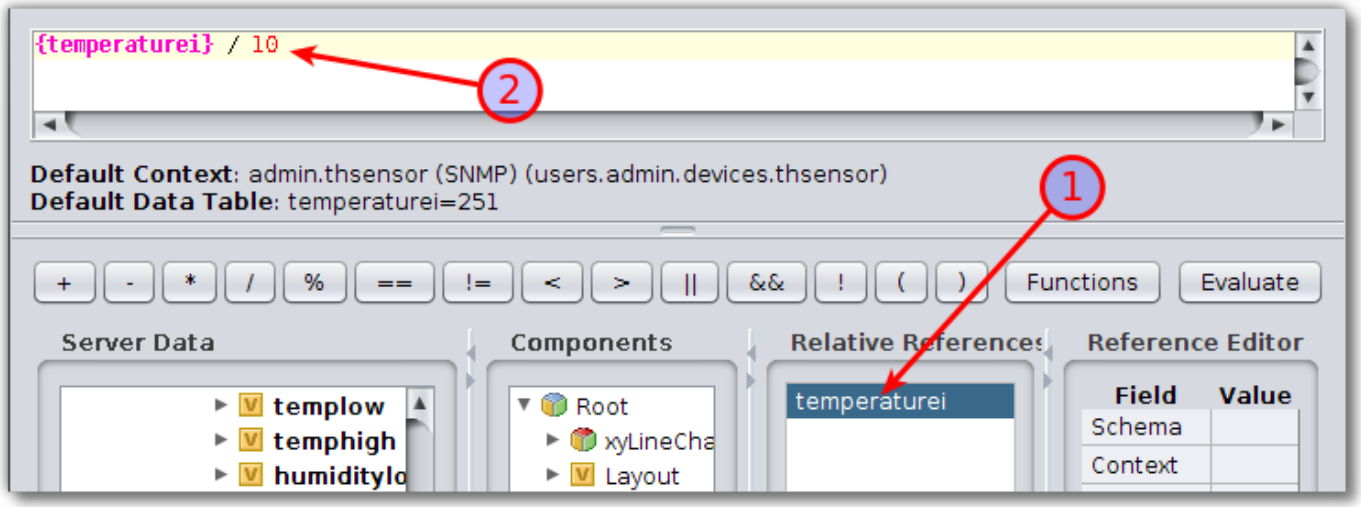
Теперь кликните по элементу Переменные источника, чтобы установить новую серию данных. Откроется окно Переменных источника. Кликните **Добавить ряд** (), чтобы создать новую серию данных, а затем отредактируйте ее:

- Установите в качестве **Имени** `Temperature`;
- Переключитесь в поле **Контекст** и кликните **[...]**, чтобы отредактировать путь контекста интерактивно в селекторе объектов. Выберите устройство **thsensor** из дерева (название должно быть выделено жирным шрифтом, поскольку это *контекст по умолчанию*) и кликните OK:



Действительное значение для поля **Контекст** будет точка ("**.**") вместо полного пути к [контексту устройства](1494). Это относительный путь, который позволит нашему виджету диаграмм работать с другими аналогичными устройствами, не только с тем, для которого он был создан.

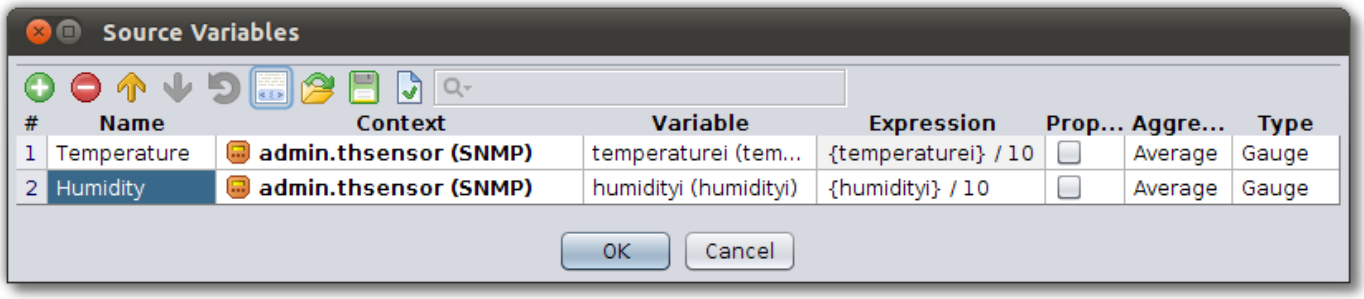
- Выберите **temperaturei** из списка **Переменные**.
- Переключитесь в поле Выражения и кликните **[...], чтобы отредактировать выражение данных интерактивно в** [редакторе выражений](404). Во-первых, в окне Редактора Выражений дважды кликните в разделе **Относительные ссылки** по полю **temperaturei,** чтобы добавить относительную ссылку. Во-вторых, добавьте / 10 к **Выражению**, чтобы разделить значение и, т.о., конвертировать его в градусы Цельсия. Это действие необходимо выполнить, поскольку поле переменной изначально содержит значение температуры, умноженное на 10, что требуется для хранения его как целого числа:



Примите выражение, кликнув по кнопке OK. Оставьте другие колонки диалогового окна [переменные источника](1055) неизмененными. Таким образом, Вы получаете запись для температуры.

- А теперь добавьте и установите вторую серию данных для переменной **humidityi** аналогично тому, как Вы выполняли для переменной **temperaturei**.

Так должна выглядеть серия данных переменной:



Добавляя вторую запись переменных источника, кликнув **Добавить ряд** (), удерживайте кнопку **Shift**. Это скопирует запись для температуры, таким образом для влажности Вам нужно установить поля **Имя**, **Переменная** и **Выражение** в дублируемой записи.

5. Настройка диаграммы трендов

Давайте настроим нашу диаграмму трендов для отображения линий тренда для диаграммы **thsensor**. Во-первых, в окне свойств переключите свойство **Типа источника данных** на тренд. Затем во вкладке ***тренд*** выберите **thsensor** в свойстве ***Имя* диаграммы источника**.

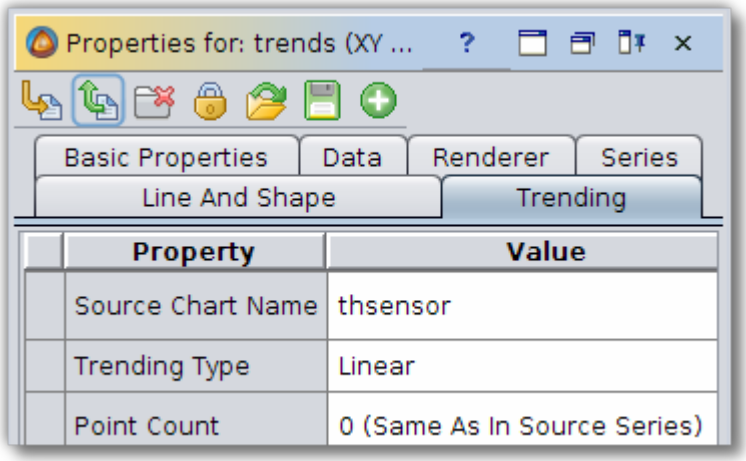
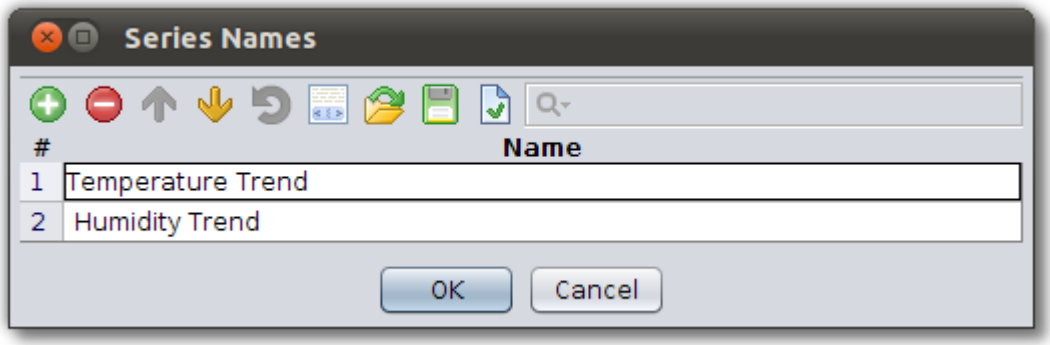


Диаграмма тренда создаст отдельные серии тренда для каждой серии диаграммы источника. Давайте теперь установим пользовательские имена серий тренда: кликните по свойству **Имена серий** и добавьте два ряда с именами `Temperature Trend` и `Humidity Trend`.

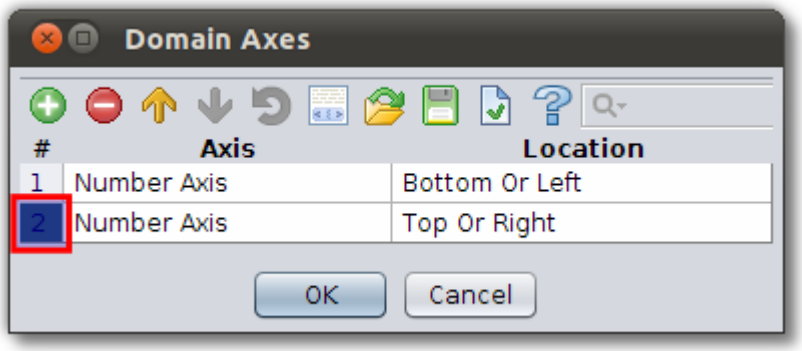


6. Очистка

Когда диаграмма настроена для тренда, она использует массив данных диаграммы источника для подсчета трендов и осей диаграмм трендов для отображения линий трендов. Таким образом, мы можем удалить диаграммы **трендов**, к которым были добавлены автоматически оси, добавив **Линейную диаграмму XY** (на этапе 2) в смешанную диаграмму.

- Выберите **mixedXYChart1** в дереве ресурсов.
- Перейдите в ярлык **Оси** и кликните в свойстве **Оси диапазона**.

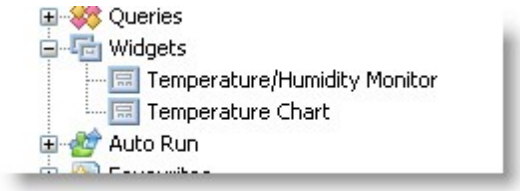
- Выберите второй ряд в появившемся диалоговом окне **Оси диапазона** и нажмите кнопку **Удалить ряд.** (), чтобы удалить неиспользованную ось домена. Кликните OK.



- Выполните аналогичное действие для **Осей домена.**
- Переключите тип для оставшейся **Оси домена** и **Оси данных**: кликните по оси и установите ее **тип** в появившемся диалоговом окне **Ось**.

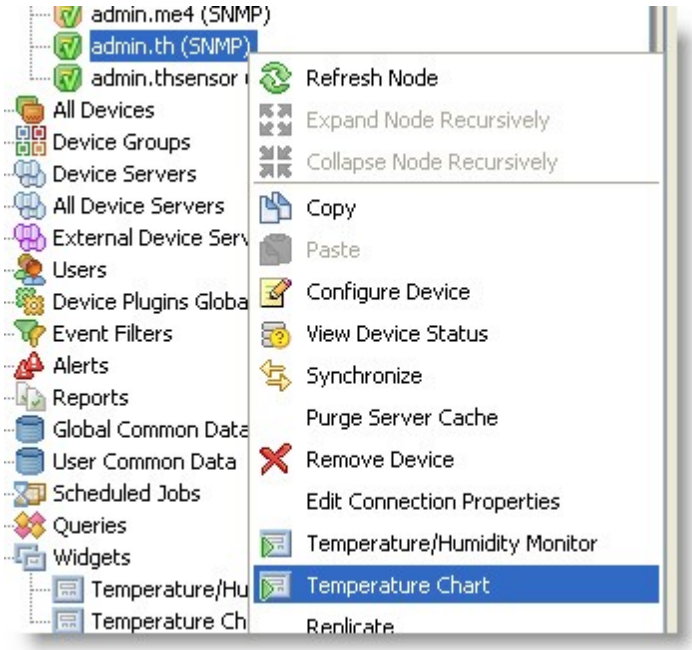
7. Сохранение виджета диаграммы

Кликните **Готово** (), чтобы закончить редактирование и сохранить виджет. Он должен появиться в Системном дереве:

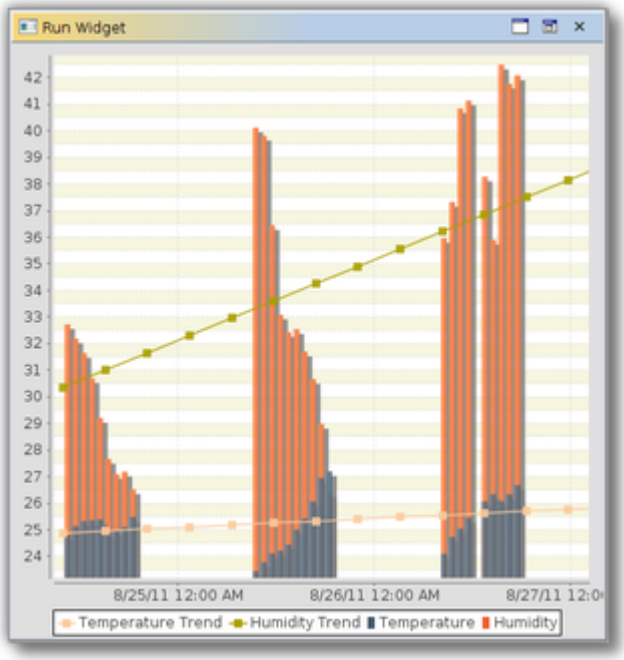


8. Отображение диаграммы

Правой кнопкой мыши щелкните по узлу устройства () Вашего температурного сенсора. Вы должны увидеть новое действие Запустить виджет[948], именуемое в контекстном меню **Диаграммой температуры** ():



Выберите это действие для запуска виджета. В AtomMind Client появится новое плавающее окно[366], содержащее виджет с нашей диаграммой.



Из этого урока Вы узнали, как сочетать простые диаграммы со смешанными диаграммами для прострoения трендовых строк. Существует также ряд иных возможностей сочетания диаграмм, которые можно использовать в проектах. См. составные диаграммы 1160.

18.10 Построение отчета о статусе устройства

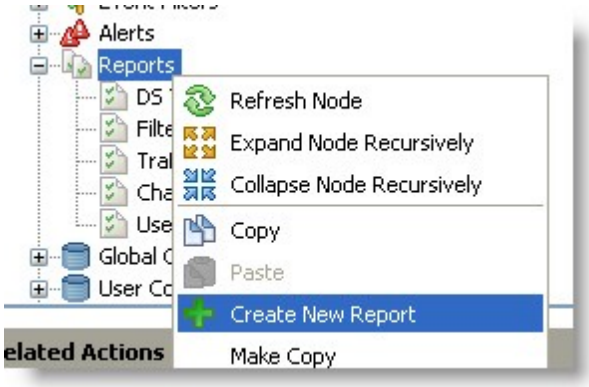
Функция оповещения 928 AtomMind полезна, когда необходимо получить данные в удобной для пользователя и пригодной для печати форме. Отчеты отображают активность устройства и другую статистическую информацию.

В этом уроке рассказывается, как создать отчет, который включает историю определенного события, сгенерированного устройством. Мы используем терминал учета времени, который подключен к AtomMind через Агент. Мы не рассматриваем тему подключения терминала к серверу в этом уроке, мы предполагаем, что он уже находится в режиме online, и его видит система.

Наш терминал генерирует событие 73 в любое время, когда сотрудник проводит своей электронной картой по считывающему устройству. Событие содержит ID владельца карты, тип события (**IN,** когда сотрудник входит в рабочую зону или **OUT** , когда он уходит) и временную метку.

1. Создание нового отчета

Чтобы создать новый отчет, найдите узел **Отчеты** ( ) в системном дереве 370. Щелкните правой кнопкой по нему и выберите **Создать новый отчет** ( ):



Можно также дважды щелкнуть по узлу **Отчеты**, чтобы запустить действие **Создать новый отчет**, поскольку это действие по умолчанию 88 в контексте Отчеты 1551.

В окне **Свойства отчета**:

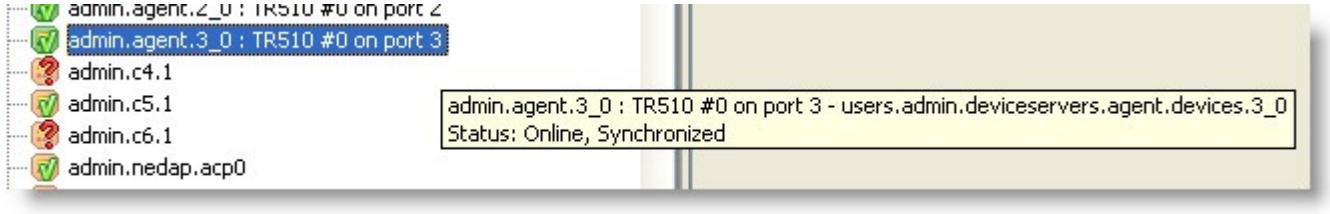
- Установите в поле **Имя** `trevents`
- Установите в поле **Описание** `Time Recorder Events`

А теперь нужно установить наиболее важный параметр отчета - **Выражение данных источника**. Это выражение 112 говорит AtomMind Server, как и где получить данные для заполнения в шаблоне отчета. На этапе создания отчета он также используется для получения данных для автоматического создания шаблона 930 отчета.

Мы будем использовать функцию получить 1516, определенную в контексте События для сбора истории событий 73 для отчета. Эта функция возвращает историю события в форме таблицы, одна запись на экземпляр события.

Функция принимает четыре параметра (см его описание 1516), но мы определим лишь первые два, а для остальных оставим значения по умолчанию.

В нашем случае первый параметр - это путь контекста устройства 1494. Мы можем определить этот путь, если найдем терминал в Системном дереве и проверим его всплывающую подсказку (tooltip). Его путь - **users.admin.deviceservers.agent.devices.3_0**:



Вы можете щелкнуть правой кнопкой мыши по устройству и выбрать **Копировать** из его меню контекста, чтобы скопировать его путь контекста в буфер.

Вы можете создать отчет, который включает события из множества терминалов. Следует лишь изменить путь контекста к маске 44, указывающей на множество устройств. Например, чтобы включить события от всех терминалов пользователя 478 **admin**, измените **Контекстную маску** на `users.admin.deviceservers.*.devices.*`.

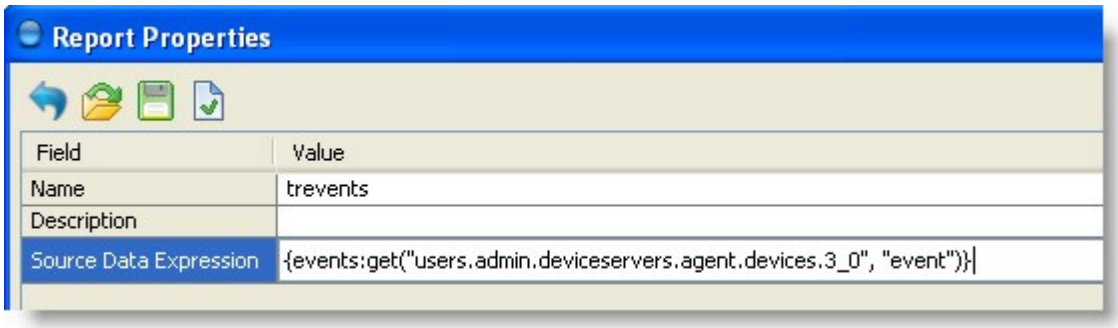
Второй параметр - это имя события, созданного терминалом. Вы можете найти его в журнале событий 398. Есть специальная колонка Событий, которая отображает имена событий. В нашем случае событие называется **событием**.

Наше **Выражение источника данных** будет состоять из одной ссылки 117, чтобы ***получить*** функцию с двумя параметрами.

- Установите в поле **Выражение данных источника**
  `{events:get("users.admin.deviceservers.agent.devices.3_0", "event")}`

Обратите внимание, что параметры функции заключены в двойные кавычки, поскольку они являются простыми текстовыми строками, не включенными выражениями. См. Функции в ссылках 120.

Окно **Свойства отчета** должно теперь выглядеть следующим образом:

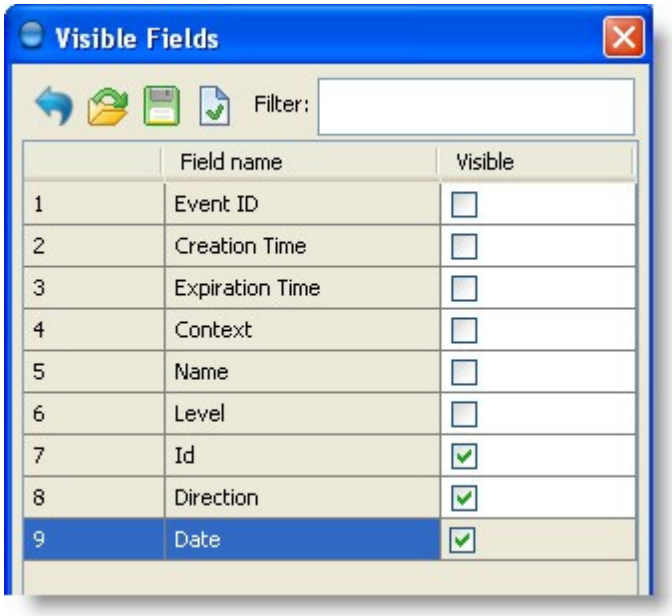


Кликните OK, чтобы продолжить. Должно открыться окно Свойства шаблона отчетов.

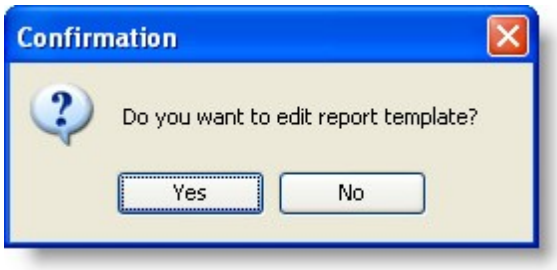
2. Настройка свойств шаблона отчетов

На данном этапе необходимо установить свойства шаблона отчетов. Шаблон отчетов будет автоматически сгенерирован 930 согласно данным свойствам и данным, собранным при помощи Выражения данных источника, заданного на предыдущем этапе.

- Установите в качестве **Заголовка отчета** `Time Recorder Events.`
- Измените **Ориентацию** на Портрет, поскольку у нас будут только три колонки в нашем отчете и данные будут лучше соответствовать в этом случае.
- Откройте таблицу Видимые поля и уберите выделение со всех полей за исключением тех, которые должны появиться в отчете. В нашем случае мы оставим **Id (идентификатор), Direction (направление)** и **Date (дату)**:



А теперь кликните дважды OK, чтобы продолжить. Сервер сгенерирует шаблон отчетов, и система предложит отредактировать его. Кликните **Да**:

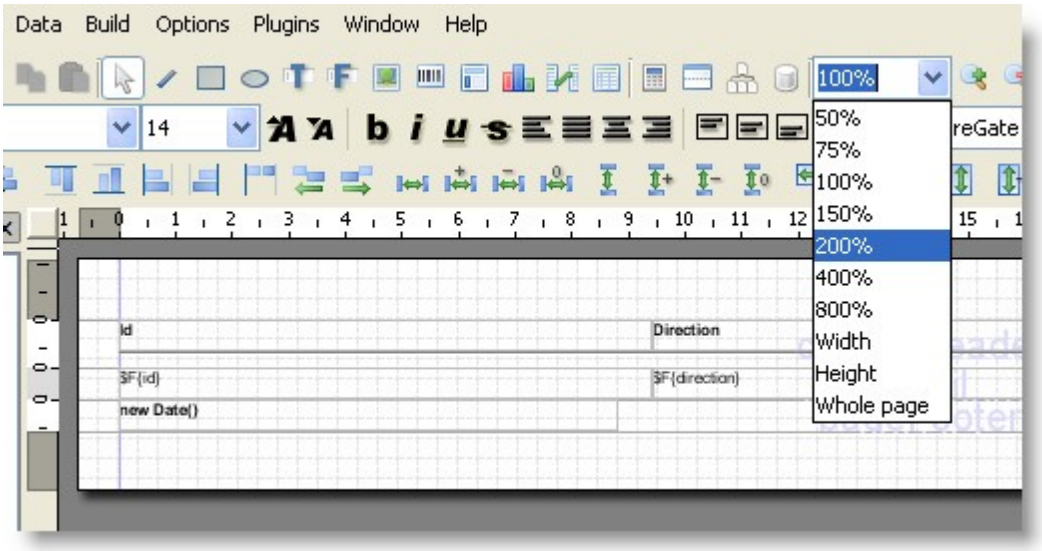


Запустится редактор отчетов 416, и теперь можно отредактировать только что сгенерированный шаблон.

3. Редактирование шаблона отчетов

Редактор отчетов - это многофункциональный компонент, но мы не собираемся выполнять сложное редактирование на данном этапе. Мы хотим лишь изменить в отчетах заголовки колонок.

Во-первых, установите размер отчета от 100 до 200%. Это облегчит последующее редактирование:



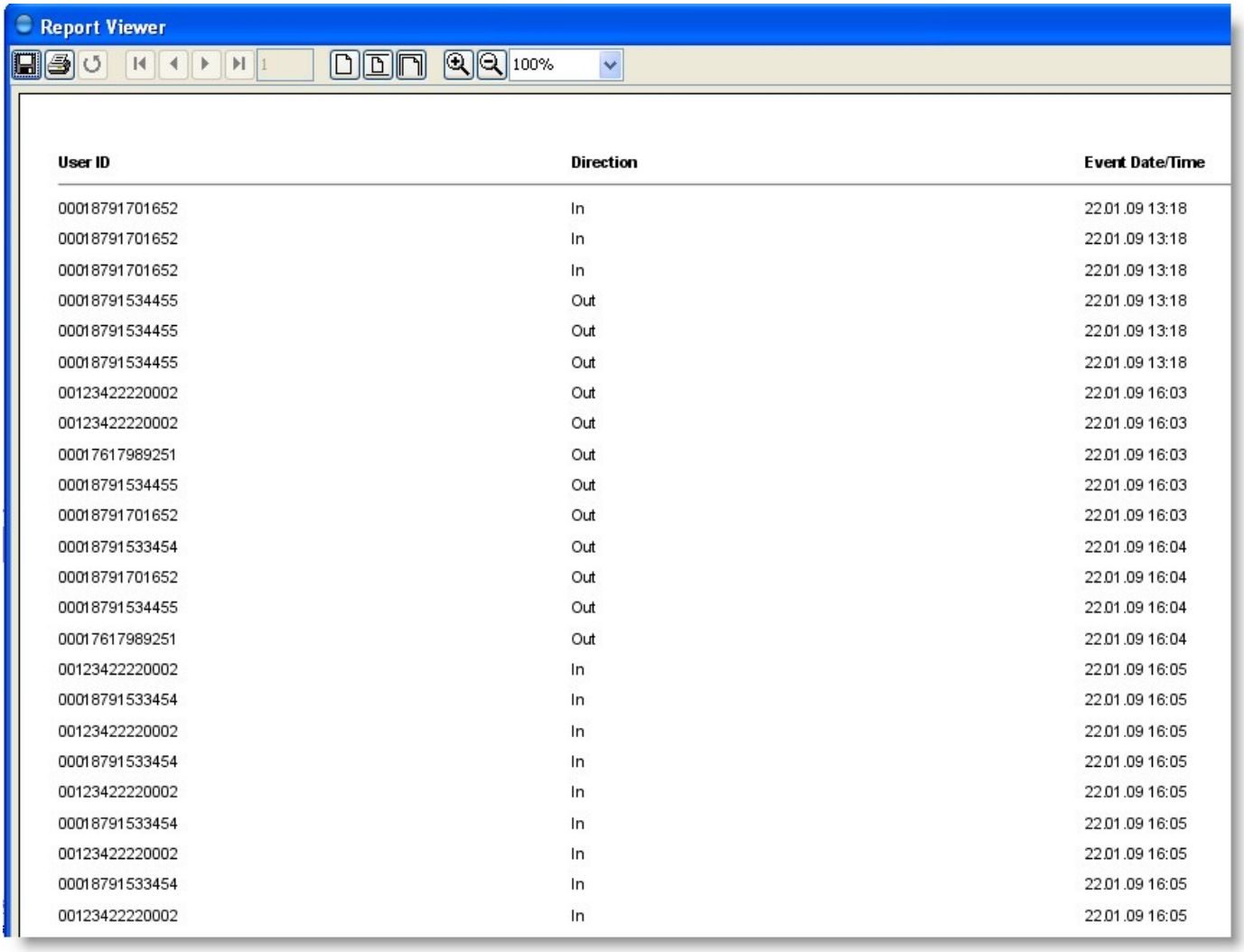
А теперь дважды кликните по заголовку первой колонки и измените его `User ID`. Смените аналогичным образом заголовок третьей колонки на `Event Date/Time`.

А теперь выберите **File > Quit,** чтобы выйти из Редактора отчетов. Подтвердите сохранение изменений шаблона отчетов.

Новый узел **Отчет** ( ) будет видимым в Системном дереве:

4. Просмотр отчета

Чтобы заполнить отчет последними данными и просмотреть его, правой кнопкой мыши щелкните по нему и выберите **Показать отчет** из контекстного меню Показать отчет. Ваш отчет откроется в Просмотрщике отчетов, и Вы сможете осуществлять поиск, экспортировать и печатать отчеты:



User ID	Direction	Event Date/Time
00018791701652	In	22.01.09 13:18
00018791701652	In	22.01.09 13:18
00018791701652	In	22.01.09 13:18
00018791534455	Out	22.01.09 13:18
00018791534455	Out	22.01.09 13:18
00018791534455	Out	22.01.09 13:18
00123422220002	Out	22.01.09 16:03
00123422220002	Out	22.01.09 16:03
00017617989251	Out	22.01.09 16:03
00018791534455	Out	22.01.09 16:03
00018791701652	Out	22.01.09 16:03
00018791533454	Out	22.01.09 16:04
00018791701652	Out	22.01.09 16:04
00018791534455	Out	22.01.09 16:04
00017617989251	Out	22.01.09 16:04
00123422220002	In	22.01.09 16:05
00018791533454	In	22.01.09 16:05
00123422220002	In	22.01.09 16:05
00018791533454	In	22.01.09 16:05
00123422220002	In	22.01.09 16:05
00018791533454	In	22.01.09 16:05
00123422220002	In	22.01.09 16:05
00018791533454	In	22.01.09 16:05
00123422220002	In	22.01.09 16:05

Вы можете также просто дважды щелкнуть по узлу отчета, чтобы запустить действие Показать отчет, поскольку это действие по умолчанию 88 для контекста отчета 1553.

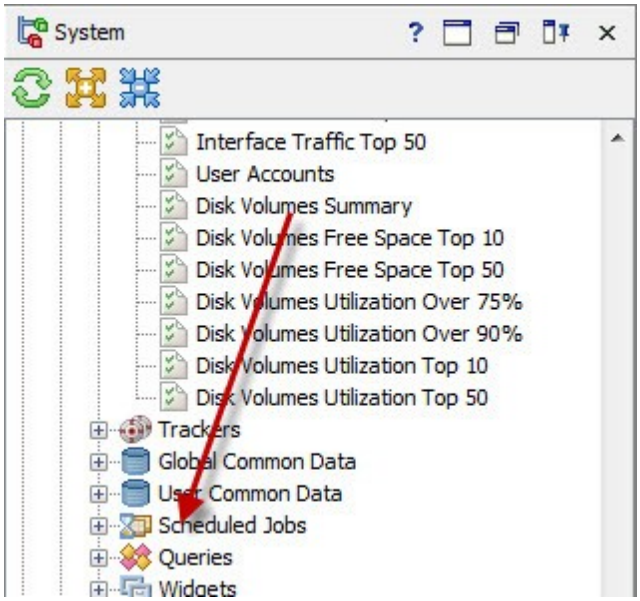
18.11 Запланированная отправка отчета по email

Часто оказывается необходимым отправить отчет по e-mail. Например, руководители компании могут пожелать получать ежемесячно статистику об ошибках/поломках в IT-оборудовании. Администратор AtomMind может отправлять отчеты по e-mail вручную, но гораздо удобнее и эффективнее назначить периодическую отправку отчета при помощи планировщика задач 823 AtomMind Server. В этом уроке объясняется, как создать запланированную задачу, выполняющую отправку отчета по email.

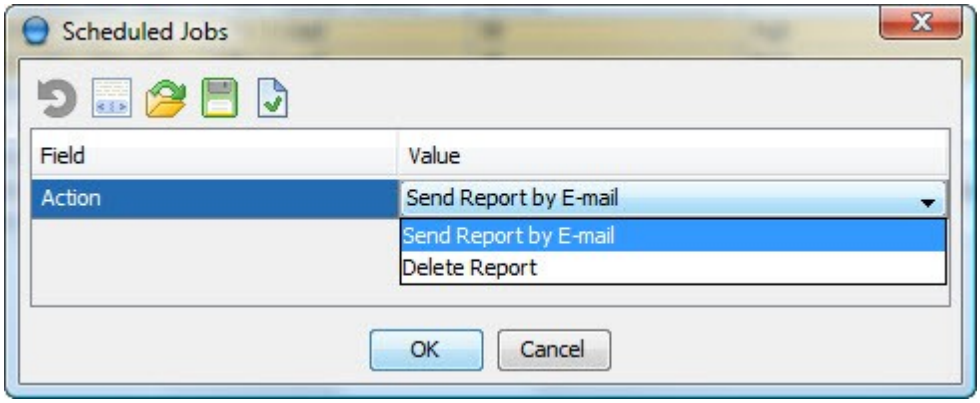
Чтобы отправить отчет вручную, выполните действие Отправить отчет по e-mail в контексте отчета 1553.

1. Добавление запланированной задачи

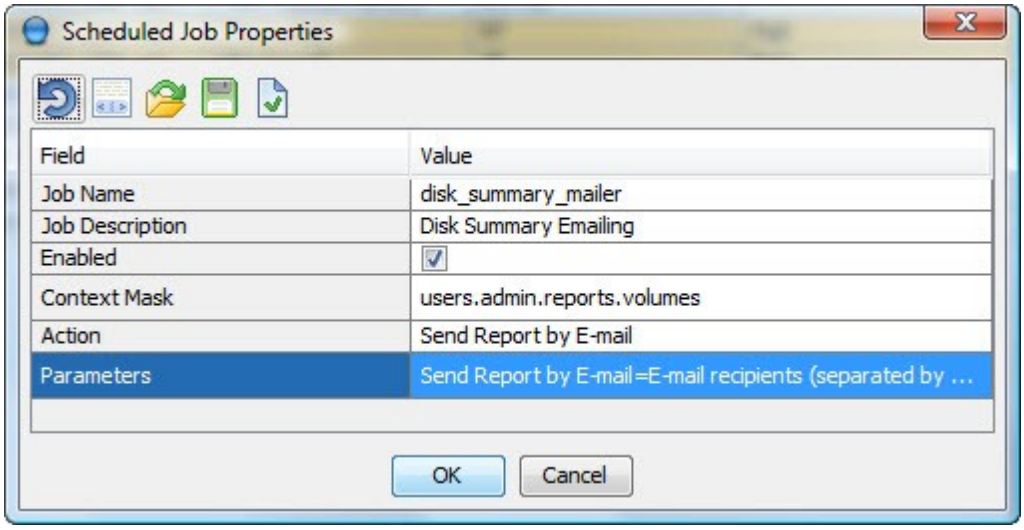
Чтобы создать новую запланированную задачу, найдите узел Вашего отчета в системном дереве 370 ( ). Перетащите его на узел Запланированные задачи ( ):



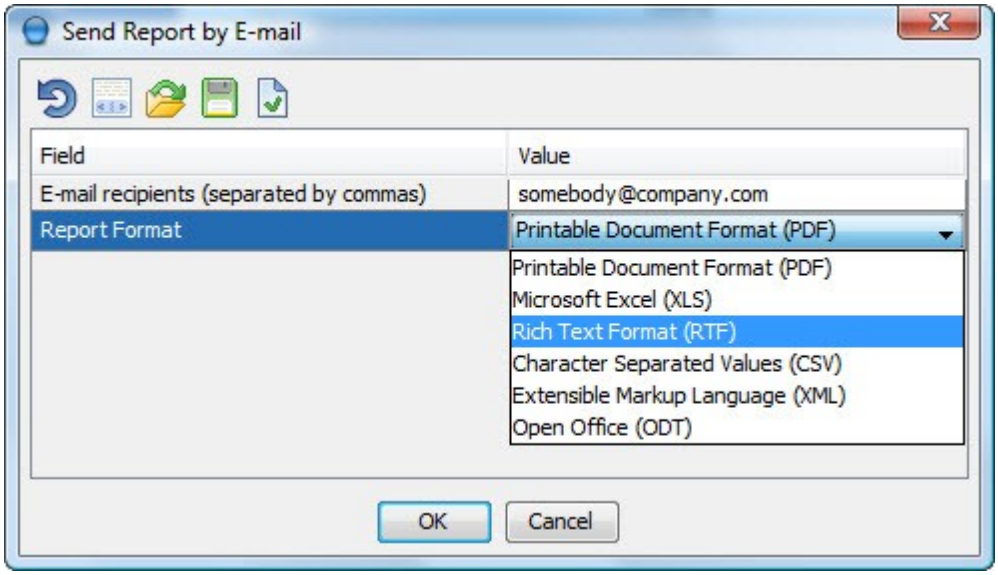
Выберите **Отправить отчет по email** из списка действий, которые могут быть запланированы:



Кликните **OK**. Теперь определите свойства для нового планировщика задач. Введите имя задачи и ее описание (например, **disk_summary_mailer** и **Disk Summary Emailing**):



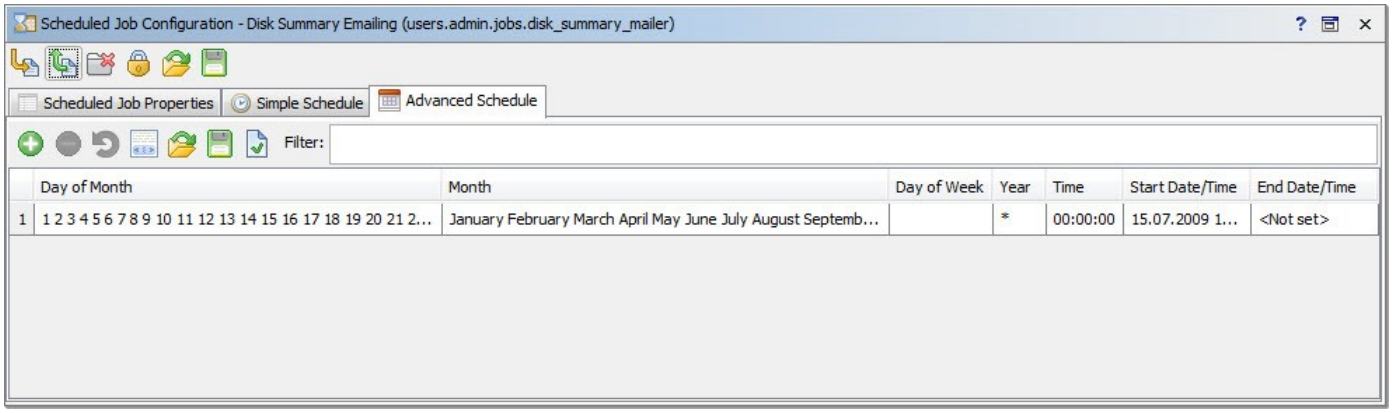
Кликните по строке **Параметры**, а затем кликните по строке **Отправить отчет по email**, чтобы открыть параметры отправки отчета по электронной почте. Введите получателей e-mail и выберите формат отчета:



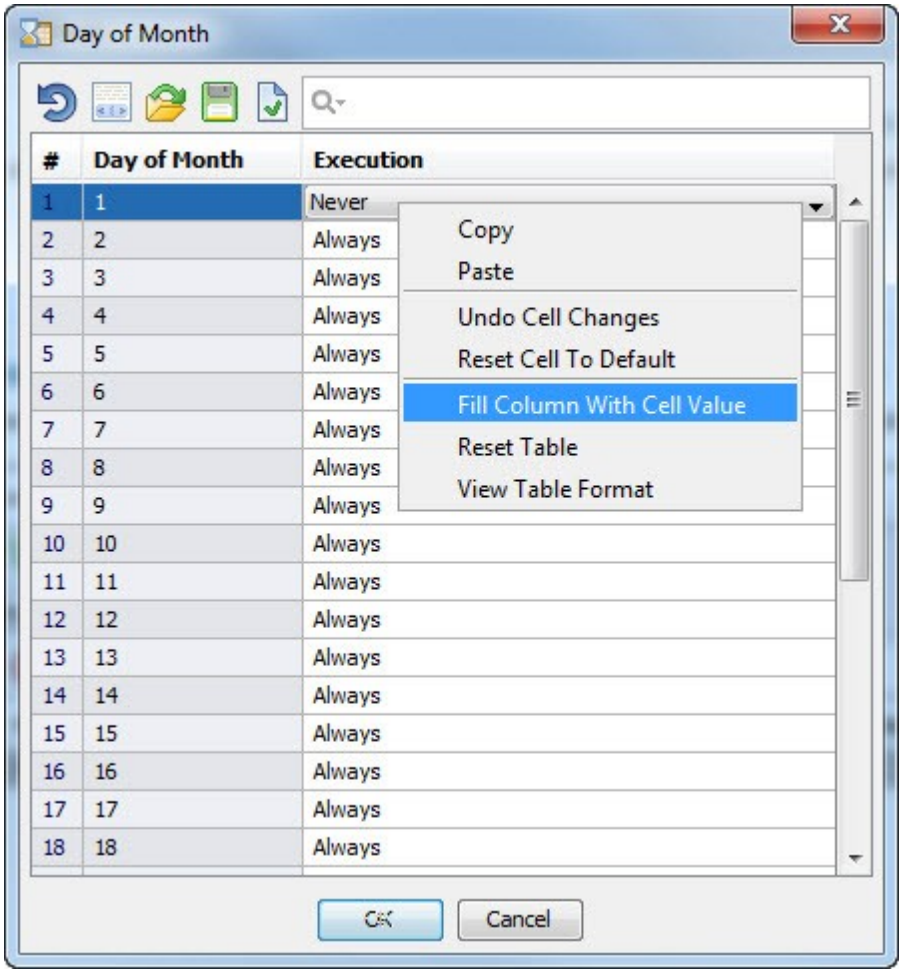
Кликните **OK,** чтобы создать новую запланированную задачу. Ее свойства отроются в новом плавающем окне.

2. Составление графика отправки

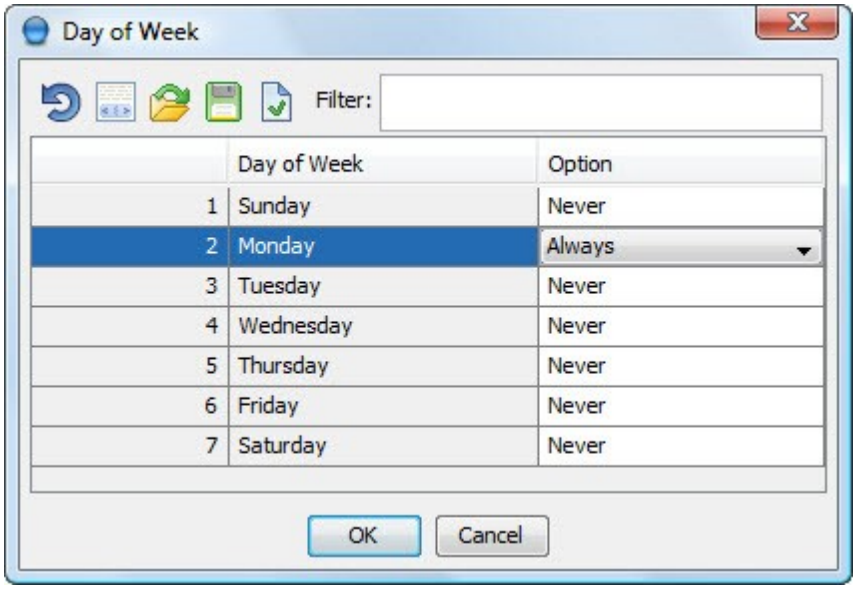
А теперь пора добавить триггеры задач, которые будут запускаться в определенное время. В окне свойств задач откройте вкладку Дополнительный график ( ). Кликните по кнопке Добавить ряд ( ), чтобы добавить новый триггер:



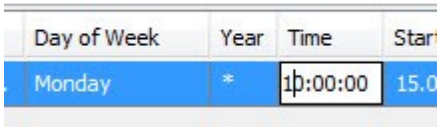
Давайте настроим задачу, которая бы запускалась каждый понедельник в 10:00 AM. Для этого нам необходимо переключить триггер с работы по определенным дням в месяце на определенные дни в неделе. Во-первых, кликните в поле **День месяца**. День за днем ежемесячный график будет открывать новое окно. Выберите для первого дня **Никогда,** правой кнопкой мыши кликните в той же ячейке и выберите из контекстного меню **Заполнить колонку значением из ячейки:**



Кликните **OK**, чтобы закрыть диалоговое окно. А теперь кликните по полю **День недели.** В новом открывшемся окне установите **Опции** для понедельника на **Всегда**:



Кликните **OK**, чтобы применить изменения. Последнее действие устанавливает параметр **Время** триггера задач на 10:00:00:



Когда действие выполнено, кликните **Сохранить свойства** (). Запланированная задача теперь готова, а отчет будет отправлен в следующий понедельник.

18.12 Добавление дополнительных свойств

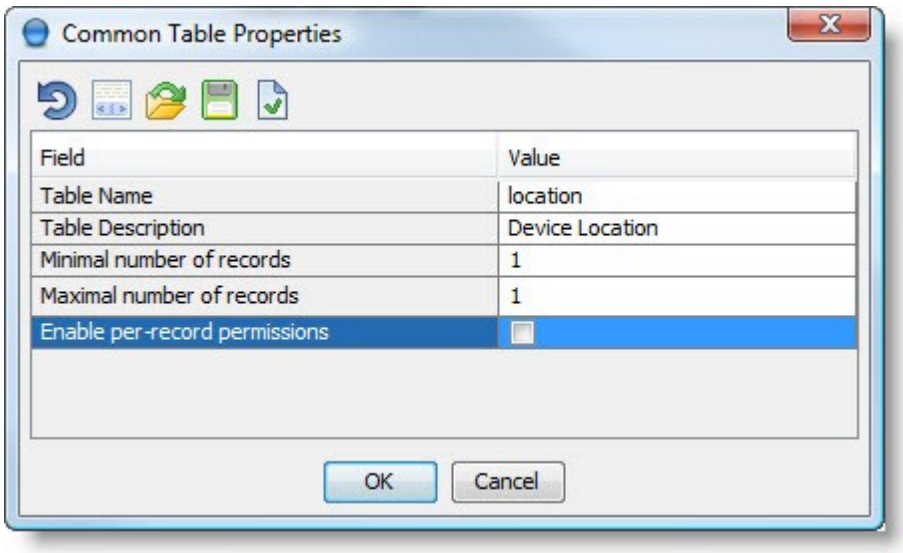
У каждого Device есть ряд стандартных свойств, которые осуществляют контроль за подключением к аппаратному оборудованию и обработкой данных на сервере, например, за **Периодом синхронизации (Synchronization Period)** или **Использованием расширенного статуса (Use Extended Status)**. Однако может быть необходимым ассоциировать некоторую нестандартную информацию с Device. Например, задачи по управлению IT-инфраструктурой требуют легкого обнаружения текущего месторасположения Device, т.е. здания, этажа и номера офиса. AtomMind разрешает добавлять любые пользовательские данные в учетную запись устройства или системные ресурсы, используя свойство общих данных 217. В этом уроке показано, как добавить информацию о Месторасположении устройства в каждое устройство при помощи AtomMind Client 359.

1. Создание общей таблицы

Во-первых, нам необходимо создать общую таблицу, которая будет описывать формат наших пользовательских свойств. Копия данной таблицы позднее будет добавлена в каждую учетную запись устройства.

Дважды кликните по узлу Глобальные общие данные (), чтобы запустить создание таблицы. Введите свойства таблицы следующим образом:

- Имя таблицы: `location`
- Описание таблицы: `Device Location`
- Минимальное количество записей: `1` (поскольку мы добавляем не табличное свойство)
- Максимальное количество записей: `1`



Кликните OK, чтобы добавить новую общую таблицу. Она появится в системном дереве 370:

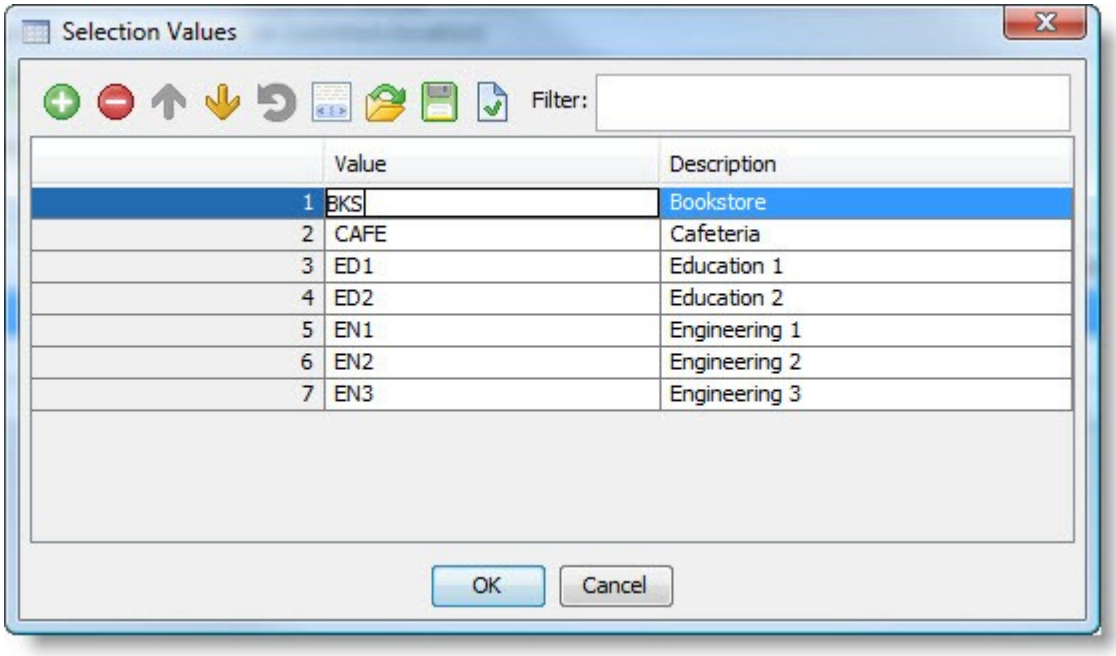
2. Добавление полей месторасположения в общую таблицу

Если диалоговое окно конфигурации таблицы не всплывает автоматически после создания таблицы, кликните правой кнопкой мыши по только что созданной таблице и выберите **Настроить общую таблицу** (). Переключитесь во вкладку **Поля**, чтобы приступить к добавлению полей таблицы.

Кликните по **Добавить ряд** () и введите свойства поля:

- Имя: `building`
- Тип: `String`
- Описание: `Building`
- Значения расширенного выбора: `Yes`

Кликните по **Значения выбора** и введите несколько зданий. Ниже следует пример:



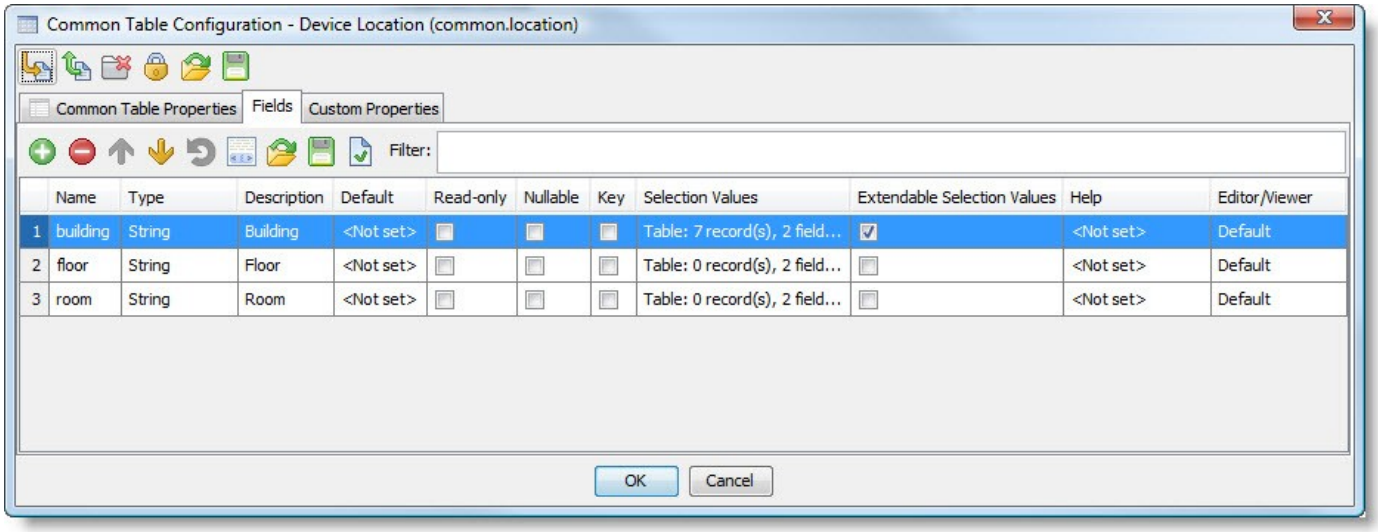
Оставьте другие значения по умолчанию и добавьте два других поля:

- Имя: `floor`
- Тип: `String`
- Описание: `Floor`

и

- Имя: `room`
- Тип: `String`
- Описание: `Room`

Таблица Поля должна выглядеть следующим образом:



3. Назначение таблицы месторасположения устройства как пользовательского свойства

А теперь пора сообщить AtomMind Server, что наша новая таблица - дополнительное свойство устройства. Переключитесь во вкладку Дополнительные свойства и выберите **Использовать общую таблицу как дополнительное свойство**.

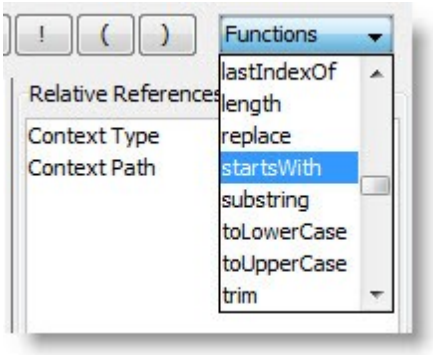
На этом этапе поле **Выражение пригодности** станет доступным для редактирования. Оно определяет[2177], какие контексты[41] следует ассоциировать с нашей таблицей. Кликните по кнопке **[...]** внутри поля Выражения пригодности, чтобы открыть редактор выражений[404].

4. Написание выражения пригодности

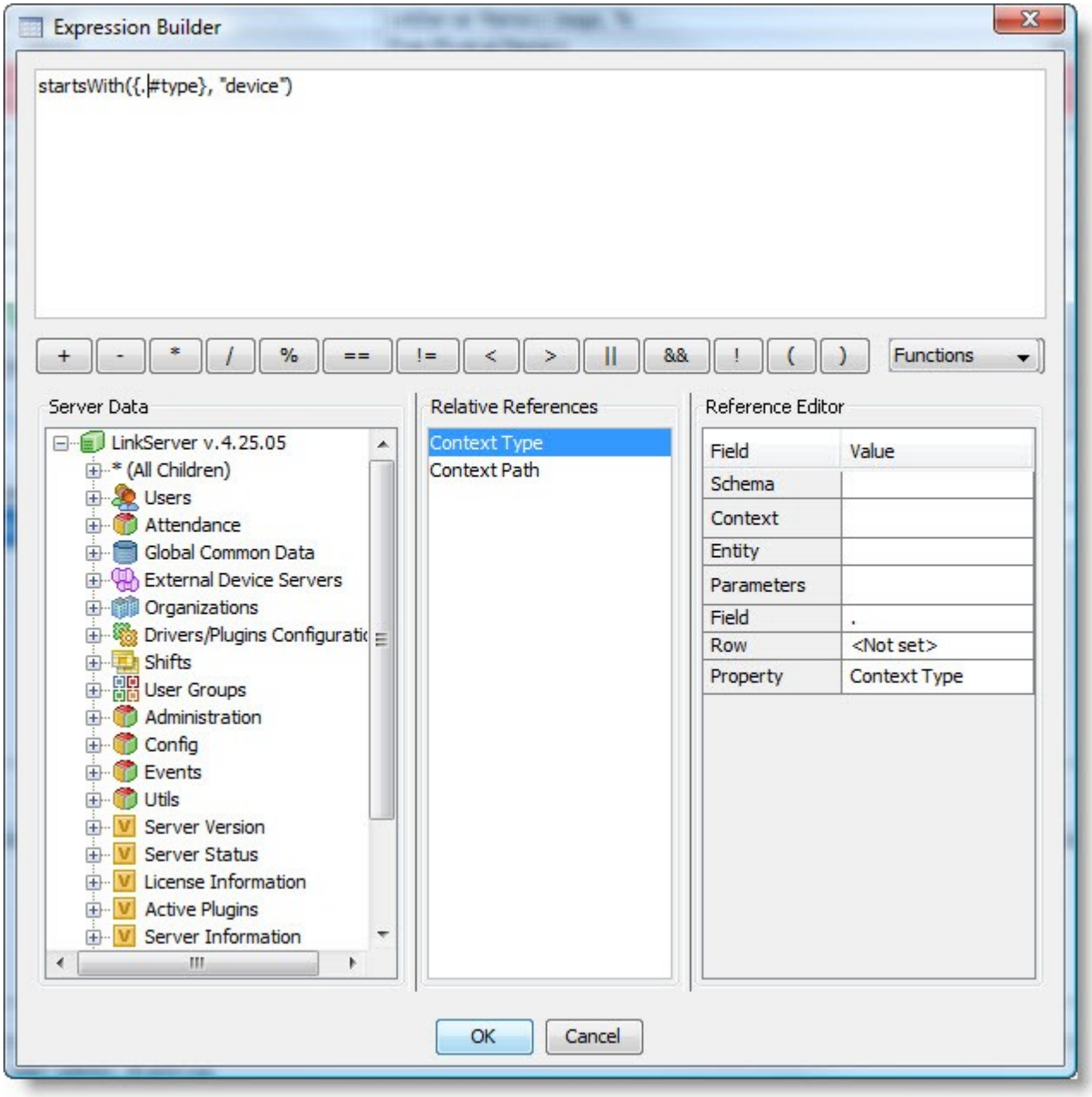
В нашем случае выражение пригодности следует назначить как верное (TRUE) для любого контекста устройства и неверное (FALSE) для всех остальных контекстов. Раздел контекст устройства[1494] в ссылке контекста[1450] указывает, что тип контекста для контекстов устройства имеет следующую форму:

```
device.DEVICE_TYPE
```

Это означает, что любой тип контекста, *начинающийся со слова "устройство",* соответствует контексту устройства. Выберите функцию **startsWith** из раскрывающегося списка **Функций**, чтобы вставить ее в выражение:



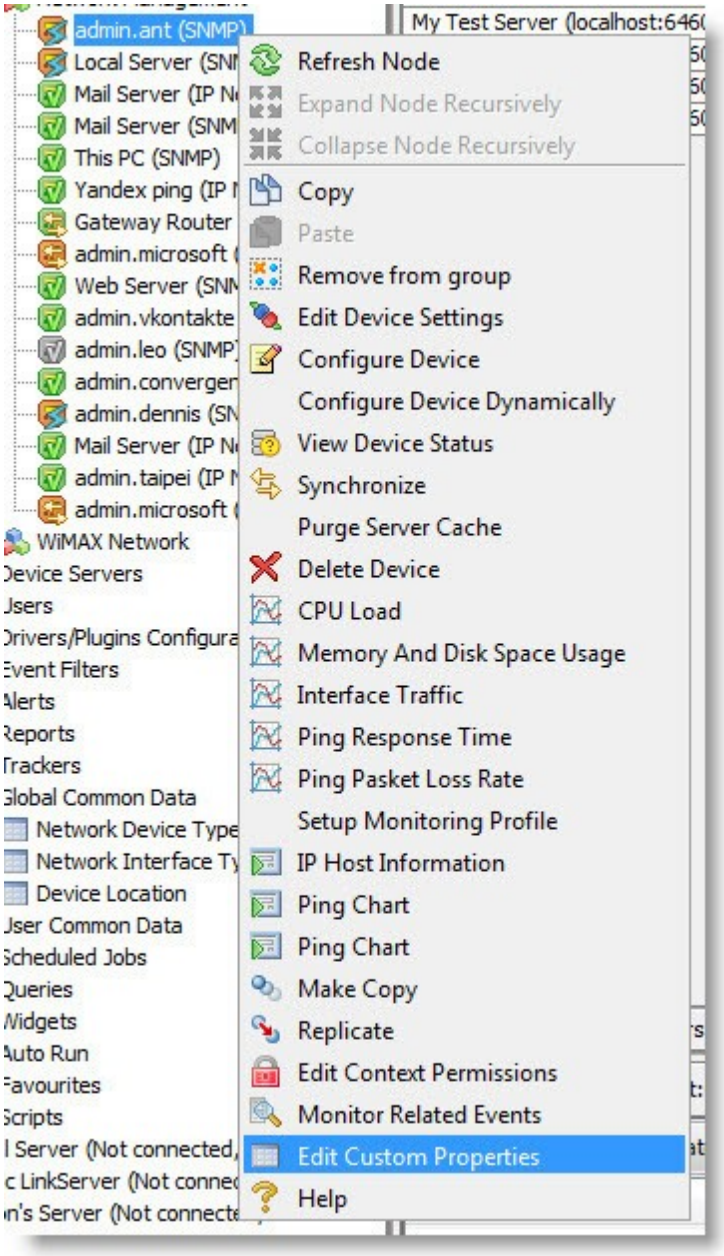
Согласно описанию[124] данной функции у нее есть два параметра строки; она возвращает TRUE, если первая строка начинается со второй строки ("префикса"). Кликните параметры внутри функции и дважды кликните по **Типу контекста** в области **Относительных ссылок**, чтобы добавить тип контекста как первый параметр. Затем вручную добавьте буквенную строку "device" аналогично второму параметру:



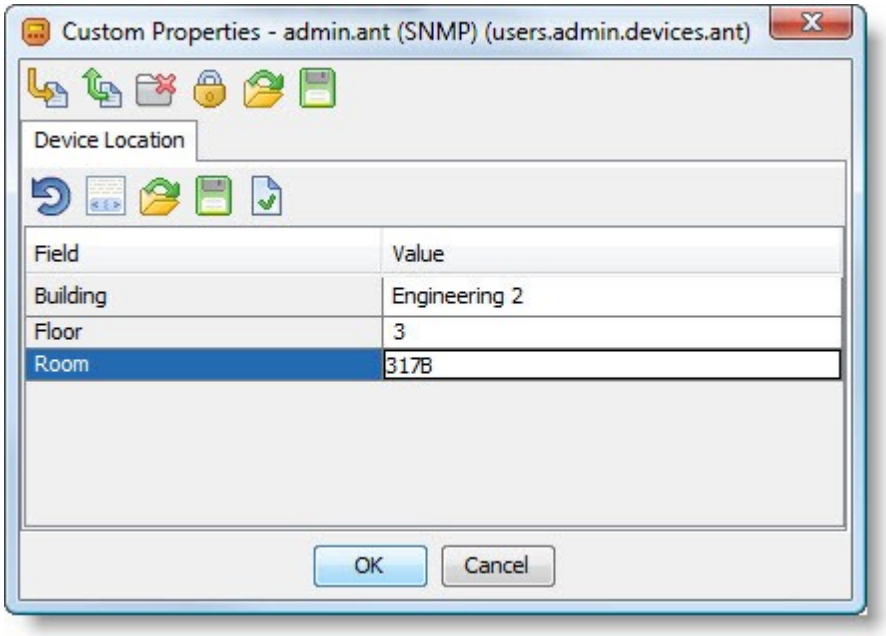
Кликните OK дважды, чтобы применить изменения в настройке Выражения пригодности и Общей таблицы. Теперь каждое устройство в системе имеет свое собственное свойство Месторасположения.

5. Редактирование месторасположения устройства

Чтобы получить доступ к свойству Месторасположение устройства определенного устройства, щелкните по нему правой кнопкой мыши и выберите **Редактировать дополнительные свойства.**



Теперь можно заполнить вкладку **Местоположение устройства**:



18.13 Использование шаблонов ресурсов и устройств

Часто может оказаться необходимым использовать одну и ту же настройку для нескольких устройств или системных ресурсов. В AtomMind существует два метода создавать аналогичные конфигурации для различных объектов:

- репликация 110 конфигурации
- копирование 109 объектов

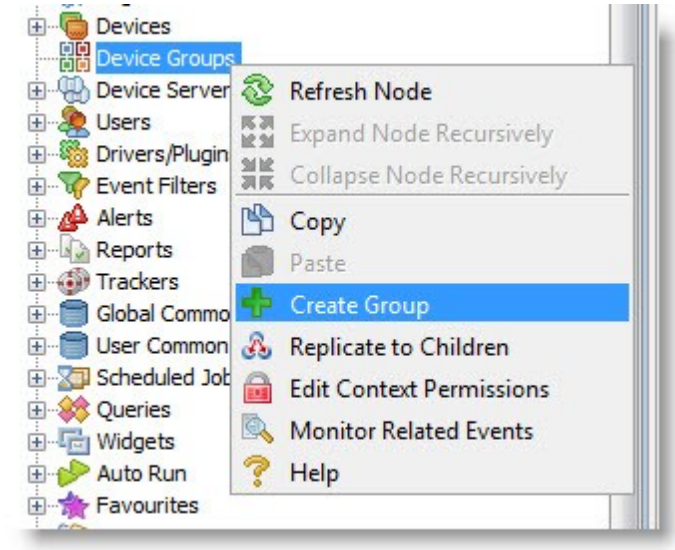
Иногда Вам может потребоваться использовать несколько устройств или ресурсов в качестве источников копирования. Эти ресурсы деактивируются и не выполняют действия внутри системы (например, деактивированная тревога никогда не будет вызвана). Такие ресурсы называются *шаблонами.*

В этом уроке рассказывается, как создать шаблоны устройства для быстрой настройки устройств с аналогичной конфигурацией. Аналогичный метод можно использовать для создания шаблонов для любого другого типа ресурсов.

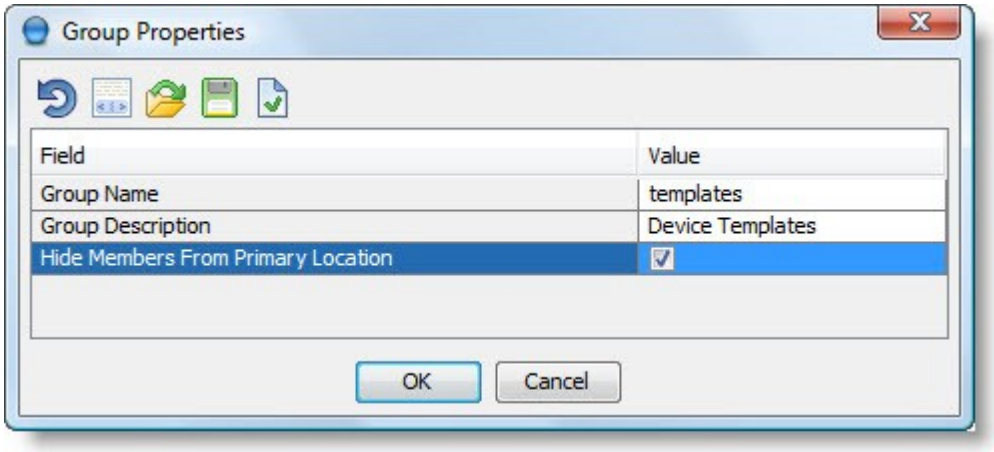
1. Создание группы для хранения шаблонов

Во-первых, давайте создадим группу устройств для хранения наших шаблонов. Этот этап необязательный, но может оказаться полезным для отделения активных устройств от шаблонов.

В системном дереве 370 AtomMind Clienta щелкните правой кнопкой мыши по элементу **Группы устройства** ( ) и выберите **Создать группу** ( ):



Введите **Имя** группы и ее **Описание**. Выберите **Скрыть членов от первичного месторасположения**, чтобы отделить шаблоны от обычных устройств:

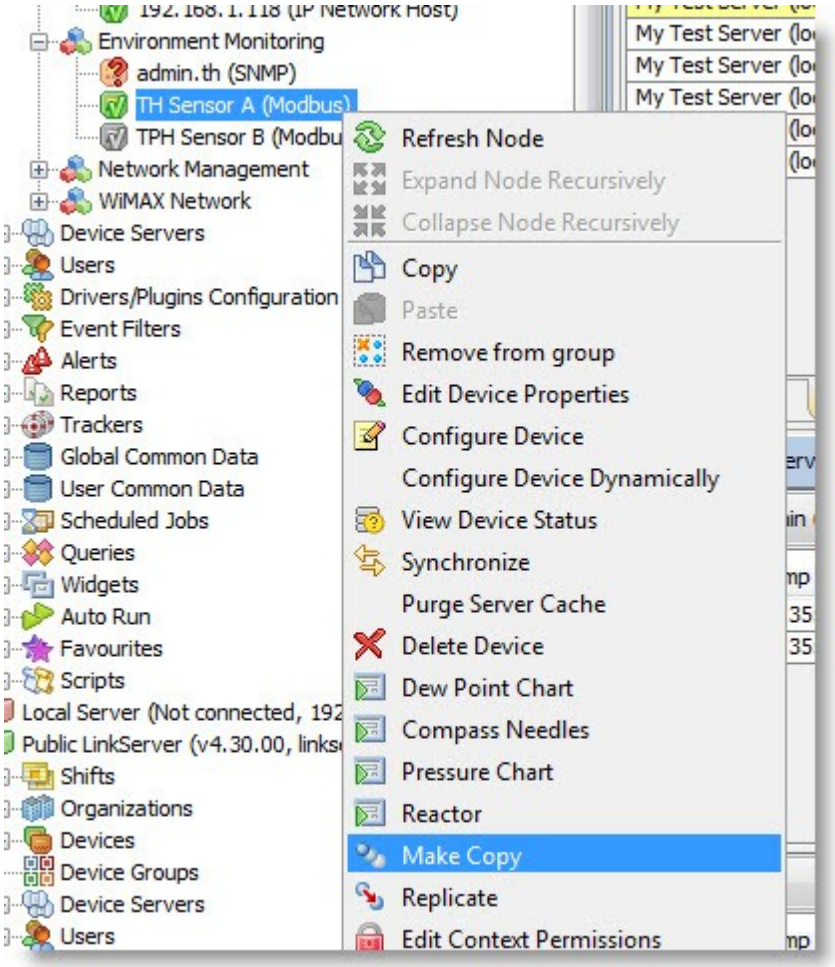


Кликните OK, чтобы создать новую группу.

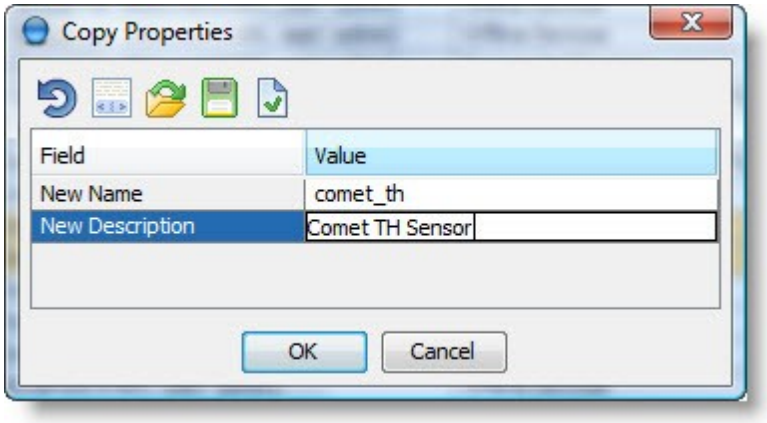
2. Создание шаблона

Нам нужно клонировать устройство, чтобы использовать его копию в качестве шаблона.

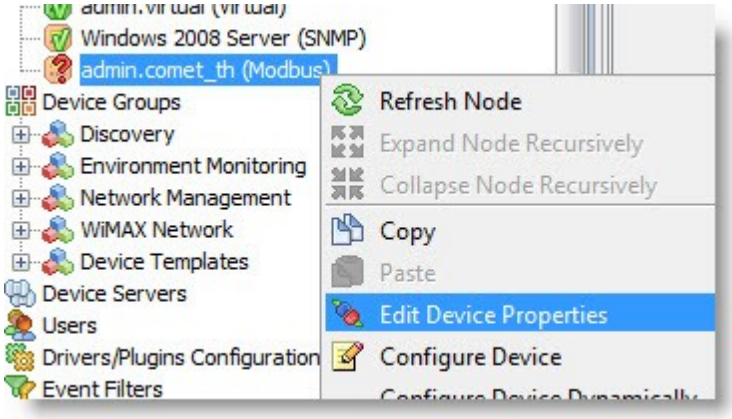
Правой кнопкой мыши щелкните по устройству ( ), которое Вы хотите использовать в качестве шаблона, и выберите **Сделать копию** ( ):



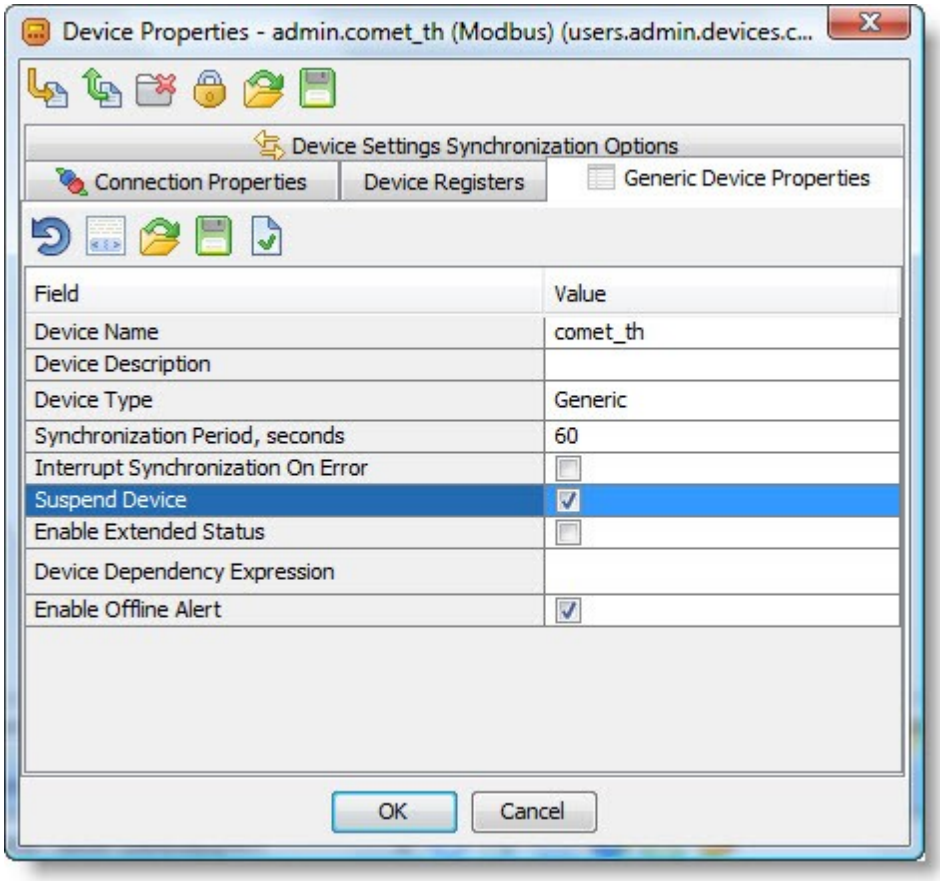
Введите имя и описание для шаблона:



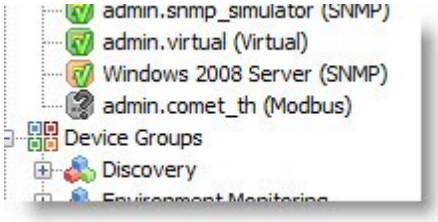
Кликните OK, чтобы продолжить. А теперь у нас есть только что созданное нормальное (активное) устройство. Давайте преобразуем его в шаблон, временно приостановив его. Правой кнопкой мыши кликните по устройству ( ) и выберите Редактировать свойства устройства ( ):



Переключитесь во вкладку **Общие свойства устройства** ( ) и выберите **Временно приостановить устройство**:



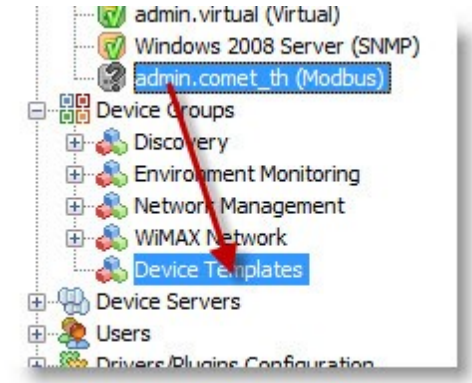
Кликните OK, чтобы временно приостановить устройство. Его иконка станет серой, что означает, что устройство больше не активно:



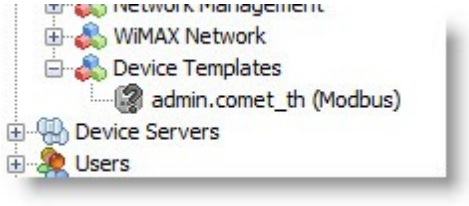
Методы деактивации системного источника, когда оно преобразуется в шаблон, могут быть разными. Тревоги 779, датчики 2181 и большинство других "активных" ресурсов имеют свойство **Включен**. Оно должно быть отключено, когда ресурс используется в качестве шаблона. Неактивные ресурсы, т.е. ресурсы, не выполняющие задачу (такие как отчеты 928), не требуют деактивации.

Наш шаблон - это просто нормальное, временно приостановленное устройство для AtomMind. Еще один повод для временной приостановки устройства (когда он не используется в качестве шаблона) - это временно остановить все операции определенного элемента ПО.

А теперь перетащите его в группу **Шаблоны устройств** ( ):



Шаблон будет добавлен в группу:



3. Использование шаблона

Чтобы создать новое устройство, основанное на шаблоне, нам необходимо клонировать сам шаблон:

- Правой кнопкой мыши кликните по нему и выберите **Сделать копию** ( )
- Введите имя и описание для создаваемого устройства
- Кликните OK, чтобы создать устройство

В конце необходимо активировать новое устройство:

- Щелкните правой кнопкой мыши по устройству и выберите элемент **Редактировать свойства устройства** ( )
- Переключитесь во вкладку **Общие свойства устройства** ( ) и выберите **Временно приостановить устройство**
- Кликните OK, чтобы сохранить свойства

Новое устройство, основанное на шаблоне, теперь будет работать нормально:

Еще один метод использования шаблона - перетащить его в контекст любого контейнера и выполнить действие создать из шаблона 105.

18.14 Сохранение истории изменений объекта

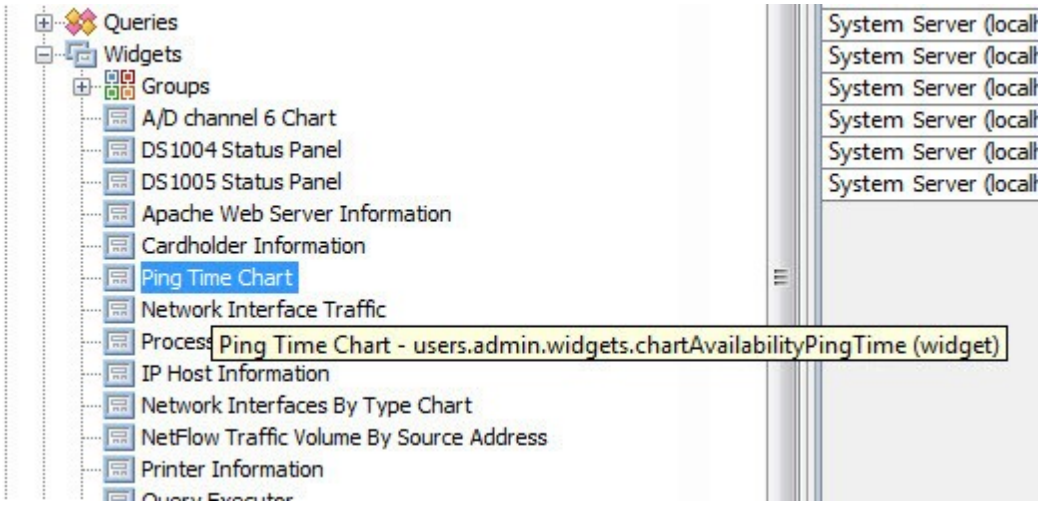
В то время как для базы данных AtomMind следует периодически выполнять резервное копирование 173, также может оказаться полезным хранить историю о важных изменениях системных объектов. Например, если у отдельного виджета 943 есть сложный шаблон, может оказаться полезным отслеживать каждое изменение, что позволяет определить и отменить ошибочные модификации.

В этом кратком уроке объясняется, как активировать и просмотреть исторические изменения для любого системного объекта или группы объектов.

1. Обнаружение пути контекста объекта

На первом этапе необходимо определить путь контекста 42 или контекстной маски 44 объектов, история которых должна быть сохранена. Переведите мышь на узел объекта в системном дереве 370, чтобы просмотреть путь контекста во всплывающей подсказке.

Путь контекста виджета Диаграммы времени ping (см рис ниже) - **users.admin.widgets.chartAvailabilityPingHelper**.

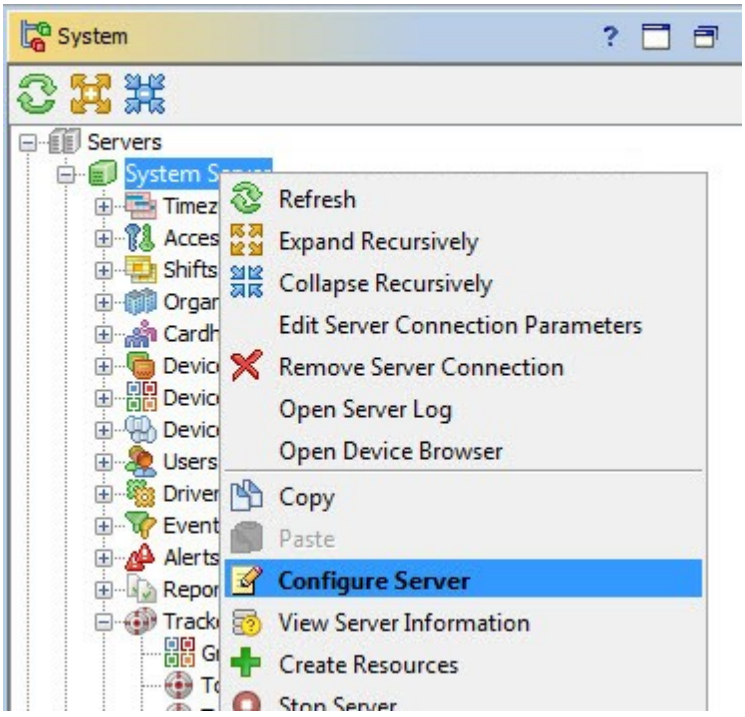


Правой кнопкой мыши кликните по узлу виджета и выберите **Копировать** ( ) из контекстного меню, чтобы скопировать путь контекста в область обмена данными.

2. Включение хранения истории изменений

Каждый раз при изменении на пользовательском уровне определенного свойства любого системного объекта, AtomMind Server создает событие об изменении 84. Нам необходимо включить хранение исторических значений для данного события, чтобы иметь доступ к истории изменений свойств объекта.

Типичный способ включить постоянное хранение для событий AtomMind любого типа - использовать глобальную таблицу времени окончания действия события. Чтобы получить доступ к этой таблице, кликните правой кнопкой мыши по узлу AtomMind Server ( ) в Системном дереве и выберите **Настроить сервер**:



Переключитесь во вкладку **Время окончания действия события** и кликните по иконке **Добавить ряд** ( ), чтобы добавить новую запись, а затем:

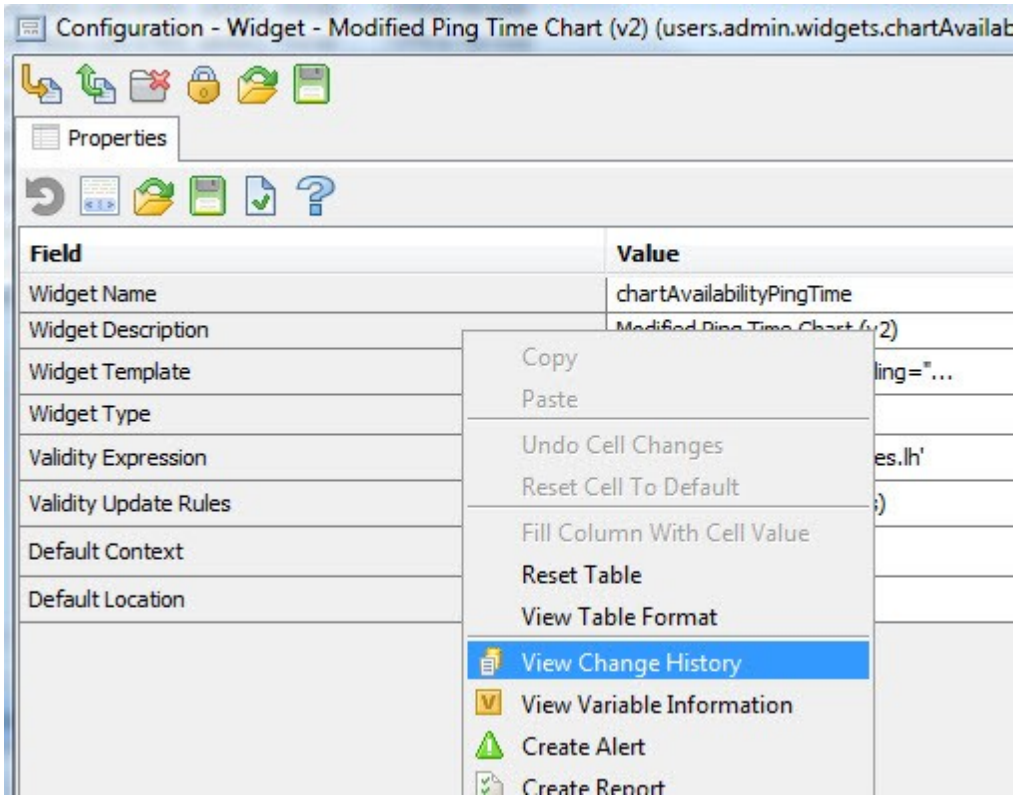
- Вставьте ранее скопированный путь контекста в поле Контекстной маски или введите его вручную
- В поле **Имя события** введите change
- Выберите желаемый период хранения истории изменений и поле **Период истечения срока**

Используйте контекстную маску вместо пути, чтобы активировать хранилище истории событий для множества объектов. Например, **users.*.widgets.*** активирует хранение для всех виджетов, которые принадлежат всем пользователям 478 системы.

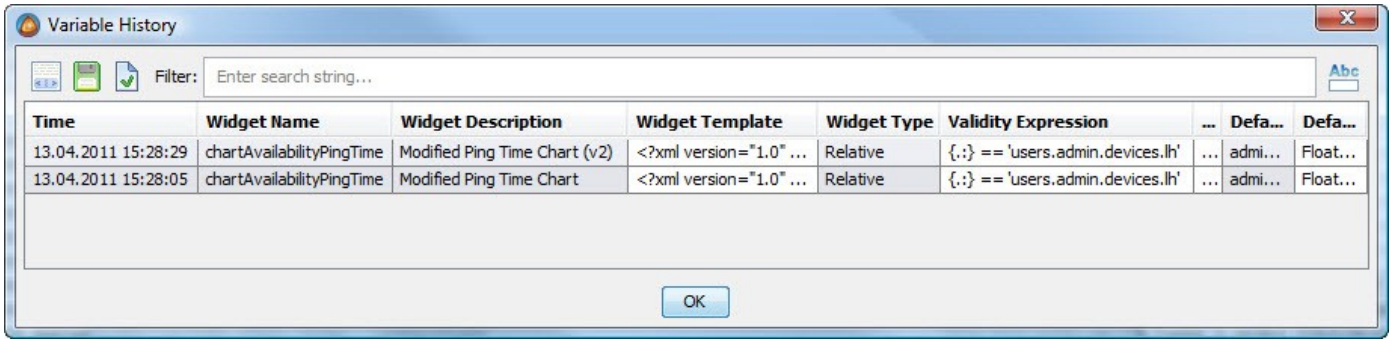
Кликните OK, чтобы закончить редактирование глобальной настройки. Вам не нужно перезапускать сервер после этого изменения.

3. Просмотр истории изменений

После того как история изменений активирована, можно как обычно редактировать свойства Вашего объекта. Чтобы получить доступ к истории изменений, правой кнопкой мыши кликните по свойству в редакторе свойств 377 и выберите Просмотреть историю переменных ( ):



История откроется в новом окне:



18.15 Планирование времени простоя для устройства

Устройство в назначенный период времени должно периодически подвергаться профилактическому осмотру/ремонту. Это означает, что устройство не будет доступно для AtomMind во время данного периода, и оно не будет синхронизировано 514 с сервером. Система не должна генерировать предупреждение о недоступности устройства во время назначенного времени простоя.

В этом уроке показано, как использовать выражение зависимости 512 устройства, чтобы временно отключить синхронизацию устройства в определенное время.

Хотя выражения зависимости чаще всего используются для проверки статуса другого устройства и отключения синхронизации, если указанное устройство опущено, их также можно использовать и для проверки текущего времени.

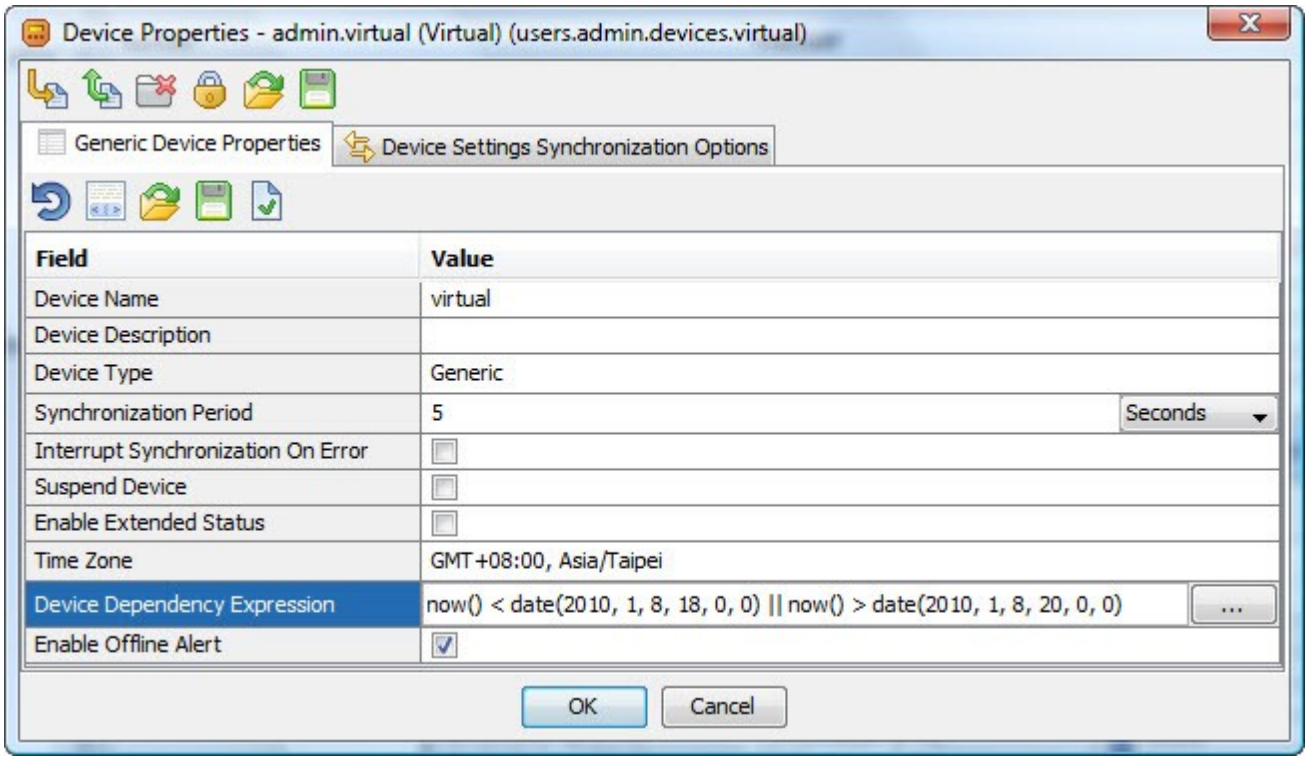
Назначение времени простоя в абсолютном времени

В этом случае мы установим выражение зависимости, которое разрешается в FALSE за определенный интервал времени или в TRUE в ином случае. Мы будем использовать функцию now() , которая возвращает текущую временную метку (т.е. время проверки зависимости) и функцию date() , которая создает временную метку из аргументов года, месяца, дня, часов и минут. Описания этих функций доступны в здесь 124. Обратите внимание, что аргумент месяца функции даты () основан на нуле, поэтому следует присвоить январю 0.

Давайте назначим время простоя для устройства на 08 фев 2010 с 18:00 до 20:00. Наше выражение зависимости выглядит следующим образом:

```
now() < date(2010, 1, 8, 18, 0, 0) || now() > date(2010, 1, 8, 20, 0, 0)
```

Свойства учетной записи устройства в AtomMind Client:



Выражение, которое назначает для времени простоя целый день:

```
now() < date(2010, 1, 8, 0, 0, 0) || now() > date(2010, 1, 9, 0, 0, 0)
```

Назначение периодического времени простоя

Этот же метод можно использовать для назначения периодического времени простоя. В этом случае выражение зависимости будет использовать другие функции обработки данных/времени для установления вывода текущей функции ().

Ниже приводится выражение зависимости, которое отключает синхронизацию устройства каждый понедельник:

```
dayOfWeek(now()) != 1
```

Обратите внимание, что функция dayOfWeek() возвращает нуль для воскресенья. Используется оператор "Не равен" ("Not equals"), поскольку выражение зависимости должно возвращать TRUE (чтобы включить синхронизацию), когда текущий день не понедельник.

Чтобы назначить время простоя для каждой среды с 22:00 до 23:00:

```
!(dayOfWeek(now()) == 3 && hour(now()) > 22 && hour(now()) < 23)
```

Альтернативный метод назначения времени простоя для устройства - использовать планировщик задач 823, чтобы активировать/деактивировать свойство Временно отключить устройство 511 учетной записи устройства в определенное время.

18.16 Выбор и обработка событий

Часто бывает необходимым найти, отфильтровать и агрегировать определенное количество событий[73] устройства. Это может оказаться полезным для:

- создания отчета[928]
- создания и заполнения общей таблицы[2176]
- обработки событий с использованием скрипта[879]
- просмотра данных события вручную

Общий знаменатель всех перечисленных задач заключается в том, что нам следует извлечь выбранные исторические данные из базы данных[692] AtomMind Server и преобразовать их в таблицу данных[49]. Эта таблицы данных будут использоваться для заполнения отчета или общей таблицы, заполнения скрипта данными ввода или предоставления администратору.

Есть три разных способа выбора событий и конвертирования их в таблицу данных:

- использование языка выражения[112] AtomMind
- использование запроса[829] AtomMind
- использование устройства записи событий[199] для хранения событий в отдельной таблице и драйвера устройства БД[647] для их выбора с использованием собственного SQL

Ниже представлено краткое сравнение этих методов:

	Язык выражения	**Язык запросов**	**SQL**
Выполнение фильтрации и выбора	высокое	низкое	высокое
Агрегация (общая, средняя, минимальная/максимальная и пр.)	не доступно	доступно	доступно
Потребление памяти	среднее	высокое	низкое
Издержки на хранение событий	нет	нет	удваивает использование диска

Подсказка выбора метода:

- Если Вам необходимо обработать большое количество событий (100000 и более), используйте собственный SQL
- Если количество событий небольшое, но требуется агрегация, используйте язык запросов AtomMind
- Во всех остальных случаях используйте язык выражений

Выбор событий с использованием языка выражений AtomMind

Чтобы выбрать события при помощи языка выражения, следует использовать функцию получить историю событий[1518] контекста событий. Определите имя/маску контекста, имя события, выражение фильтрации и дату начала в параметрах ввода.

Давайте создадим отчет, показывающий события входа в систему[1606] для пользователя `john`. Выражение данных источника[929] этого отчета будет вызывать функцию Получить историю событий, имеющую другое выражение (используемое для фильтрации событий), заданное в виде строки в списке параметров функции:

Обратите внимание, что последний параметр (Дата начала) опущен, и вместо него используется значение по умолчанию NULL.

```
{events:get("users", "login", "{username} == 'john'")}
```

Выбор событий с использованием языка запросов AtomMind

Вы этом случае мы выбираем сырые данные события, используя ту же самую функцию Получить историю событий (описанную ранее), но выражение фильтрации при этом не задается. Затем мы запускаем Запрос AtomMind в этой таблице для фильтрации и агрегирования данных, используя стандартные операции SQL.

Ниже приводится запрос, который возвращает все события входа в систему пользователя `john` (см. выше):

```
SELECT
  *
```

```
FROM
 events:get("users", "login") as logins
WHERE
 logins.get$username = 'john'
```

Чтобы данный запрос использовался в качестве отчета Выражение данных источника, используйте следующее выражение:

```
{:executeQuery("SELECT * FROM events:get('users', 'login') as logins WHERE
logins.get$username = 'john'")}
```

Выбор событий с использованием SQL

Что касается настройки, этот метод наиболее сложный, поэтому его следует использовать только в том случае, если другие методы не подходят. Чтобы выбрать события, используя SQL, следует выполнить следующее:

- установите писчик пользовательских событий 199, который будет хранить определенное событие в назначенной таблице БД с полями, предназначенными для событий. Писчик событий автоматически создаст таблицу.
- проверьте структуру таблицы, используя утилиту управления базой данных, чтобы определить, какие поля доступны
- создайте новое устройство 497 и выберите **Базу Данных SQL** в качестве драйвера устройства
- скопируйте устройство БД (Драйвер БД, URL, имя пользователя, пароль и пр.) из глобальной настройки БД AtomMind Server 182.

После этого Вы можете выполнить любой SQL-запрос по таблице пользовательских событий, используя операцию Выполнить запрос 650, предоставленную драйвером устройства БД.

Например, чтобы выбрать события, используя SQL, следует использовать следующий отчет Выражение данных источника:

```
{users.admin.devices.linkserver_database:executeQuery("SELECT * FROM logins WHERE
username = 'john'")}
```

18.17 Создание отчетов о потреблении

Этот урок объясняет, как создать отчет о потреблении на основе агрегированных значений метрик поребления. Такие отчеты суммируют результаты поребления ниже приведенных и подобных им метрик по часам/дням/неделям/месяцам/годам:

- Расход электроэнергии (ватт-час)
- Расход газа (кубические метры)
- Время работы оборудования (часы)

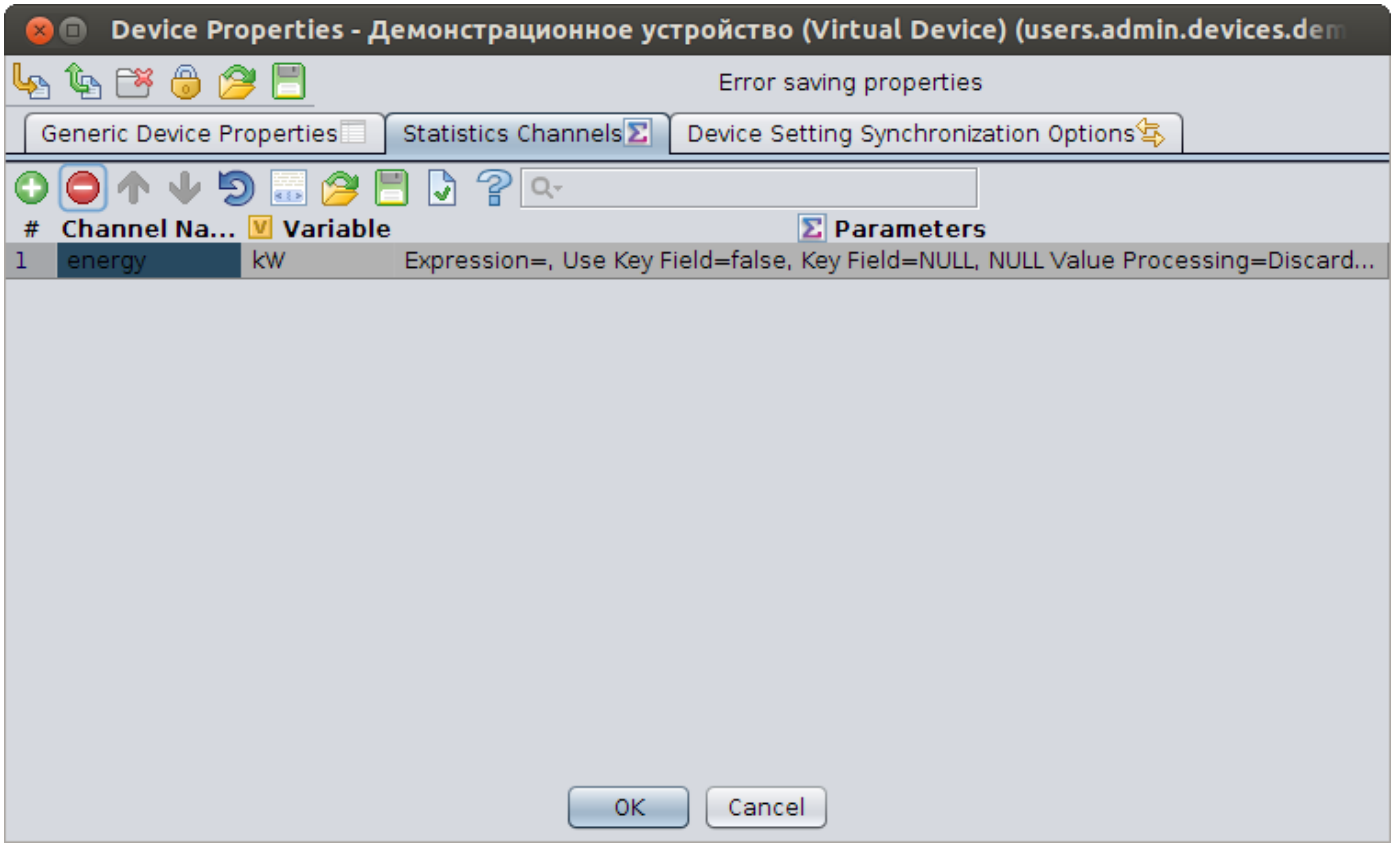
РАСЧЕТ ЭНЕГОПОТРЕБЛЕНИЯ

Предположим, что у нас есть устройство с переменной `kW`, аккумулирующее потребляемую электроэнергию в киловаттах. Эта переменная является счетчиком, который содержит количество киловатт, потребляемых устройством. Необходимо получить информацию о потреблении, сгруппированную по разным периодам: дни, недели, месяцы и годы.

1. Создание статистического канала

Мы будем использовать модуль Контроль статистического процесса 718 для расчета показателей потребления и обобщение их средних значений по часам/дням/неделям/месяцам.

Прежде всего, нам нужно создать статистический канал. Он будет выполнять все расчеты показателей энергопотребления. Для создания статистического канала кликните на всплывающем меню **Редактировать свойства устройств**, перейдите на вкладку **Статистические каналы** и добавьте новый ряд к таблице каналов:

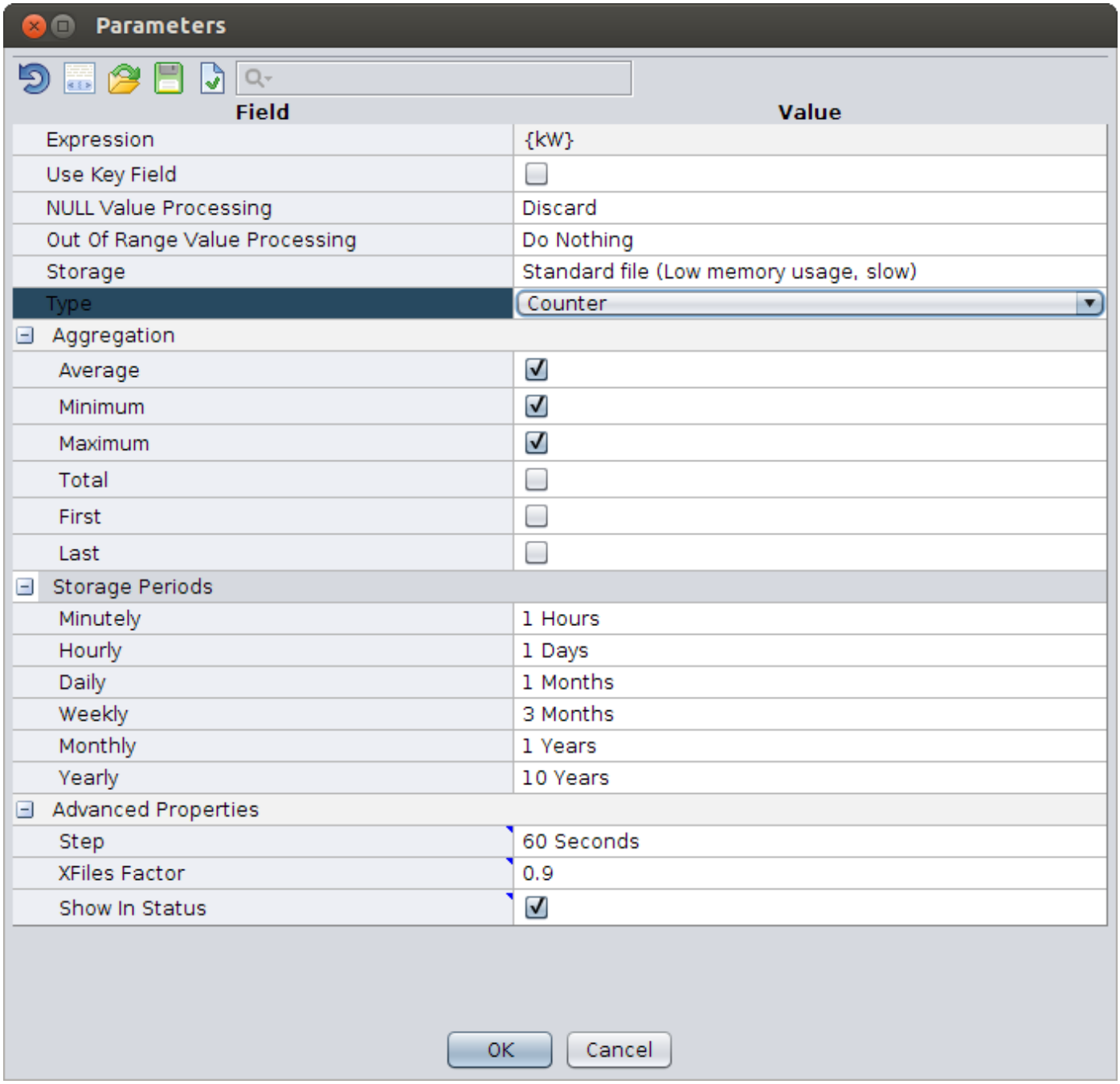


Затем кликните на **Параметрах**. Вы увидите диалог параметров. Введите ссылку[117] на желаемое поле как **Выражение**. Возможно, ваша перменная `kW` будет иметь одно поле с тем же именем (`kW`). Таким образом, текст выражения будет `{kW}`. Здесь вы можете выполнять любые конверсии, например, переводить киловатты в ватты, написав `{kW} * 1000`.

Это выражение рассчитывается при каждом изменении переменной (т.е. каждый раз при опросе устройства). Оно должно всегда иметь результатом число.

Поскольку `kW` - это счетчик потребляемой энергии, установите тип статистического канала на **Счетчик**. Этот тип канала рассчитывает "скорость изменения" счетчика, поэтому значение канала будет измеряться в киловаттах в секунду. Больше информации о типах статистических каналов можно найти здесь[725].

Наконец, ваши опции каналов будут похожи на следующие:



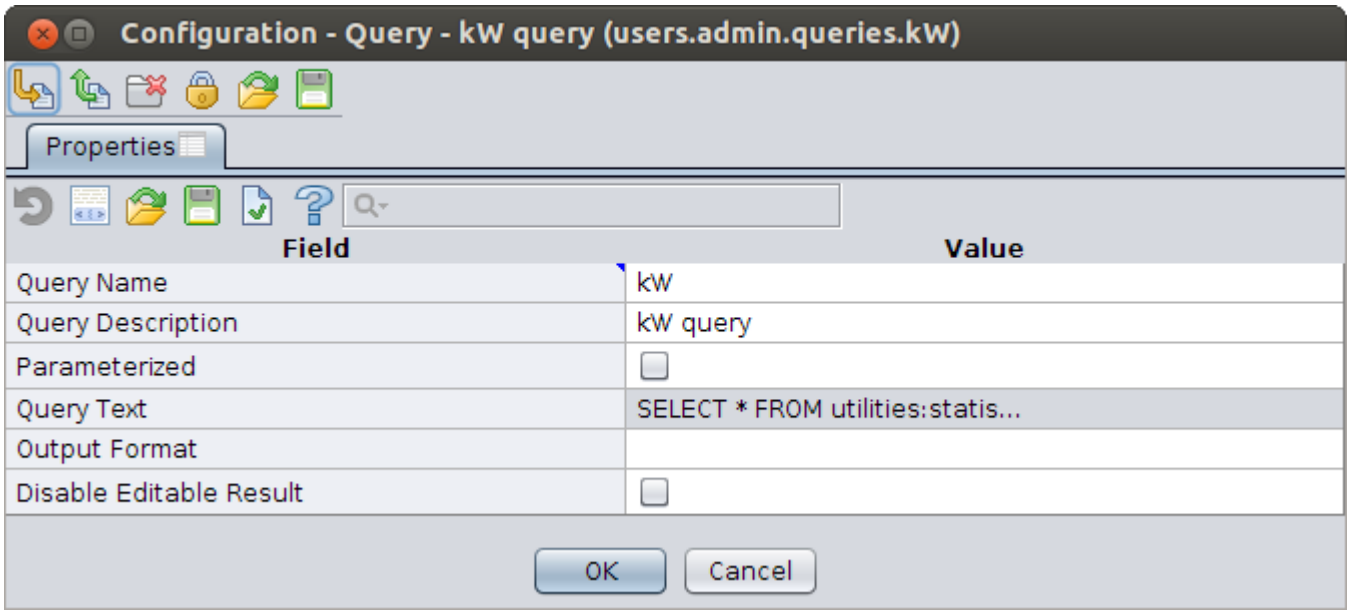
Выбранные параметры **Агрегирования** означают, что будут рассчитываться **средние**, **минимальные** и **максимальные** значения для каждого **Периода хранения** (определенного ниже). **Периоды хранения** определяют длину истрии для каждого показателя агрегирования: один час поминутной статистики (средние значения для каждой из последних 60 минут), один день статистики по часам и т.д.

2. Создание запроса о получении исходных данных для отчета о потреблении

Данный отчет о потреблении основан на запросе 829. Для создания отчета дважды кликните на узле Запросы 1546 в системном дереве. Давайте создадим запрос со следующим **Текстом запроса**:

```
SELECT
 stats.statistics$context,
 stats.statistics$end,
 stats.statistics$average * 3600
FROM
 utilities:statistics("users.admin.devices.counterDevice", "kW", null, "hour") as
stats
```

Вот окно конфигурации запроса:



Этот запрос вызывает функцию статистика[1618] из контекста утилиты[1613] для получения статистических показателей потребления. Он передает функции следующие параметры:

- `"users.admin.devices.counterDevice"` - путь к вашему устройству,
- `“kW”` - имя статистического канала,
- `“null”` - клавиша строки, не используется в данном случае,
- `“hour”` - показатель агрегирования (запрос выдаст ненулевые результаты, если прошло больше одного часа, поскольку мы создали наш статистический канал, но чтобы отладить его и посмотреть результаты раньше, здесь можно временно использовать значение “минуты”).

Запрос выбирает три поля из результата функции:

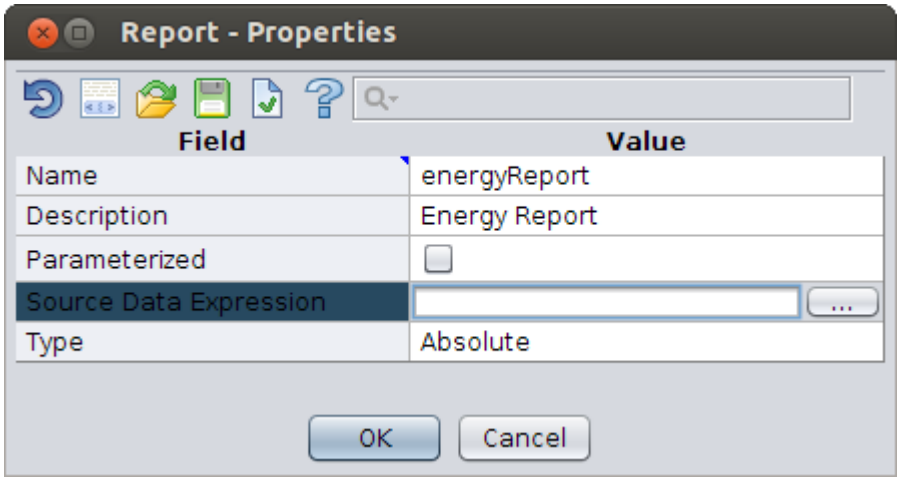
- `stats.statistics$context` - контекст устройства,
- `stats.statistics$end` - конец временной метки каждого периода агрегации,
- `stats.statistics$average * 3600` - среднее посекундное значение потребления, умноженное на 3600 секунд в час.

Когда создание запроса заканчивается, вы можете увидеть его путем двойного нажатия на него. Вы увидите обобщенную таблицу потребления.

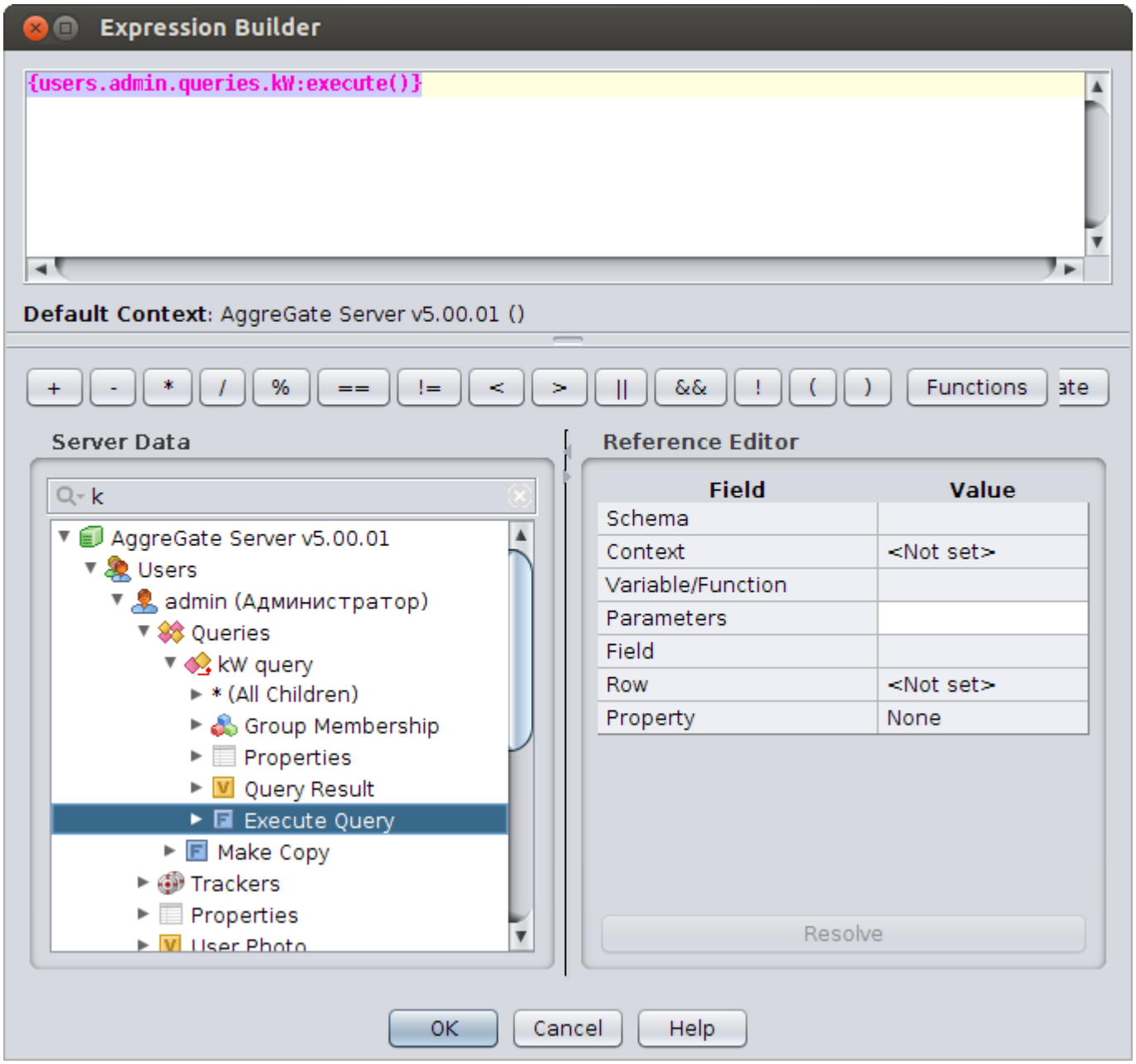
3. Создание отчета потребления

Для создания отчета[928] дважды кликните на узле отчеты[1551] в системном дереве. Введите **Имя** и **Описание** в открывшемся окне.

Затем кликните на кнопке **‘...’** для редактирования **Выражения исходных данных**.

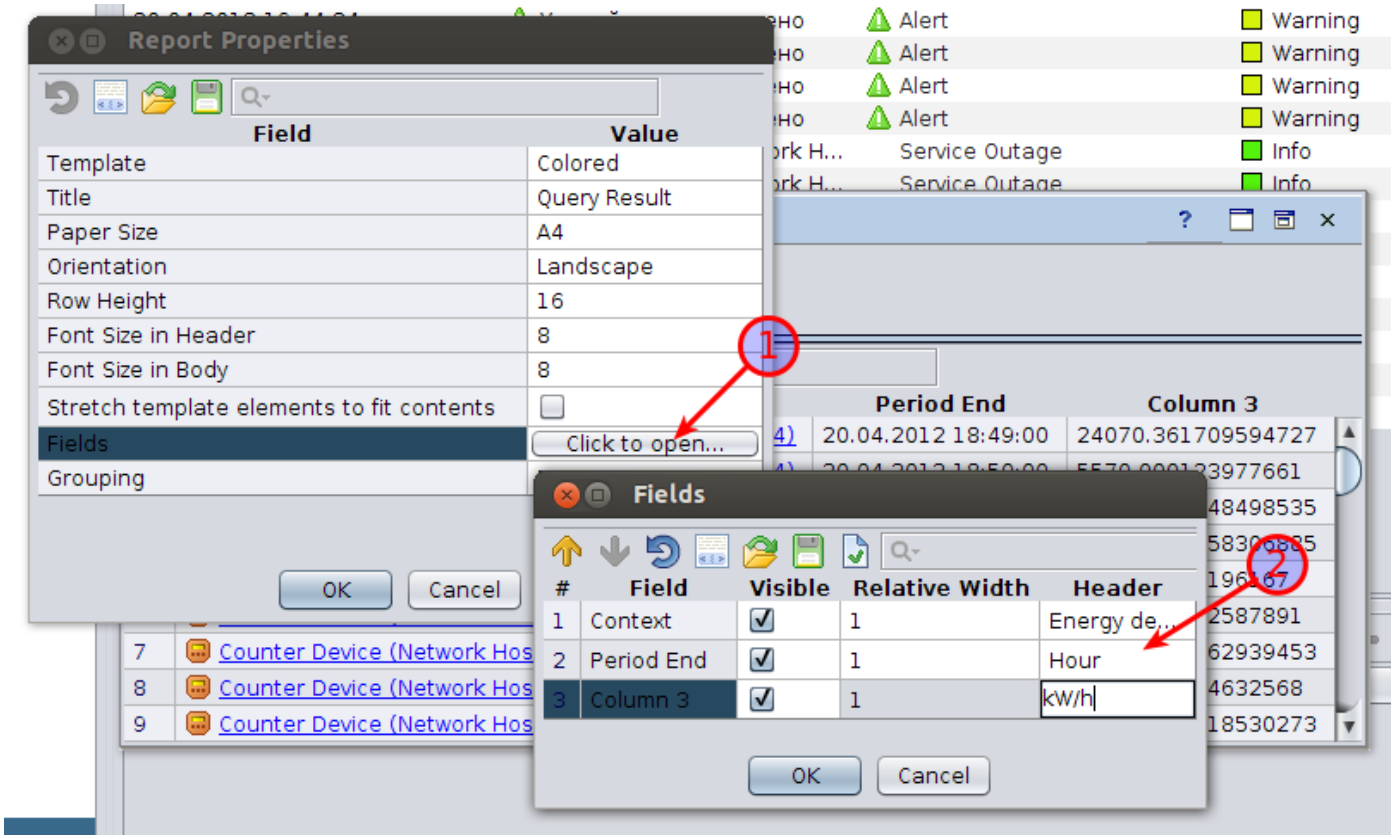


Появится окно Редактор выражений. Теперь найдите ваш запрос `kW` в дереве Данных сервера в `Users/admin/Queries`. Разверните его, чтобы увидеть узел **Выпонить запрос** и дважды щелкните по нему. Вы получите строку `{users.admin.queries.kW:execute()}`, вставленную в выражение.



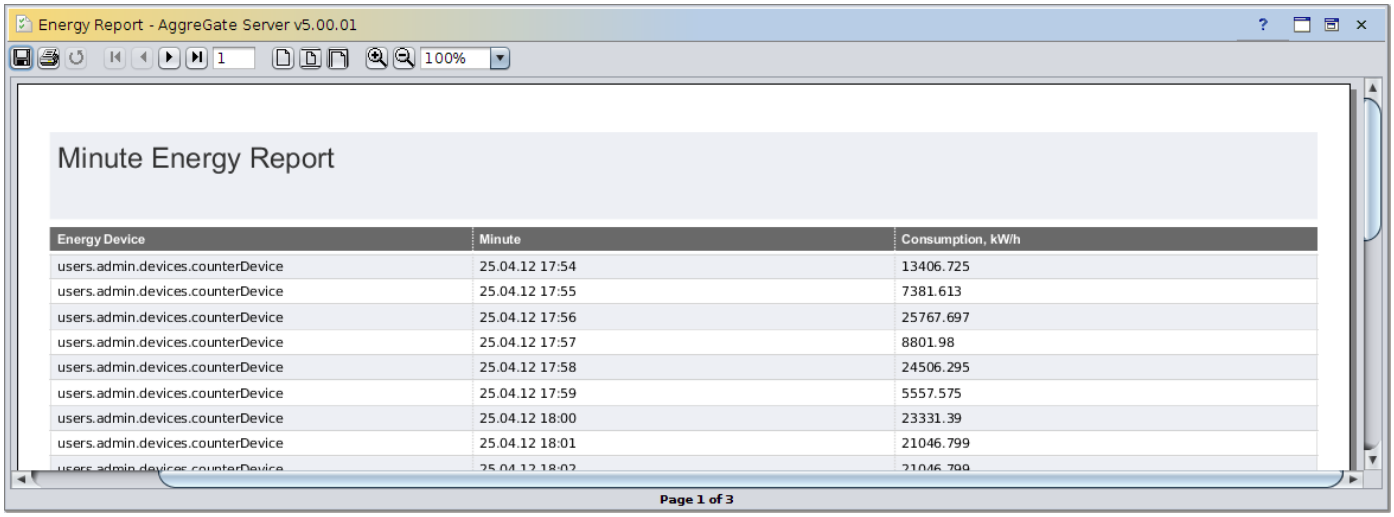
Это выражение будет выполнять запрос каждый раз при создании отчета. Кликните OK дважды, чтобы закрыть этот и родительский диалог.

Появится диалог **Свойства отчетов**. Здесь нужно исправить названия заголовков отчетов. Сперва кликните на **Полях**, а затем введите необходимые тексты заголовков колонки отчетов в колонке **Заголовок**.



Нажмите OK в обоих диалогах. После этого вам будет предложено отредактировать шаблон отчета - отклоните этот запрос, чтобы пропустить запуск Редактора отчетов 416.

Вот и все! Теперь можно запустить ваш отчет двойным нажатием на Системном дереве:



18.18 Прогнозирование нарушений SLA

Этот урок описывает, как AtomMind может помочь с проактивным управлением устройствами, сервисами и процессами. Это происходит с помощью операторов тревог 779, если числовой порог SLA будет в ближайшем будущем нарушен в сответствии с динамикой статистического тренда, основанного на определенной метрике.

Вот пример конфигураций тревог:

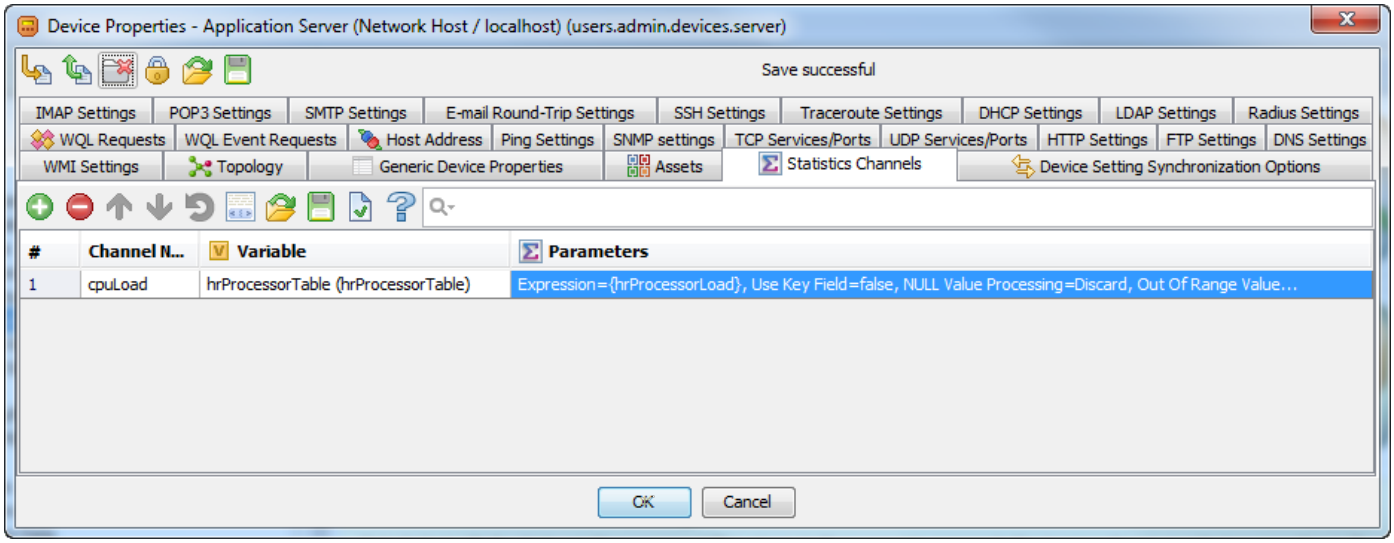
- Создать тревогу, если загрузка HDD на критически важном сервере достигнет 95% через две недели
- Создать тревогу, если средняя доступность многокомпонентного бизнес-сервиса упадет ниже 99% через три месяца

В нашем примере мы настроим тревогу, которая будет вызываться, если средняя нагрузка на ЦП на сервере поднимется выше 80% за одну неделю.

1. Создание статистического канала

Прежде всего, необходимо установить статистический канал 718 для расчета ежедневных средних показателей нагрузки ЦП на нашем сервере.

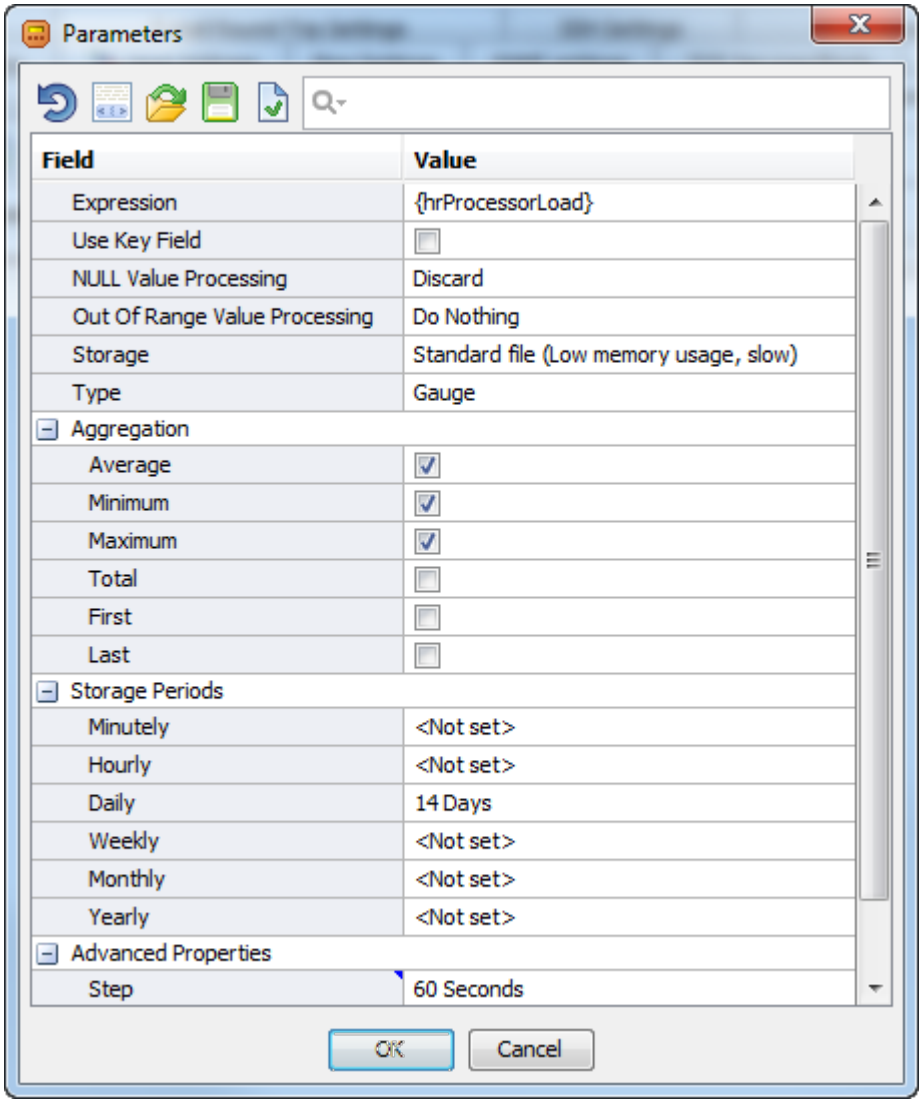
Для создания статистического канала во всплывающем меню сетевого устройства выберите **Редактировать свойства устройства**, перейдите во вкладку **Статистические каналы** и добавьте новый ряд к таблице каналов:



Настройте **Переменную** так, чтобы она указывала на переменную, содержащую значения нагрузки на ЦП. Откройте **Параметры** канала и установите **Выражение**, которое вернет числовые значения, чьи SLA будут отслеживаться тревогой.

Также хорошо отключить **Периоды хранения** для всех типов архивов, кроме Ежедневного, поскольку мы будем использовать только ежедневные данные для анализа. Чтобы это сделать, щелкните правой кнопкой в пределах любого периода хранения и выберите **Удалить значение** их контекстного меню.

Сконфигурируйте длину **Ежедневного** архива, чтобы убедиться, что прогноз будет основан на статистических данных достаточной длины.



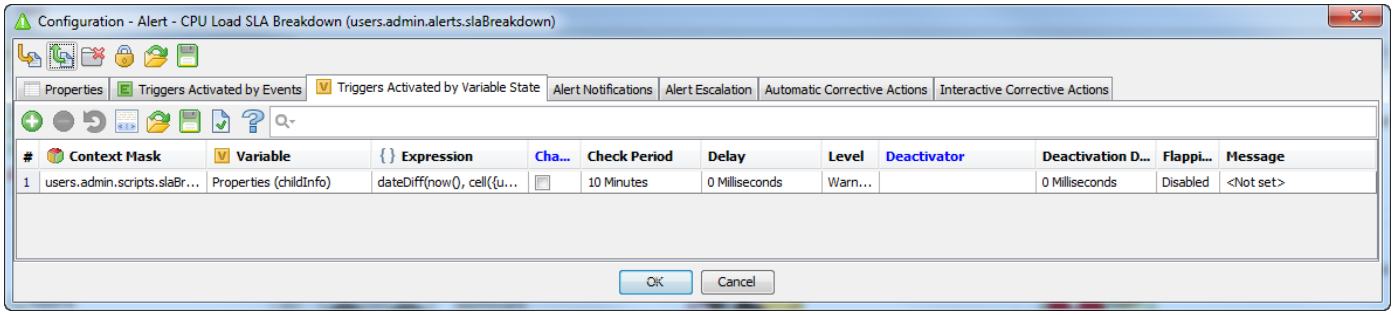
2. Создание тревоги

Создайте новую тревогу[779], которая будет появляться при определении будущих нарушений SLA. Установите уведомления[791] тревоги (такие как отправка письма или SMS).

К тревоге добавьте один триггер переменной[782] и настройте его следующим образом:

- Контекстная маска: `users.admin.scripts.slaBreakdown` (скрипт расчета даты нарушения SLA)
- Переменная: `childInfo` (свойства)
- Выражение:
```
dateDiff(now(),
cell({users.admin.scripts.slaBreakdown:execute("users.admin.devices.criticalServer",
"cpuLoad", 4, 0, 80)}), "day") < 7 && dateDiff(now(),
cell({users.admin.scripts.slaBreakdown:execute("users.admin.devices.criticalServer",
"cpuLoad", 4, 0, 80)}), "day") > 0
```
- Задержка: `0 ms`

Период проверки не должен быть очень коротким, поскольку расчет даты нарушения SLA на базе статистики - это ресурсозатратная задача. Десятиминутный или более долгий **Период проверки** должен быть достаточен.



Самая сложная часть этой тревоги - это его **Выражение**. Это выражение ссылается на скрипт 879 **Расчета даты нарушения SLA**, являющейся частью дистрибутива AtomMind. Ссылка `{users.admin.scripts.slaBreakdown:execute("users.admin.devices.criticalServer", "cpuLoad", 4, 0, 80)}` вызывает выполнение функции 1593 из контекста скрипта и передает ему следующие параметры:

- `users.admin.devices.criticalServer` - путь контекста устройства 1494, соответствующий нашему целевому серверу
- `cpuLoad` - имя статистического канала, откуда берутся данные
- `4` - группирование констант временных единиц 2160 для ежедневного группирования
- `0` - агрегирование констант типов 2160 для выведения среднего значения
- `80` - числовое значение SLA

Этот скрипт вернет дату, когда тренд ежедневных средних занчений пересечет порог нагрузки на ЦП в 80%. Однако дата возвращается, "завернутая" в таблицу данных 49 (поскольку наша ссылка вызывает контекстную функцию, а контекстные функции всегда возвращают таблицы данных), поэтому нужно использовать функцию языка выражений 124 `cell()` для извлечения актуального значения ячейки (т.е. дату нарушения SLA).

Наконец, выражение использует функцию языка выражений 124 `dateDiff()`, чтобы вернуть true, если нарушение SLA произойдет в течение следующих семи дней.

18.19 Умные базовые тревоги

Простые тревоги часто настариваются так, чтобы возникать сразу после того, как определенная метрика превышает заданный порог. Например, тревога возникает, когда нагрузка на ЦП сервера превышает 80% и держится на этом уровне больше часа.

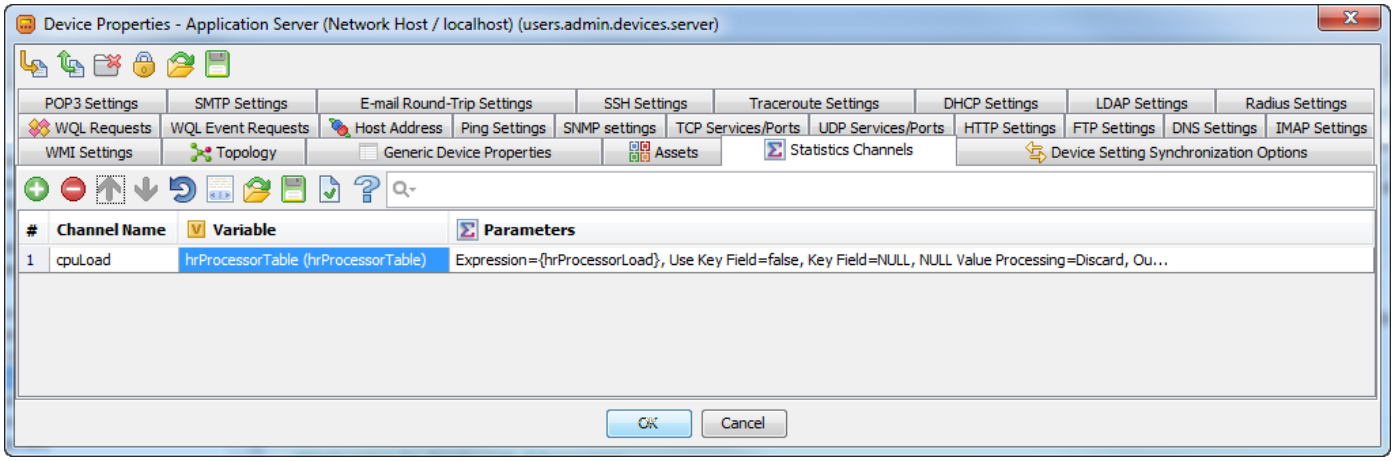
Подобная настройка тревог хороша для контроля за выполнением SLA. Однако в некоторых случаях тревога вызывается, если определенное значение метрики или ее среднее значение за короткий срок значительное превышает среднее значение в течение длительного срока. Эта ситуация может сигнализировать о предстоящей поблеме, даже если текущее значение все еще ниже уровня SLA.

Этот урок объясняет, как настроить тревогу так, чтобы она возникала, если среднее значение нагрузки на ЦП сервера за последний час превышет это же среднее значение за последний месяц на 20 процентов.

1. Создание статистического канала

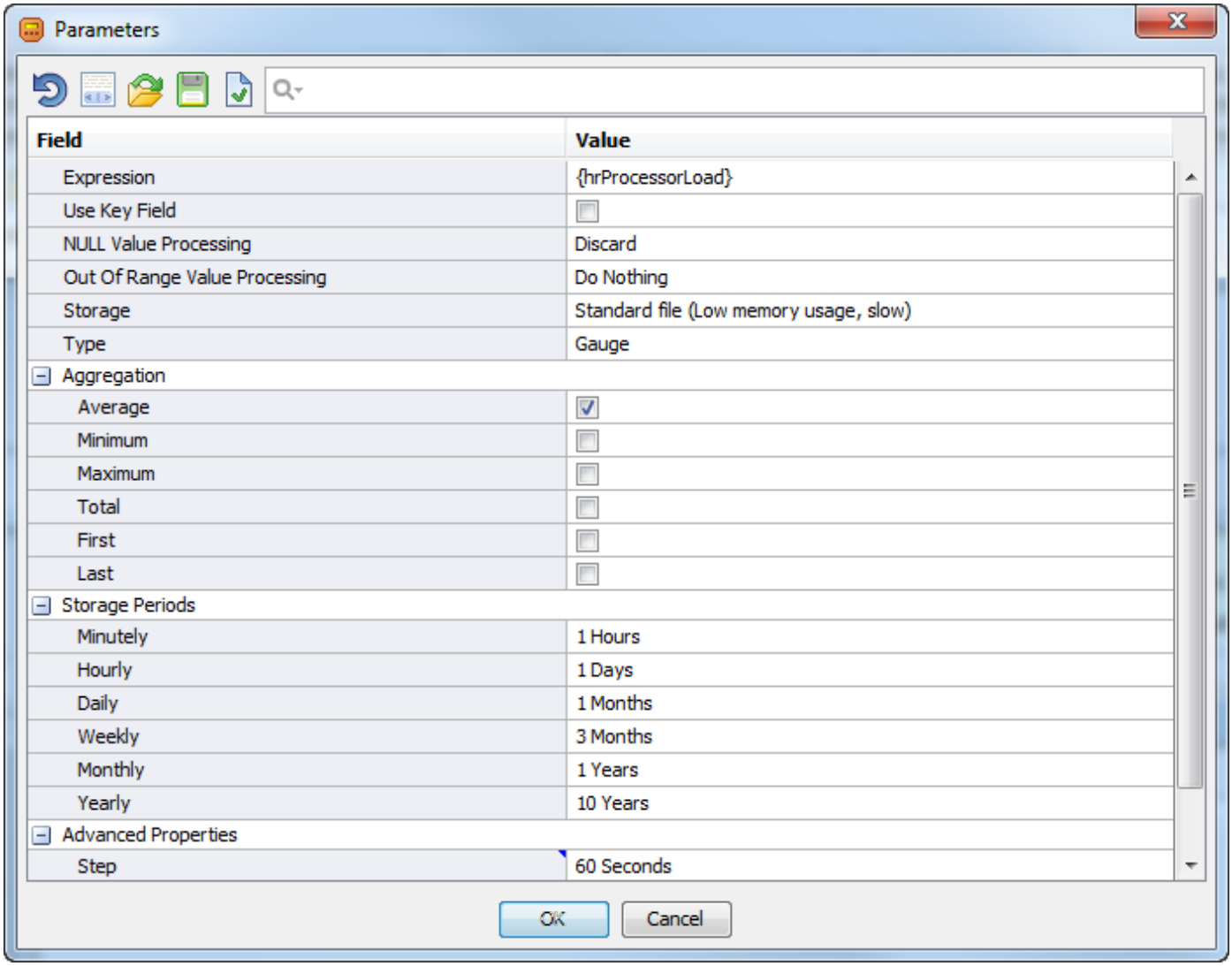
Прежде всего, нужно настроить статистический канал 718 для расчета средних значений нагрузки на ЦП сервера.

Для создания статистического канала выберите **Редактировать свойства устройства** из всплывающего меню сетевого устройства, перейдите во вкладку **Статистические каналы** и добавьте новый ряд к таблице каналов:



Настройте **Переменную** так, чтобы она указывала на переменную, содержащую значения нагрузки на ЦП. Откройте канал **Параметры** и настройте **Выражение**, которое вернет те числовые значения, чьи средние значения будут отслеживаться тревогой.

Также желательно отключить все типы **Агрегирования**, кроме **Среднего**. Это позволит сохранить место в статистической базе данных.



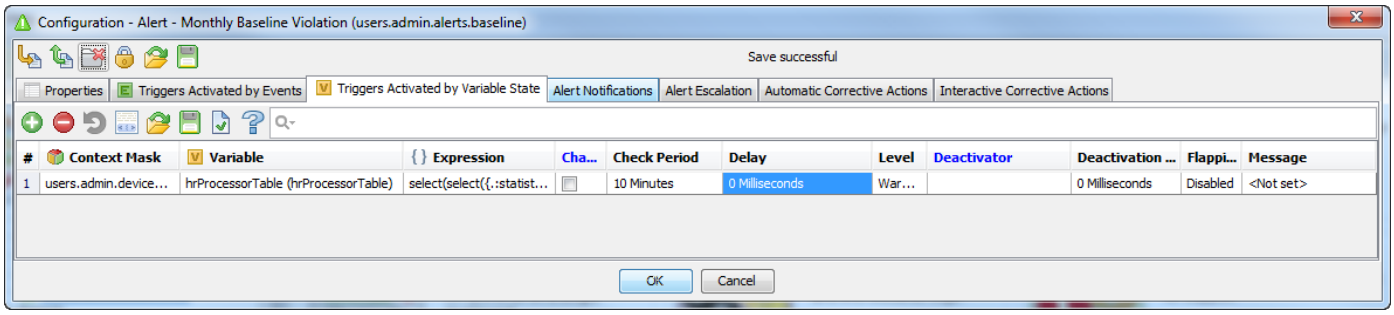
2. Создание тревоги

Создайте тревогу[779], которая будет возникать при серьезных нарушениях базовых параметров. Настройте уведомления[791] тревоги (такие как отправка электронного письма или SMS).

Добавьте к тревоге один триггер переменной[782] и настройте следующим образом:

- Контекстная маска: `users.admin.devices.criticalServer` (контекст сервера, чья нагрузка на ЦП отслеживается)
- Переменная: `hrProcessorTable` (таблица нагрузки на ЦП)
- Выражение: `select(select({.:statistics}, "statistics", "name", "cpuLoad"), "average", "period", 3) > select(select({.:statistics}, "statistics", "name", "cpuLoad"), "average", "period", 6) + 20`
- Задержка: `0 ms`

Период проверки не должен быть очень коротким, поскольку расчет даты нарушения SLA на базе статистики - это ресурсозатратная задача. Десятиминутный или более долгий **Период проверки** должен быть достаточен.



Выражение этой тревоги несколько сложновато. Оно сравнивает средние значения нагрузки на ЦП за последний день и последний месяц путем их извлечения из переменной статистики 1494.

Давайте проанализируем первую часть `select(select({.:statistics}, "statistics", "name", "cpuLoad"), "average", "period", 3)`. Во-первых, `select({.:statistics}, "statistics", "name", "cpuLoad")` извлекает статистические значения для канала `cpuLoad` из значения переменной `statistics`. Во-вторых, функция `select(statistics, "average", "period", 3)` возвращает среднее значение последнего дня (в процентах) из предыдущей таблицы.

Числовая константа `3` соответствует временной единице 2160 Час.

Вторая часть выражения получает среднее значение за последний месяц таким же образом. Наконец, если среднее значение за последний час превышает среднее значение за последний месяц на 20% или больше, все выражение будет иметь результатом true, и будет возникать тревога.

18.20 Добавление ссылок в таблицы

Инструментальная панель 912 AtomMind, представляющая обзор инфраструктуры устройств, обычно содержит несколько таблиц, в которых можно найти:

- Устройства, имеющие некоторые неполадки в данный момент
- Устройства, имеющие чрезвычайно высокие или низкие показатели производительности
- Другие ресурсы, такие как группы устройств, сегменты сети и т.д.

Такие таблицы полезны сами по себе, но операторам хотелось бы, чтобы при нажатии на определенном ряду с именем устройства в таблице могло вызываться определенное действие, ассоциирующееся с этим устройством, т.е. открытие его "персональной" инструментальной панели.

Этот урок объясняет, как добавлять ссылки к таблице.

Теория

Тема ссылок ячеек довольна сложна, поэтому можно опустить эту часть, если вашей целью является добавление ссылок без понимания того, как они работают.

Технически ссылки отрисовываются в ячейках таблицы, использующей Отрисовщик ссылок 52. Значение, содержащееся в ячейке, будет отображаться как привязка ссылки (т.е. текст ссылки). Текст ссылки, т.е. действие, которое вызывается при нажатии на ссылку, содержится в опциях редактора 51.

Однако если мы зададим опции статического редактора для поля этой таблицы, каждая ссылка в ряду приведет к той же цели. Нужно, чтобы у каждой ссылки была собственная цель, поэтому мы используем следующий метод, чтобы динамически задать опции редктора:

- Добавьте новую колонку к таблице. Значение этой колонки в каждом ряду содержит цель ссылки для ссылки, содержащейся в другой ссылке того же ряда.
- Скройте вышеобозначенную колонку, чтобы она не отображалась в таблице.
- Используйте привязку 741 для получения значения вышеобозначенного скрытого поля и настройте опции редактора поля ссылки. Эта привязка работает по рядам, предлагая отдельные опции редактирования для всех ссылок.

Реализация

Предположим, у нас есть таблица, отображающая показатели температуры разных устройств. Она имеет две колонки: **Устройство** (имя устройства: `device`) и **Температура** (имя поля: `temperature`). Мы хотим, чтобы по колонке **Устройство** можно было кликать, открывая ассоциируемую с ним инструментальную панель (например, график температур).

1. ДОБАВЛЕНИЕ СКРЫТЫХ КОЛОНОК

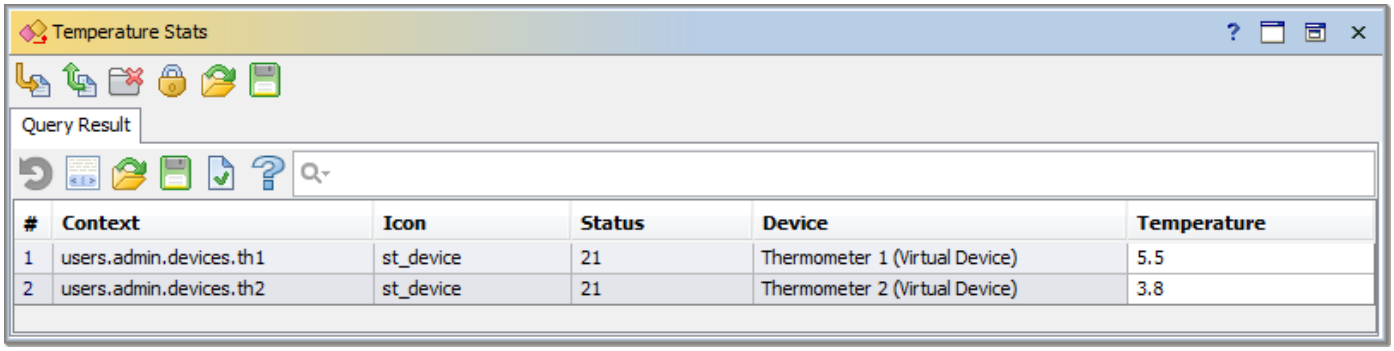
Значения в колонке Устройства являются описаниями устройств, не их системных идентификаторов. Таким образом, нужно добавить новую скрытую колонку `context`, в которой будет обозначен путь контекстов 41 устройств. Существует множество способов это сделать. Например, если мы делаем привязку запроса, мы может добавить ссылку к полю CONTEXT_ID:

```
SELECT
  data$CONTEXT_ID AS context,
  data.info$icon AS icon,
  data.contextStatus$status AS status,
  data.info$description AS device,
  data.temperature$temperature AS temperature
FROM
  users.*.devices.*:info:contextStatus:temperature AS data
```

Обратите внимание, что запрос будет иметь результатом таблицу с пятью колонками:

- Путь первой колонки контекста устройства, который будет скрыт.
- Вторая и третья колонки содержат основные идентификаторы иконки и статуса устройства. Они будут использоваться для отображения иконки статуса устройства рядом со ссылкой. Эти колонки будут также скрыты.
- Четвертая и пятая колонки будут отображать ссылку устройства и чтение температуры устройства.

Результат запроса будет следующим:



#	Context	Icon	Status	Device	Temperature
1	users.admin.devices.th1	st_device	21	Thermometer 1 (Virtual Device)	5.5
2	users.admin.devices.th2	st_device	21	Thermometer 2 (Virtual Device)	3.8

2. ДОБАВЛЕНИЕ ПРИВЯЗКИ ССЫЛКИ

Привязка 741, которую нужно добавить к формату Таблица данных и которая имеет результатом вышеобозначенный запрос, выглядит следующим образом:

```
device#options='action/'+{context}+':configure$' + {icon} + ({status} != null ? '_' + {status} : '')
```

Цель привязки - `device#options`. Она задает параметры редактора в каждой ячейке колонки `device`.

Выражение привязки - `'action/'+{context}+':configure$' + {icon} + ({status} != null ? '_' + {status} : '')`. Она формирует строку редакторских опций, которая является ссылкой 118 на действие 87, которое нужно запускать при нажатии на ссылку.

Актуальный текст параметров редактора, возвращенный вышеобозначенным выражением, будет выглядеть следующим образом (пример): `action/users.admin.devices.thermometer1:configure$device_21`.

Вышеобозначенные опции заставляют систему запустить действие `configure` из контекста `users.admin.devices.thermometer1`, соответствующее устройству термометр. Строка `device_21` обозначает идентификатор иконки, представляющей состояние текущего устройства. Действие конфигурирования 1495 открывает инструментальную панель(и) устройства.

Значения для контекстного имени и статуса/иконки устройств берутся из скрытых полей, определеных в нашем запросе.

Если вам не нужны иконки рядом со ссылками, вы можете убрать вторую и третью линии из пункта SELECT этого запроса и использовать простую привязку:

```
device#options='action/'+{context}+':configure'
```

Запросы AtomMind имеют свойство 837 **Формат выхода**, который был разработан для добавления необходимых отметок к результатам запроса. Можно использовать следующий формат выхода, чтобы скрыть наши дополнительные колонки и добавить привязки ссылок:

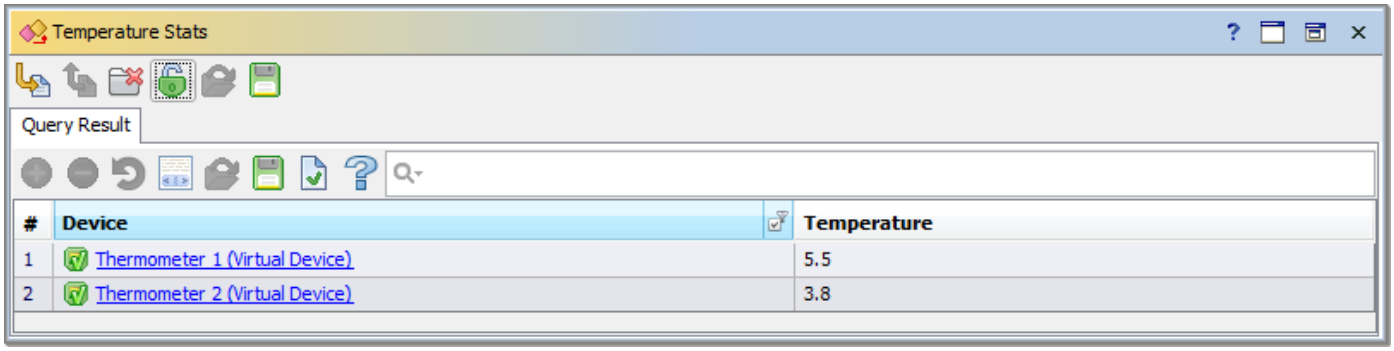
```
<<context><S><F=H><A=>>
<<icon><S><F=H><A=>>
<<status><S><F=H><A=>>
<<device><S><A=><D=Device><E=reference>>
<<temperature><F><A=><D=Temperature>>
<B=<device#options='action/'+{context}+':configure$' + {icon} + ({status} != null ? '_' + {status}
```

Этот формат выхода вносит следующие необходимые исправления в формат выхода запроса по умолчанию:

- Настраивает скрытый флажок в колонках `context`, `icon` и `status`
- Настраивает редактор `reference` в колонке `device`
- Добавляет привязку ссылки

Вот и все! Теперь можно ссылаться на запрос из инструментальной панели для просмотра статистики температуры на интерфейсе оператора первой линии.

Финальная таблица со ссылками:



18.21 Открытие всплывающих отчетов

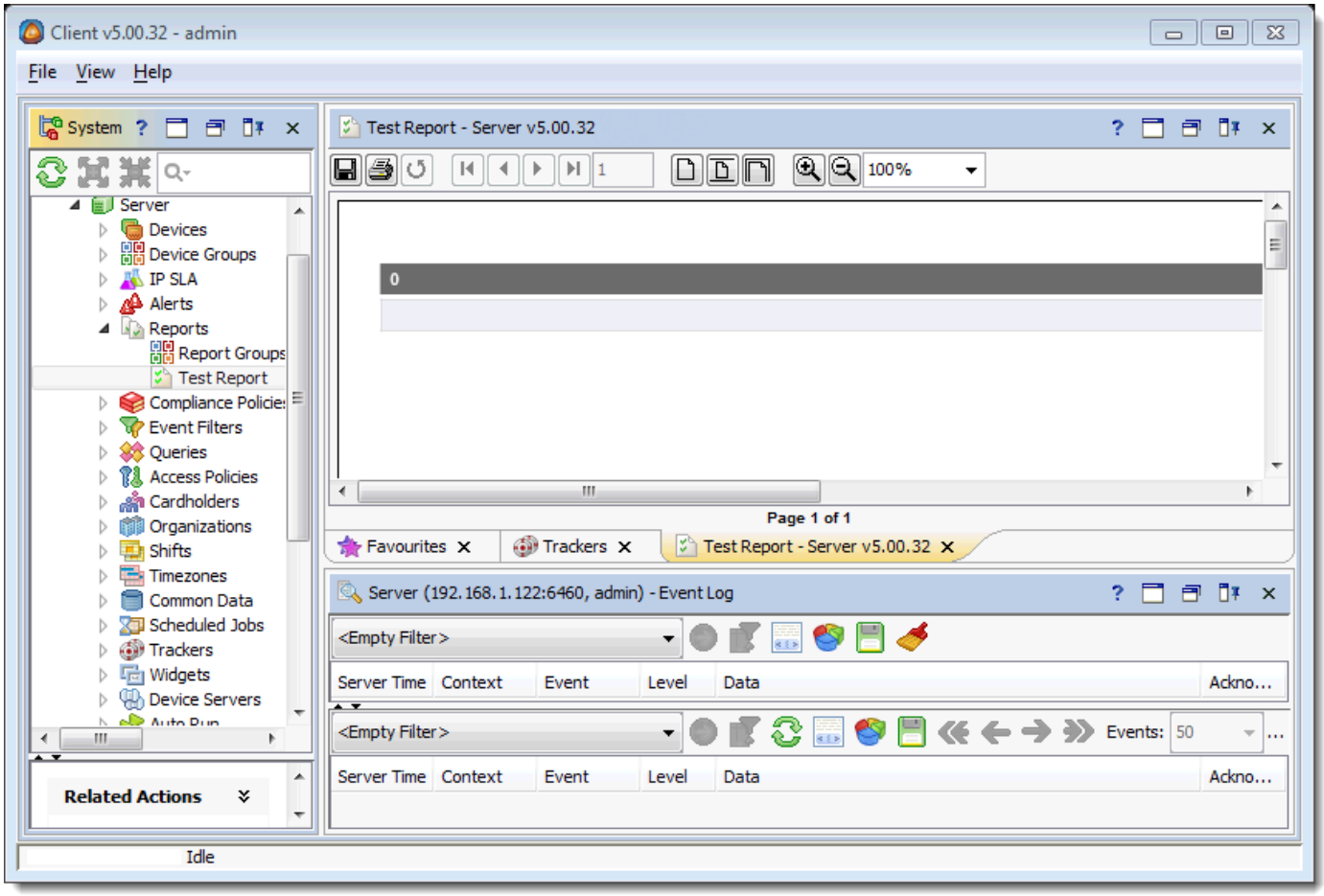
Как открыть отчет из виджета? Как задать местоположение окна отчета? Вот простая инструкция по решению этой задачи.

Создание отчета

Для начала давайте создадим отчет, используемый в этом примере. Для этого:

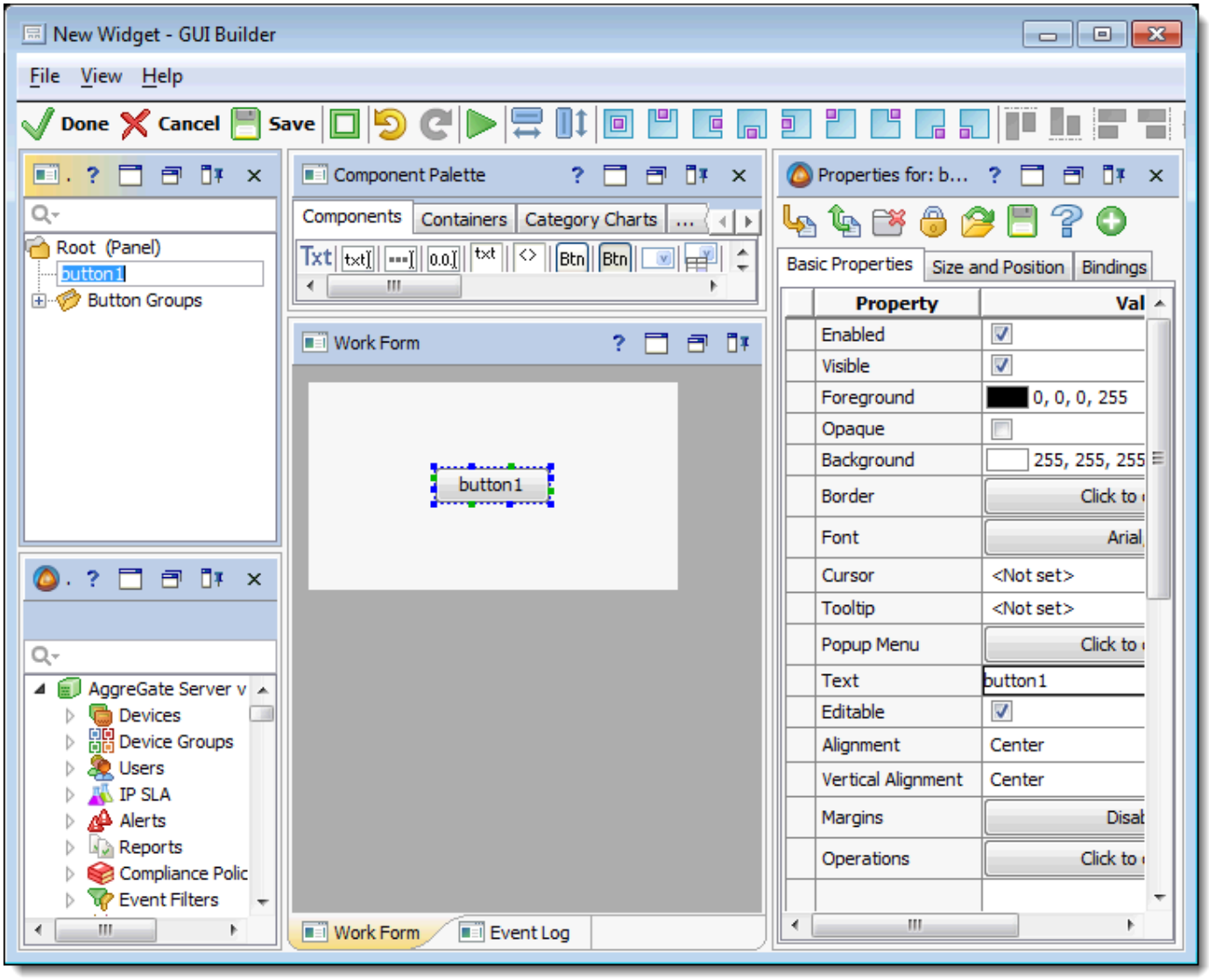
- Дважды щелкните на контексте `Reports`
- Впишите имя (специфичное, пусть будет `testReport`) и описание (скажем, `Test Report`)
- Просто нажмите `OK` в следующем окне, а затем откажитесь редактировать шаблон отчета (поскольку сейчас мы не хотим тратить время, это просто пример).

Теперь мы можем проверить отчет: дважды щелкните по его контексту, и он должен открыться в дополнительном окне справа.



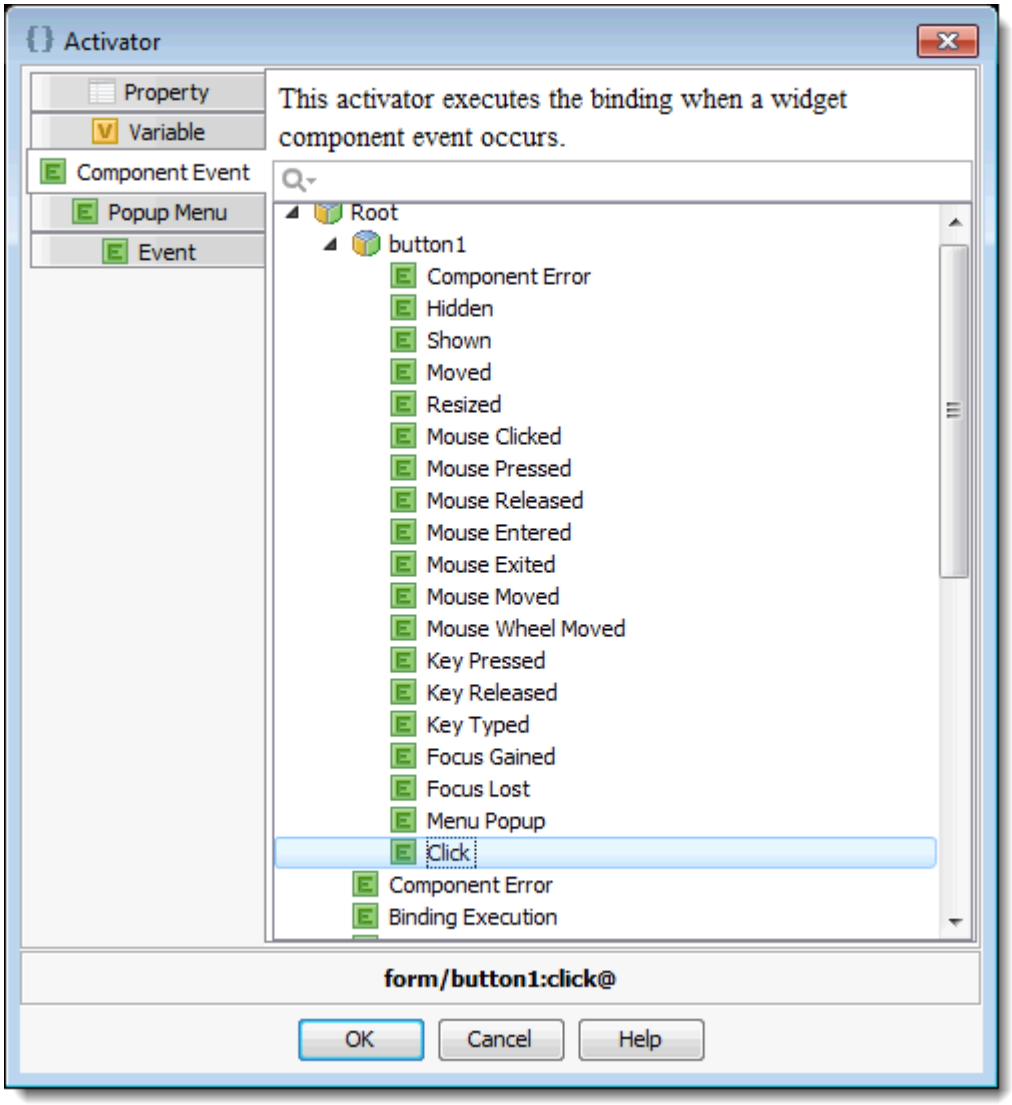
Создание "вызывающего" виджета

Дважды щелкните по контексту `Widgets` и внесите его имя; здесь можно использовать любое имя (и описание), которое вам нравится. Для остальных полей оставьте значения по умолчанию. Окно Редактора виджетов теперь должно появиться с пустой формой. Перетащите кнопку из инструментальной панели компонента в форму. Заметьте, что у кнопки есть имя `button1`.



Мы хотим открыть отчет по имени `testReport`, когда пользователь нажимает на эту кнопку. Чтобы это сделать, нужно связать событие клика с вызовом действия отчета `Show`. Для добавления привязки щелкните правой кнопкой мыши по кнопке и выберите меню `Edit bindings`. В открывшемся окне нажмите на зеленую кнопку со знаком "плюс" для добавления привязок. Для добавленной записи очистите кнопку-флажок `On start`, оставьте настройку флажка `On event` и убедитесь в том, что не задан `Periodically`.

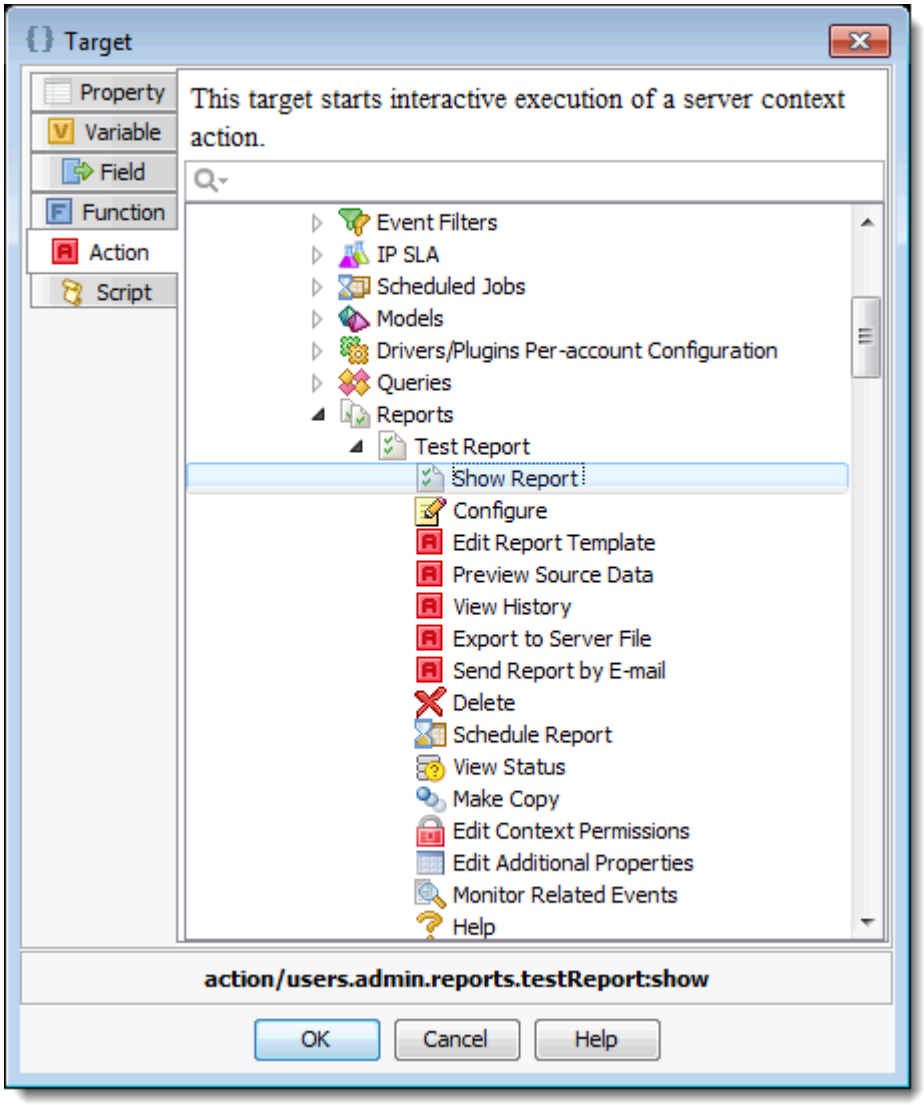
Задайте выражение в событии `click` поля `Activator`. Вам не придется запоминать зашифрованное выражение, как то, что мы собираемся создать. Просто нажмите в поле кнопку `...`. Она откроет окно `Activator`. Откройте вкладку `Component Event` в панели слева и выберите пункт `Click` узла `button1` слева.



Теперь нажмите `OK` и убедитесь, что выражение `Activator` находится в этом поле:

```
form/button1:click@
```

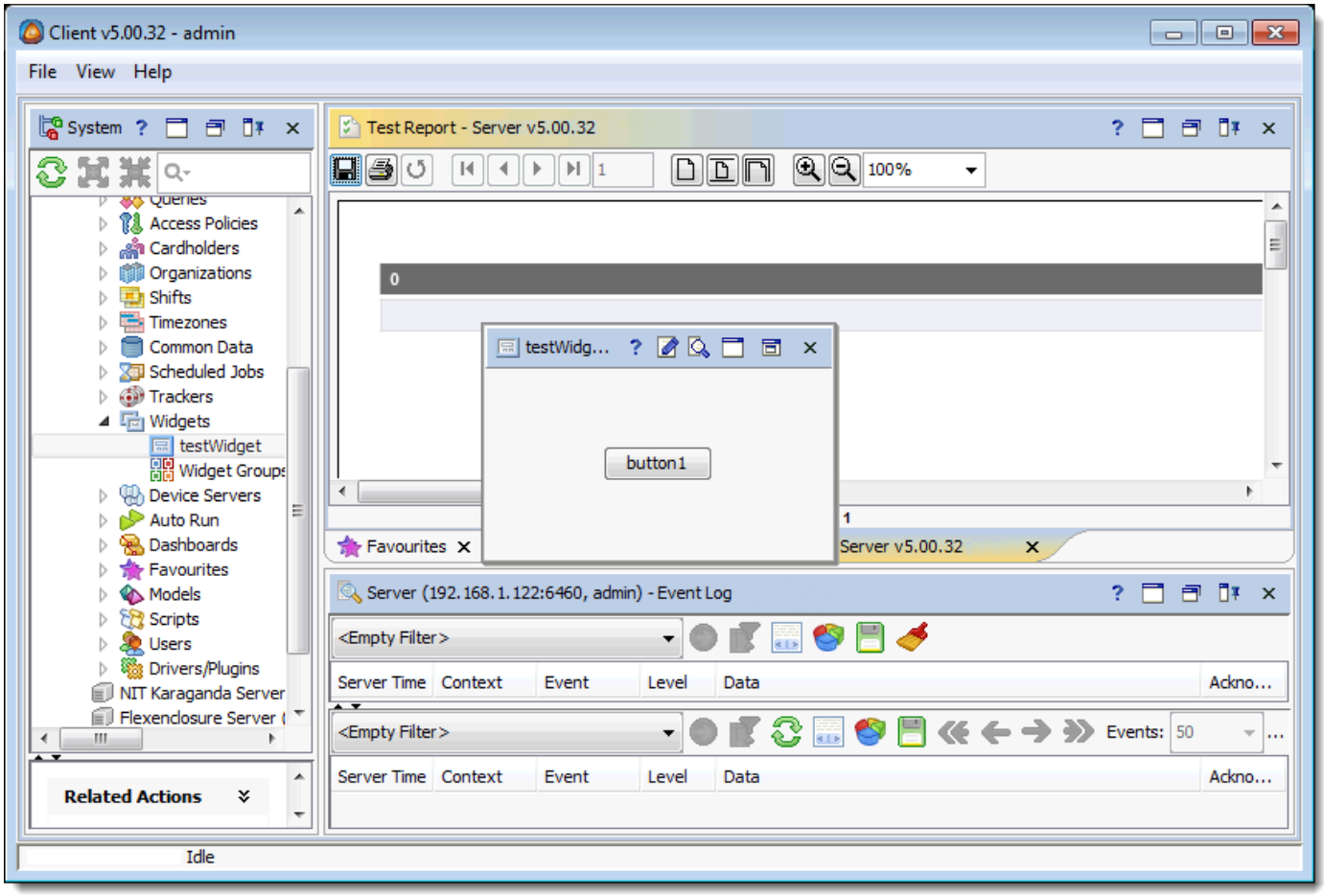
Закончив с активатором, нужно прикрепить действие для его выполнения при появлении события. В нашем случае событие можно задать в поле `Target` следующим образом: нажмите кнопку `...` в этом поле, выберите вкладку `Action` в окне `Action`, а в дереве справа выберите последовательно `Users` -> `admin (Administrator)` -> `Reports` -> `Test Report` и, наконец, действие `Show Report`.



Нажмите OK и проверьте выражение:

```
users.admin.reports.r:show!
```

Теперь мы готовы проверить наш виджет. Итак, сохраните его и запустите. Кликните по кнопке в открывшемся виджете и посмотрите открывшийся `Test Report` в отдельнйо вкладке в пределах `Main Dashboard`. Заметьте, что система автоматически переключается на окно главного клиента и отображает отчет.



Настройка местоположения для отчета

На данном этапе местоположение нашего отчета выбирается автоматически. Зачастую это не то, чего мы хотим. Например, если наш управляющий виджет находится в инструментальной панели, тогда отчет откроется на том же месте в пределах новой вкладки. Это будет выглядеть некрасиво, так как не будет достаточно места для отображения всего отчета, а компоновка других компонентов инструментальной панели будет нарушена. В нашем случае, например, было бы лучше открыть отчет в "плавающем" окне. Тогда окно отчета можно увеличить по необходимости, не меняя компонент инструментальная панель.

Чтобы это реализовать, вернитесь к открытому виджету и закройте его. В Редакторе виджетов откройте привязку кнопки и наберите следующую строку в поле Выражение:

```
table("<<location><T>>", table("<<state><I>>", 1))
```

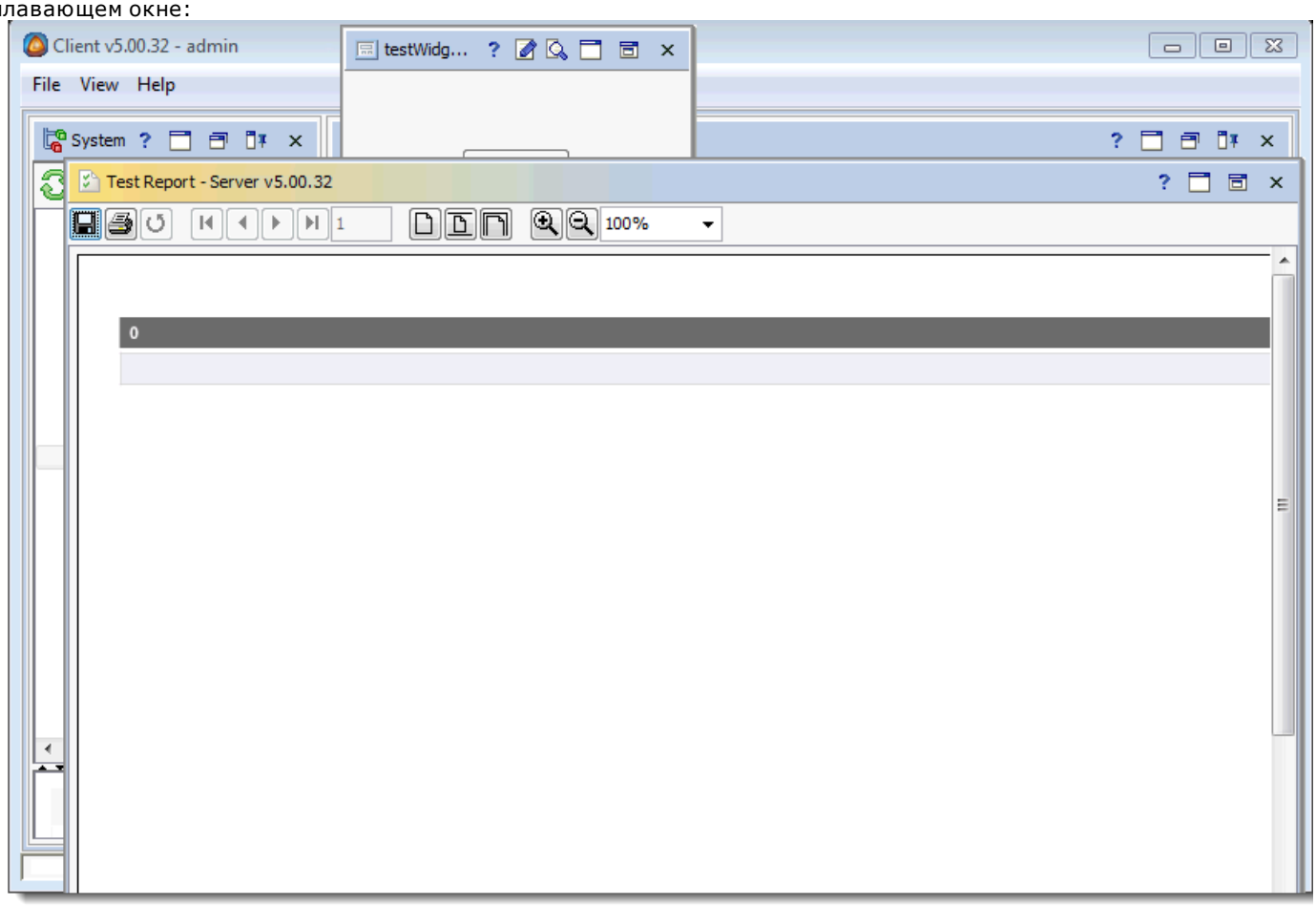
Здесь мы задаем дополнительные параметры для действия, которое мы вызываем кликом. Все параметры действия должны быть занесены в одну таблицу. Действие `show` требует определения местоположения окна в поле по имени `location`. Это поле должно содержать другую таблицу, определяющую само местоположение. Для полного описания таблицы Расположение окна зайдите в специфичная тема 926. Теперь нужно задать состояние окна как "плавающее", которое закодировано целым числом `1` в поле `state`. При использовании функции `table` мы создаем таблицу, имея только одно поле с целым числом под именем `state` (формат таблицы `<<state><I>>`) и введите `1` как его значение:

```
table("<<state><I>>", 1)
```

Затем с помощью той же функции `table` мы вводит эту таблицу как значение для поля `location`:

```
table("<<location><T>>", table("<<state><I>>", 1))
```

Сохраниет виджет, запустите его и будьте уверены, что нажатие кнопки теперь будет открывать виджет в плавающем окне:



18.22 Использование многоуровневых подвиджетов

Виджеты 943 AtomMind позволяют создавать очень сложные компоновки, совмещая множество контейнеров разного типа, используя различные компоновки контейнеров (абсолютный или сетка), а также вложенные друг в друга контейнеры. Однако этого недостаточно для создания динамических компоновок, чья "общая" структура компонентов (а не просто параметры компонентов) динамически меняется в зависимости от среды. Такие динамические виджеты можно создать, используя вложенные подвиджеты 1049, чей шаблон выбирается и меняется "на лету" во время операции по созданию виджета. Этот урок иллюстрирует следующие концепции шаблонов динамических виджетов:

- Использование множества уровней вложенных подвиджетов
- Изменение ссылок шаблонов подвиджетов в режиме выполнения
- Передача параметров подвиджетам

Исходные данные

Чтобы проиллюстрировать концепцию вложенных подвиджетов, мы разработаем виджеты, которые визуализируют состояние и содержание *вендингового аппарата*. Современные вендинговые аппараты удаленно мониторятся и контролируются через GPRS, предоставляя данные об остатке товара и диагностике, близкие к режиму реального времени.

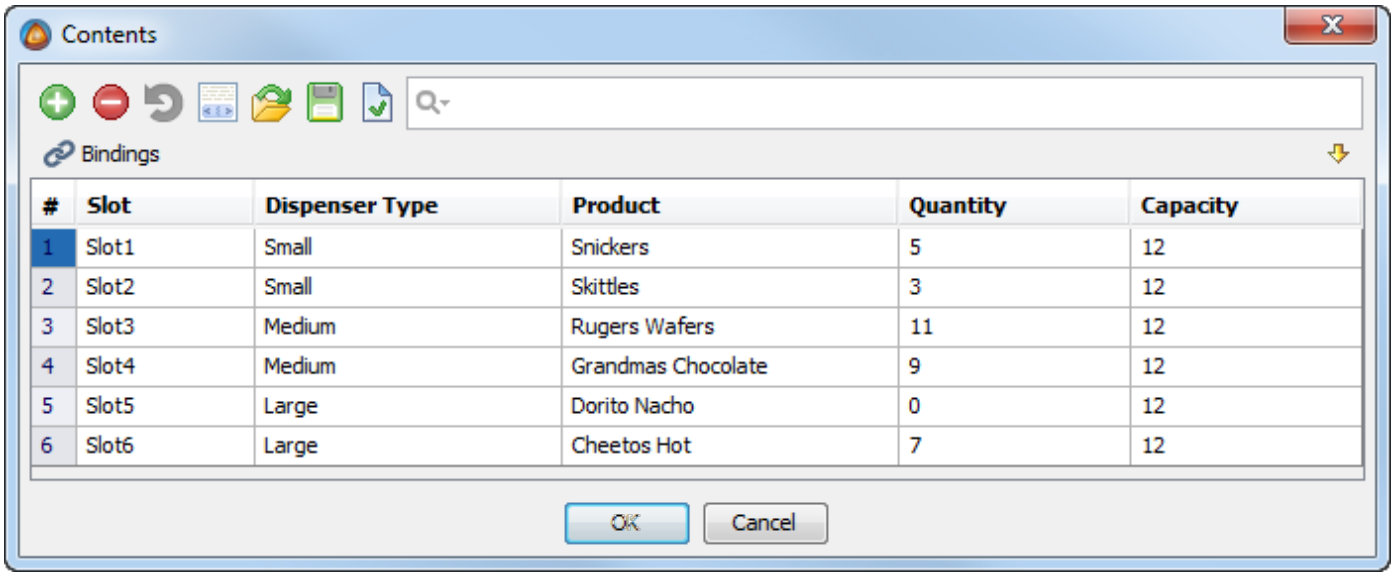
Чтобы урок был простым и коротким, мы смоделировали 810 очень простой вендинговый аппарат легких закусок с таблицей, описывающей *диспенсоры,* установленные в *слотах* машины, типы и емкость этих диспенсоров, типы товара, загружаемого в эти диспенсоры, и количество оставшейся закуски.

Аппарат предоставляет таблицу **Содержимое** (`contents`) со следующими полями:

- **Слот** - описание диспенсора
- **Тип** - тип диспенсора (маленький, средний или большой)
- **Продукт** - имя продукта, загруженного в данный момент в диспенсор

- **Количество** - количество легкой закуски, которая в данный момент находится в диспенсоре
- **Емкость** - текущая емкость диспенсора

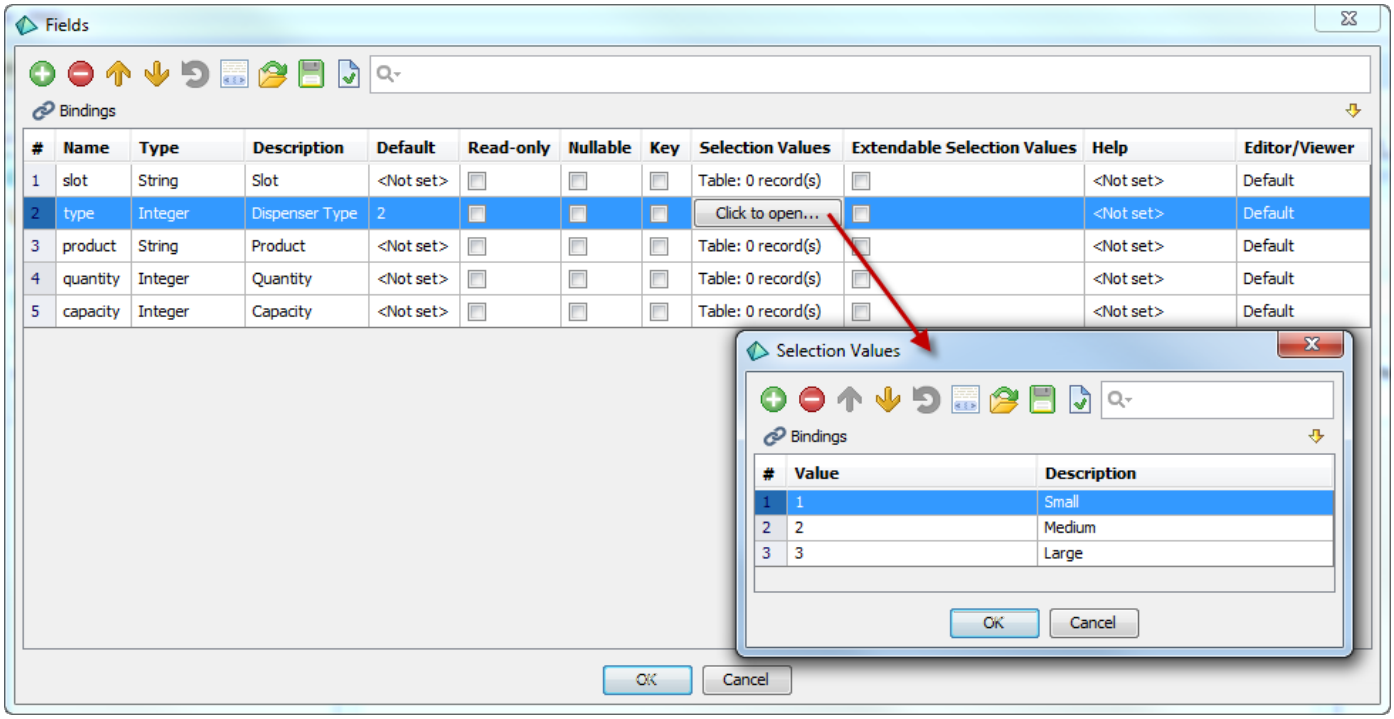
Вот как выглядит таблица **Содержимое**:



#	Slot	Dispenser Type	Product	Quantity	Capacity
1	Slot1	Small	Snickers	5	12
2	Slot2	Small	Skittles	3	12
3	Slot3	Medium	Rugers Wafers	11	12
4	Slot4	Medium	Grandmas Chocolate	9	12
5	Slot5	Large	Dorito Nacho	0	12
6	Slot6	Large	Cheetos Hot	7	12

Таким образом, в машине шесть *слотов* с диспенсорами различных типов, установленных в них. Предположительно каждый слот может иметь диспенсор любого типа (маленький, средний или большой). В нашем случае слоты Slot1 и Slot2 имеют Маленькие диспенсоры, Slot3 и Slot4 - Средние диспенсоры, а Slot5 и Slot6 заняты Большими диспенсорами. Это динамическая компоновка, сконфигурированная инженерами во время установки аппарата.

Ниже представлен скриншот формата таблицы содержимого аппарата:



#	Name	Type	Description	Default	Read-only	Nullable	Key	Selection Values	Extendable Selection Values	Help	Editor/Viewer
1	slot	String	Slot	<Not set>	☐	☐	☐	Table: 0 record(s)	☐	<Not set>	Default
2	type	Integer	Dispenser Type	2	☐	☐	☐	Click to open...	☐	<Not set>	Default
3	product	String	Product	<Not set>	☐	☐	☐	Table: 0 record(s)	☐	<Not set>	Default
4	quantity	Integer	Quantity	<Not set>	☐	☐	☐	Table: 0 record(s)	☐	<Not set>	Default
5	capacity	Integer	Capacity	<Not set>	☐	☐	☐	Table: 0 record(s)	☐	<Not set>	Default

#	Value	Description
1	1	Small
2	2	Medium
3	3	Large

Структура виджетов компоновки аппарата

Наша машина будет отрисовываться одним виджетом под названием *Компоновка машины*. У этого виджета будут компоненты, обеспечивающие общую информацию о машине и количестве подвиджетов для отрисовки диспенсоров. Каждый такой подвиджет будет называться *Компоновка диспенсора*, а количество подвиджетов Шаблоны диспенсора будет соответствовать количеству доступных типов диспенсоров.

Модель нашей демо-машины определяет три типа диспенсоров: **Маленький** (код: 1), **Средний** (код: 2) и **Большой** (код: 3). Таким образом, необходимо разработать три виджета Компоновка диспенсора для использования их в качестве подвиджетов в Компоновке машины.

Разработка компоновок диспенсора

Начнем с разработки первого виджета компоновки машины для отрисовки Маленьких диспенсоров. Во-первых, создадим абсолютный 946 виджет и назовем его **dispensor_1** с описанием **Шаблон маленького диспенсора**. Здесь важно название, поскольку нам нужно будет динамически ссылаться на виджеты шаблонов диспенсоров по их именам в зависимости от типов диспенсоров, установленных в машине.

Виджет должен использовать компоновку сетка 950, чтобы вся система отрисовки содержимого машины масштабировалась для различных машин и разрешений экрана.

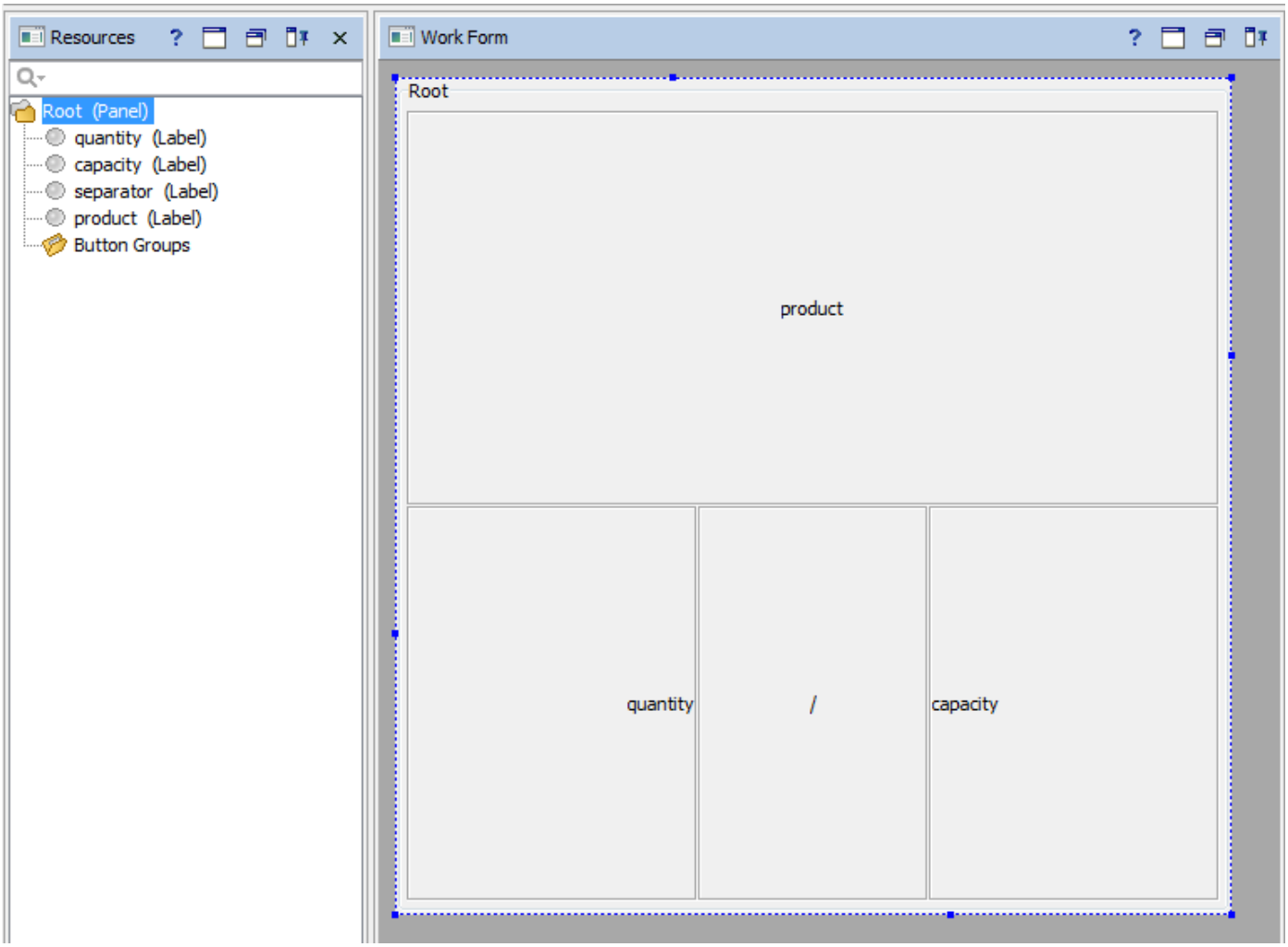
Отметим, что размер подвиджета шаблона диспенсора можно насроить как угодно, если это удобно для отладки, но запущенный подвиджет будет "помещен" в определенный отдел шаблона виджета высокого уровня, оставляя ему неопределенное (и, возможно, маленькое, если виджет высокого уровня масштабирован для низкого разрешения экрана) количество места. Таким образом, шаблон подвиджета должен быть разработан так, чтобы помещаться на маленьком пространстве экрана.

Наш упрощенный шаблон диспенсора будет содержать четыре компонента метка 956:

- Метка имени продукта
- Метка количества
- Метка емкости
- Метка сепаратора для демонстрации использования емкости как "количество / емкость"

Конечно, возможно показывать как количество, так и емкость на одной метке путем связывания нескольких строк в выражении привязки виджета 1296.

Структура компоновки сетка щаблона диспенсора показана на скриншоте снизу. Отметим, что декорации ячейки 427 были включены для того, чтобы получить понятный для восприятия скриншот:



В реальной жизни шаблоны различных типов диспенсоров могут выглядеть совсем по-другому. В этом уроке мы создадим Средний и Большой шаблоны диспенсоров на основе Маленького путем увеличения размера шрифта. Этого достаточно для иллюстрации идеи.

Разработка компоновки машины

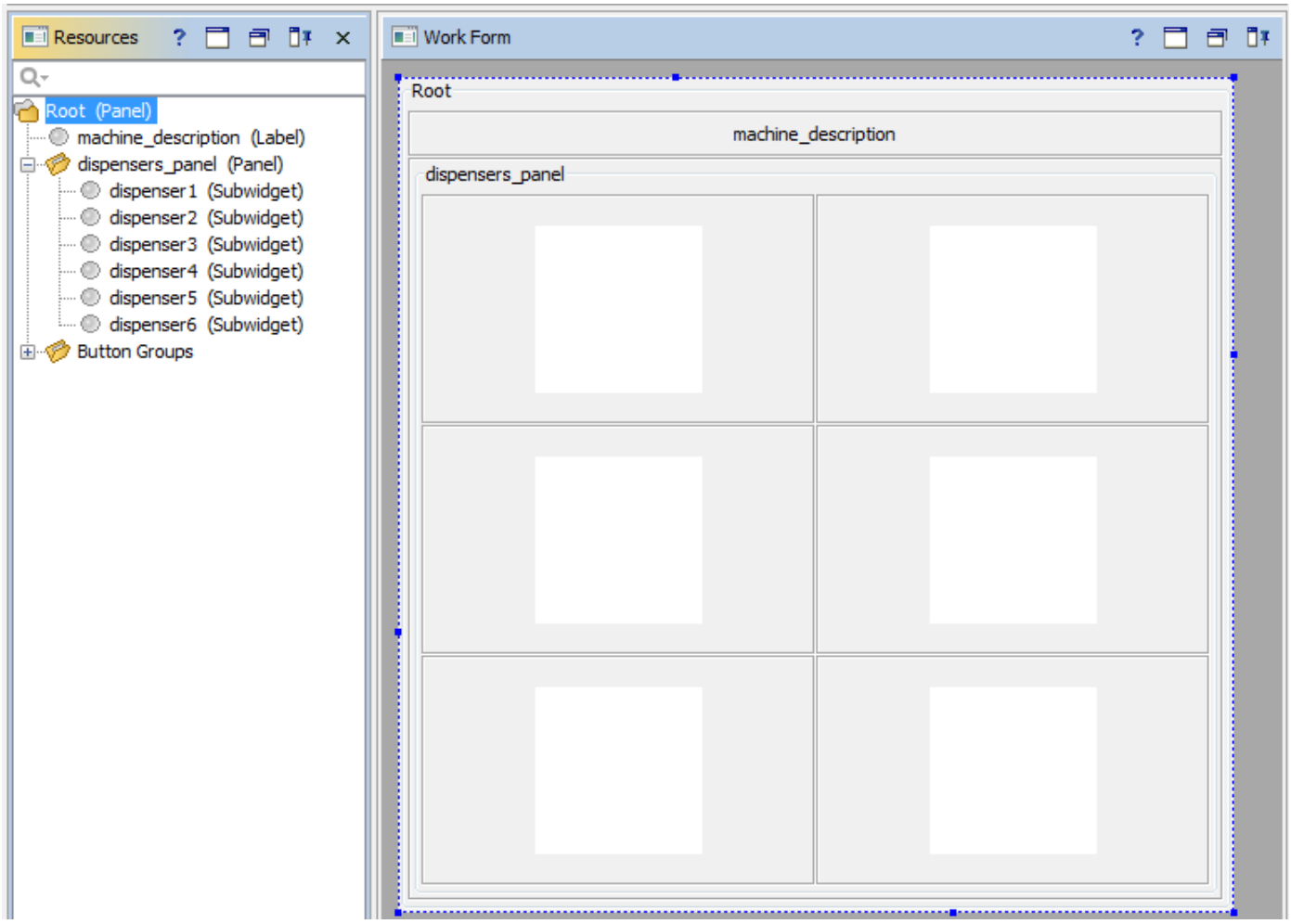
Виджет компоновки машины - это относительный 946 виджет, который прикрепляется к учетным записям вендинговых машин 497. Выражение пригодности 749 виджета Компоновка виджета должен проверять определенные параметры устройств, чтобы использовалась правильная компоновка для отрисовки содержимого устройств.

В этом уроке у нас есть одна компоновка машины, которая будет прикрепляться к контексту нашей модели вендингового аппарата. Наш виджет компоновки машины будет назван **machine_layout**.

Наш простой виджет компоновки машины состоит из двух вертикально расположенных частей:

- Метка заголовка, показывающая описание машины
- Панель, содержащая шесть подвиджетов диспенсоров

Вот как он выглядит в рабочей форме Редактора виджетов (с активированными декорациями):



Динамическое определение шаблонов подвиджетов

Тип диспенсора, в данный момент установленного в слоте, определяется полем **Тип диспенсора** (`type`) в таблице **Содержимое**. Таким образом, нам потребуется шесть привязок, которые меняют свойство **Ссылки** компонентов подвиджета 1049 во время запуска главного виджета. Вот как должна выглядеть привязка шаблона подвиджета:

Цель	`form/slot1:reference`
Выражение	`'users.admin.widgets.dispenser_' + select({.:contents}, "type", "slot", "Slot1")`
Активатор	

Условие

Опции On Startup

Эта привязка меняет ссылку шаблона подвиджета, представляющего содержимое Slot1. Она рассчитывается при запуске главного виджета. Его выражение выстраивает путь контекста 42 подвиджета, который нужно использовать для орисовки содержимого текущего слота. Это делается путем связывания префикса пути контекста (users.admin.widgets.dispenser_) и значения типа диспенсора, установленного в слоте (возвращенного путем выбора значения поля type из строки таблицы contents, чье поле slot равно Slot1).

Декларирование параметров подвиджетов диспенсора

Каждый подвиджет диспенсора должен визуализировать содержимое и статус определенного диспенсора. Таким образом, подвиджет должен иметь несколько параметров, позволяющих ему независимо загружать необходимые данные с сервера. В нашем уроке это следующие параметры:

- Путь контекста устройства 1494 машины, направляющий подвиджет диспенсора на определенную машину (в нашем уроке контекст устройства будет заменен контекстом модели 810)
- Идентификатор строки слота, чье содержимое представлено экземпляром подвиджета диспенсора

См. статью подвиджеты 1049 для получения более подробной информации о параметрах подвиджета.

Для определения парметра подвиджета диспенсора:

- Откройте шаблон в Редакторе виджетов
- Выберите Корневую панель 1042 в Дереве ресурсов 428
- Щелкните правой кнопкой по любой ячейке Сетки свойств 431
- Выберите **Добавить пользовательское свойство**

Свойства для добавления:

- Свойство по имени device с одним полем **Строка**, также имеющим имя device (оставьте все остальные настройки по умолчанию)
- Свойство по имени slot с одним полем **Строка**, также имеющим имя slot (оставьте все остальные настройки по умолчанию)

Привязка параметров подвиджета диспенсора к его визуальным компонентам

Виджет компоновки машины создаст отдельный экземпляр подвиджета диспенсора для каждого диспенсора в машине. Каждый подвиджет получит значения свойств device и slot из виджета верхнего уровня. Теперь нужно использовать эти значения в наших подвиджетах диспенсора. В нашем случае все три подвиджета диспенсора похожи, но в реальной жизни могут значительно отличаться друг от друга.

Привязки подвиджетов диспенсора, возможно, самые сложные во всей структуре. Вот пример привязки, которая показывает имя загруженного продукта в метке 956 product:

Цель form/product:text

Выражение select(getVariable({form/:device$device}, "contents"), "product", "slot", {form/:slot$slot})

Активатор

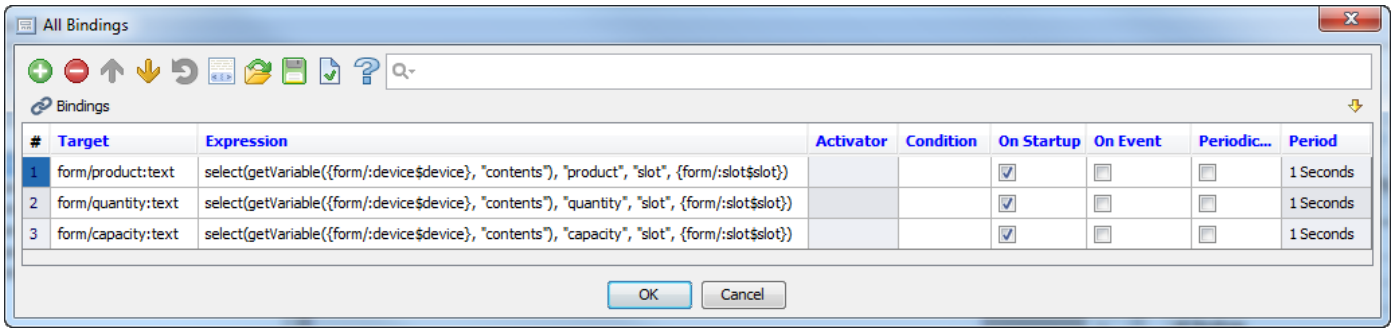
Условие

Опции On Startup

Привязка активируется один раз при запуске подвиджета. Она загружает переменную contents из контекста машины, указанной параметром device (на которого идет ссылка из {form/:device$device}). Как только загружается все содержимое таблицы, выражение привязки выбирает значение колонки product из записи, где колонка slot равна значению параметров подвиджета slot (доступных через ссылку {form/:slot$slot}).

Выражения двух других привязок, показывающие текущее количество продукта и емкости диспенсора, отличаются от вышеобозначенного только другими полями, выбранными вместо поля product.

Вот как выглядит финальная таблица привязок подвиджетов диспенсора:

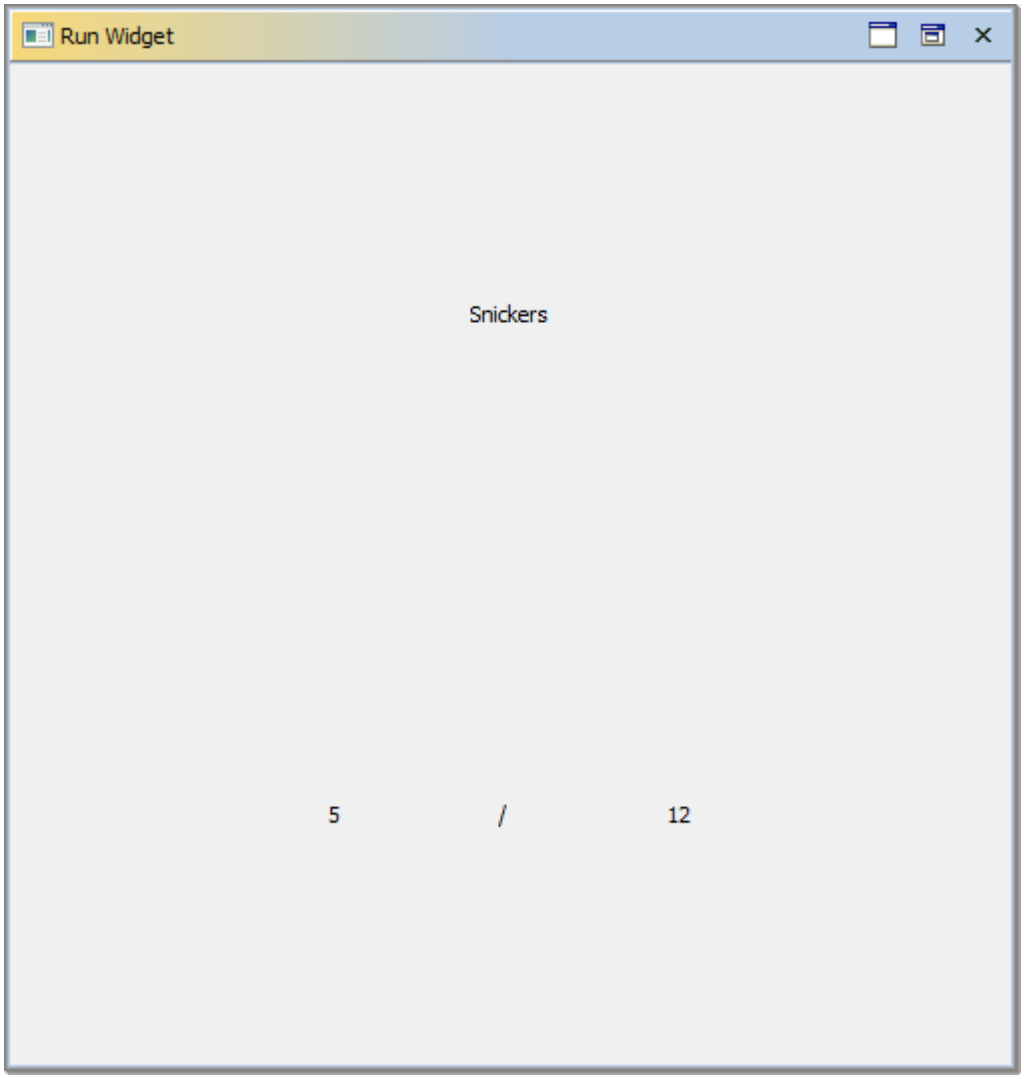


Тестирование подвиджетов диспенсора

Несмотря на относительную простоту подвиджетов диспенсора, потребуется некоторое время для их тестирования и отладки. Отладка подвиджетов, запущенных в виджете компоновки машины, может доставить ненужные затруднения, но, к счастью, возможно протестировать его отдельно, вручную вводя значения параметров `device` и `slot`:

- Откройте шаблон диспенсора в Редакторе виджетов
- Выберите Корневая панель 1042 в Дереве ресурсов 428
- Щелкните правой кнопкой по любой ячейке Сетки свойств 431
- Перейдите во вкладку **Пользовательские свойства**
- Задайте параметр `device` в виде `users.admin.models.vending_machine` (это серверый путь контекста нашего контекста модели машины; путь должен указывать на тестовую вендинговую машину)
- Задайте параметр `slot` на `Slot1` для тестирования подвиджета диспенсора в содержимом Slot1
- Запустите подвиджет путем нажатия на иконку панели управления ( )

Виджет отобразит содержимое Slot1 тестовой машины:



Создание подвиджетов других диспенсоров

После завершения и тестирования шаблона подвиджета Маленького диспенсора создайте две его копии, используя действие Создать копию ( ), задайте для этих копий надлежащие имена/описания (убедитесь, чтобы имена соответствовали образцу `dispenser_` !) и внесите необходимые изменения, чтобы другие диспенсоры отличались друг от друга.

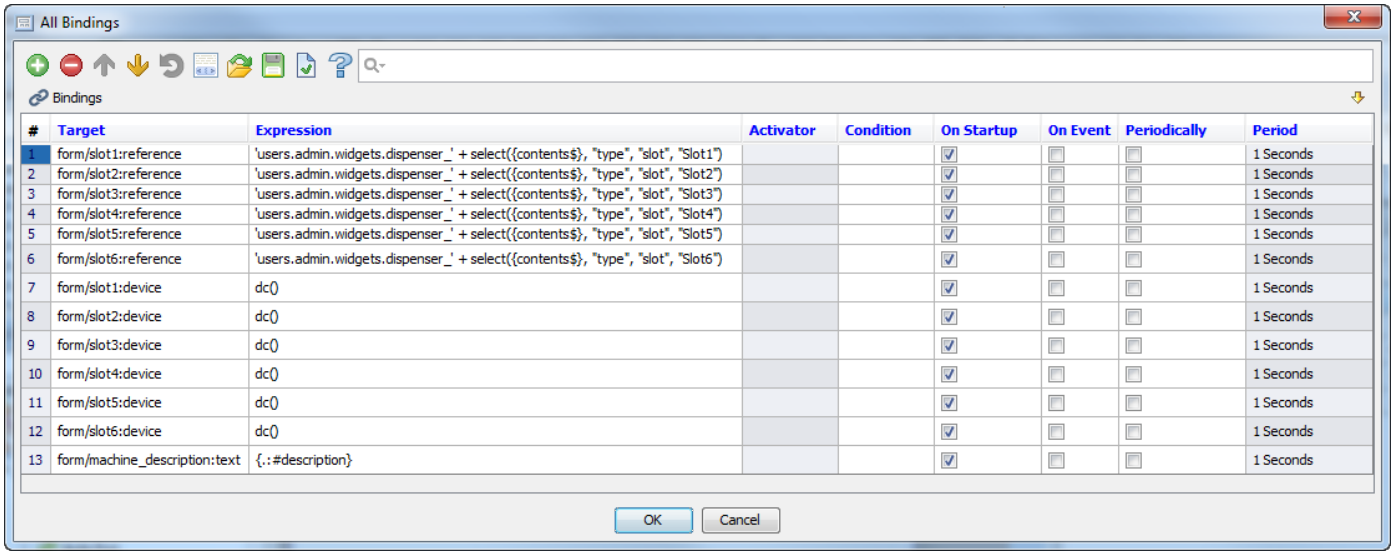
Связывание слотов в компоновке машины

Теперь мы подошли к финальному этапу урока. Последней задачей станет обеспечение корректных значений времени выполнения параметров `device` и `slot` для каждого подвиджета диспенсора в пределах виджета Компоновка машины. Параметр `device` назначается группой из шести подобных привязок:

Цель	`form/slot1:device`
Выражение	`dc()`
Активатор	
Условия	
Опции	`On Startup`

Эта привязка определяет пользовательское свойство `device` компонента подвиджета `slot1` на путь [контекста по умолчанию](946) виджета Компоновка машины (возвращенного функцией **dc()**).

Вот как выглядит финальная таблица **Всех привязок** виджета Компоновка машины:



#	Target	Expression	Activator	Condition	On Startup	On Event	Periodically	Period
1	form/slot1:reference	'users.admin.widgets.dispenser_' + select({contents$}, "type", "slot", "Slot1")			☑	☐	☐	1 Seconds
2	form/slot2:reference	'users.admin.widgets.dispenser_' + select({contents$}, "type", "slot", "Slot2")			☑	☐	☐	1 Seconds
3	form/slot3:reference	'users.admin.widgets.dispenser_' + select({contents$}, "type", "slot", "Slot3")			☑	☐	☐	1 Seconds
4	form/slot4:reference	'users.admin.widgets.dispenser_' + select({contents$}, "type", "slot", "Slot4")			☑	☐	☐	1 Seconds
5	form/slot5:reference	'users.admin.widgets.dispenser_' + select({contents$}, "type", "slot", "Slot5")			☑	☐	☐	1 Seconds
6	form/slot6:reference	'users.admin.widgets.dispenser_' + select({contents$}, "type", "slot", "Slot6")			☑	☐	☐	1 Seconds
7	form/slot1:device	dc()			☑	☐	☐	1 Seconds
8	form/slot2:device	dc()			☑	☐	☐	1 Seconds
9	form/slot3:device	dc()			☑	☐	☐	1 Seconds
10	form/slot4:device	dc()			☑	☐	☐	1 Seconds
11	form/slot5:device	dc()			☑	☐	☐	1 Seconds
12	form/slot6:device	dc()			☑	☐	☐	1 Seconds
13	form/machine_description:text	{.:#description}			☑	☐	☐	1 Seconds

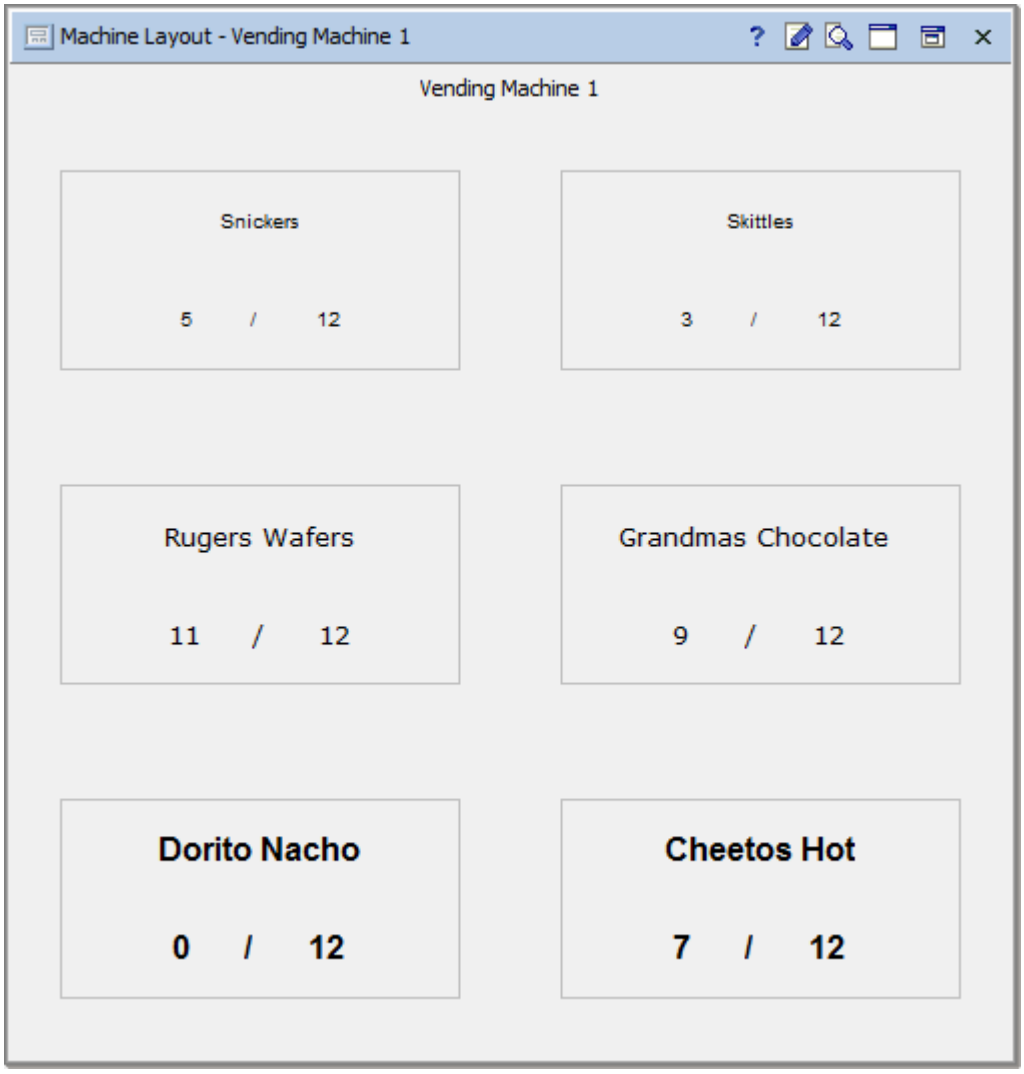
Правильное определение свойства `slot` может быть непростым. Каждый подвиджет диспенсора в слоте должен получать собственное значение статических параметров (`Slot1`, `Slot2` и т.д.). Однако если мы выбираем компоненты подвиджета в Рабочей форме, мы не видим вкладку **Пользовательские свойства** в окне **Свойства**, потому что у них не определен статический шаблон. Чтобы задать параметр `slot`, надо вручную задать каждому подвиджету параметр `Reference` к пути любого из наших подвиджетов диспенсора. После этого нужно задать параметр `slot` наших подвиджетов как `Slot1`, `Slot2` и т.д.

В таблице Все привязки нашей Компоновки машины привязки, меняющие шаблоны подвиджетов диспенсора (свойства **Ссылка**), должны предшествовать привязке, которая задает внутренние параметры этого подвиджета (`device` и `slot`). В этом случае значения параметров будут получены "правильными" подвиджетами диспенсора.

Тестирование визуализации компоновки машины

Поскольку все виджеты, составляющие систему отрисовки компонентов машины теперь закончены, можно начать их тестирование и использование. Для просмотра компоновки определенной машины выберите действие запуска виджета () **Запуск машины** из контекстного меню вендингового аппарата (или модели в нашем случае).

Вот как выглядит результат:



Что дальше

Описанная система отрисовки вендингового аппарата может быть расширена при помощи:

- Разработки виджетов для различных компоновок машин и типов диспенсоров
- Отрисовки машин более сложной структуры (например, машины с несколькими отделениями, содержащими заменяемые мультидиспенсорные держатели)
- Диспенсоры с цветовым кодированием в зависимости от количества оставшихся запасов
- Разработка профессионально отрисованных компоновок виджетов

18.23 Управление таблицей внешней базы данных SQL

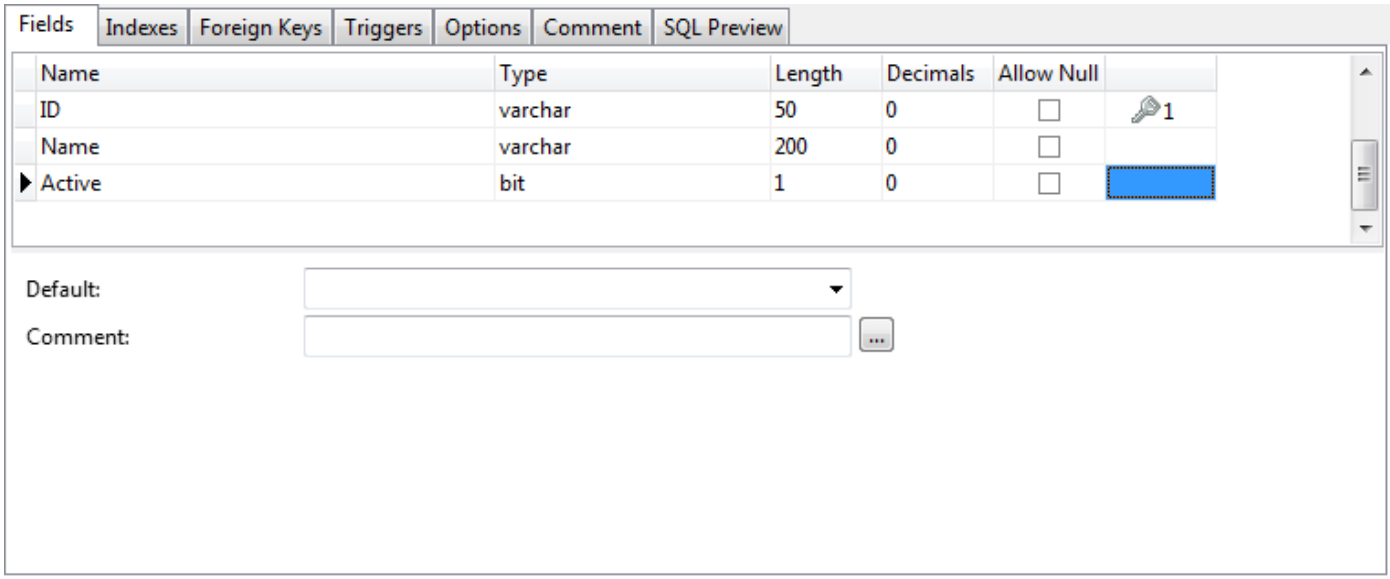
При инсталляции AtomMind в среде сложной ИТ-инфраструктуры предприятия необходимо управлять и использовать данные, хранимые во внешней базе данных. Этот урок показывает, как подключиться к внешей базе данных SQL и создать пользовательский интерфейс для поиска, фильтрации, создания, редактирования и удаления записей.

В этом уроке мы будем управлять внешней базой данных MySQL, содержащей данные RFID карт. Эту базу данных карт можно использовать для авторизации пользователей через различные устройства контроля доступа на базе RFID. Авторизация может также осуществляться через AtomMind, но описание этой функции не входит в задачи данного урока.

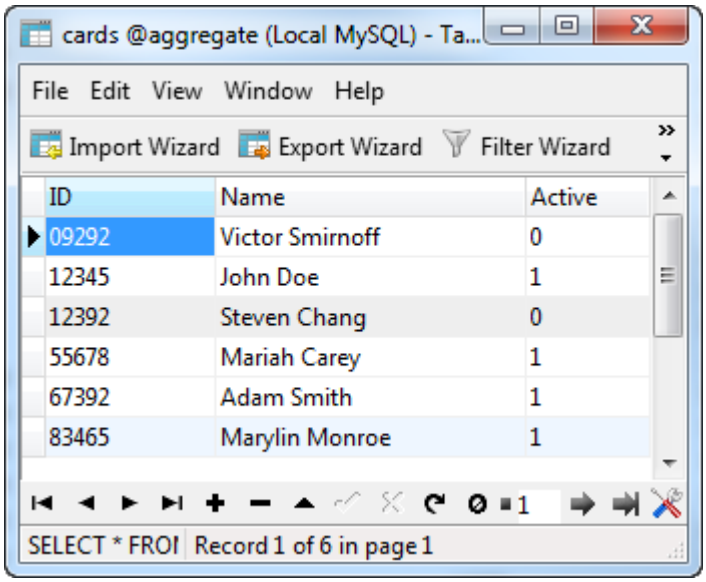
1. Подключение к внешней базе данных

Подключение к внешней базе данных MySQL будет осуществляться через драйвер устройства базы данных SQL 647 AtomMind, который также может подключаться к любому другому типу базы данных через JDBC.

Вот как выглядит структура базы данных (снимок утилиты управления сторонней базой данных):



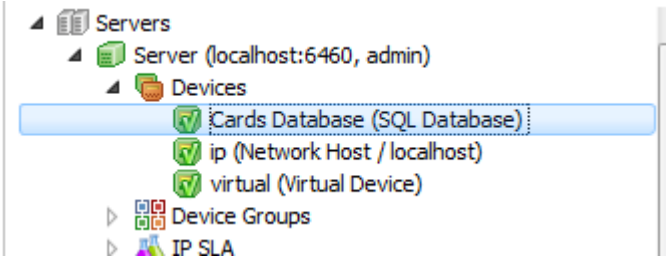
Вот данные (снимок утилиты управления сторонней базой данных):



Во-первых, мы создаем учетную запись устройства в базе данных SQL и определяем необходимые настройки доступа (URL базы данных, имя пользователя и пароль). В результате имеем зеленое устройство с зеленой кнопкой-флажком поверх него (что означает, что устройство онлайн и синхронизировано). См. документацию драйвера База данных SQL 647 для получения более подробной информации.

Как только подключено устройство базы данных, становится доступна функция устройства Выполнить запрос 650. Эта функция позволяет выполнить произвольный запрос выбрать/обновить с пользовательскими параметрами, что критически необходимо для нашего случая.

Подключенная учетная запись устройства базы данных:

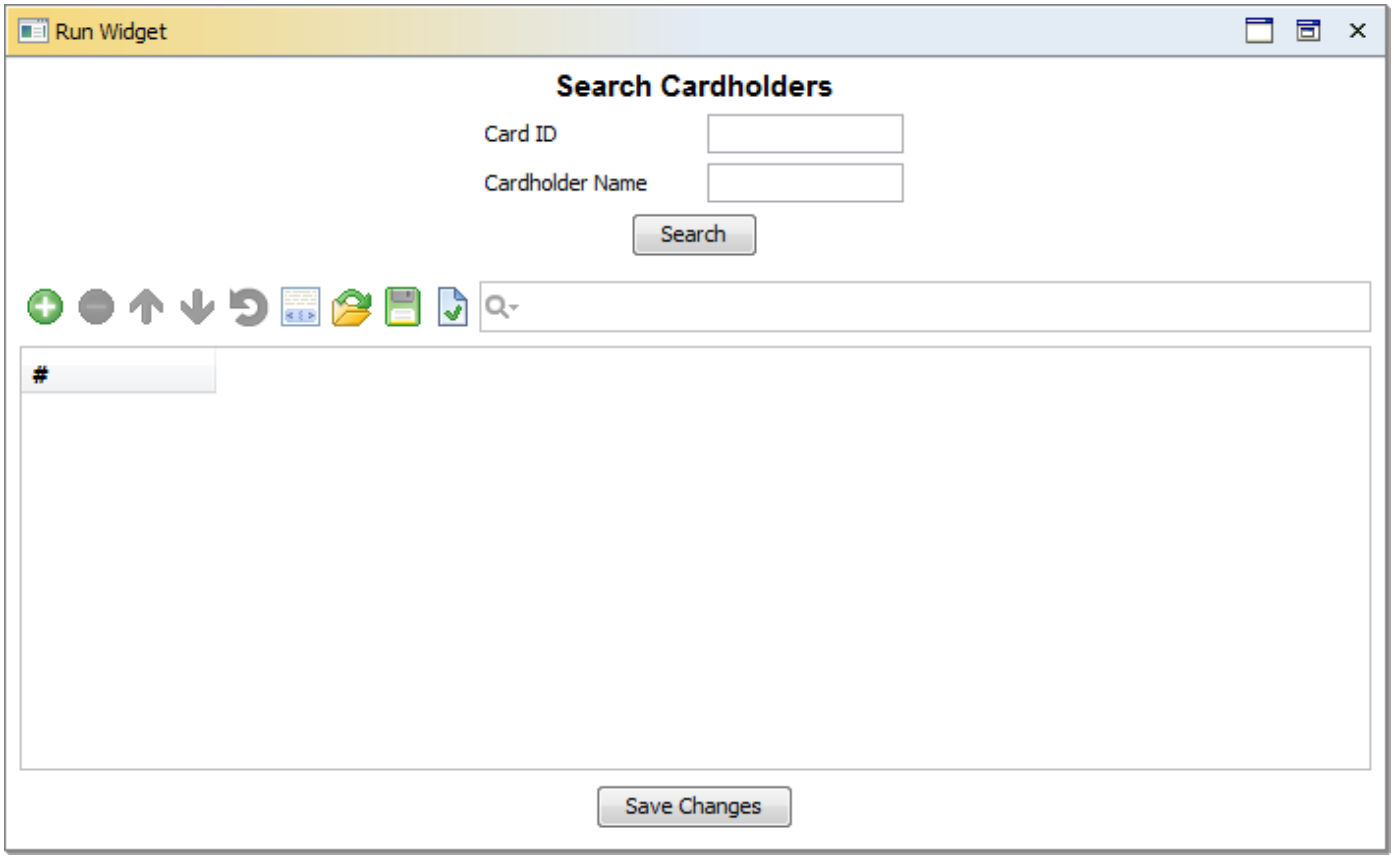


В нашем примере учетная запись устройства базы данных называется `cardsDatabase`, его путь контекста 41 - `users.admin.devices.cardsDatabase`.

2. Дизайн поиска данных и управление пользовательским интерфейсом

На втором этапе нужно создать пользовательский интерфейс, используемый системными операторами для поиска, просмотра и управления RFID картами. Интерфейс реализован как виджет 943. Поскольку интерфейсы в реальности будут в любом случае отличаться, а процесс создания пользовательского интерфеса объясняется в других уроках и статьях, мы не будем описывать все детали процесса визуальной разработки пользовательского интерфейса.

Вот как выглядит форма управления картами:



Основные элементы пользовательского интерфейса включают:

- Панель Топ Поиск с двумя текстовыми полями (идентификатор карты и имя держателя карты) и кнопка Поиск
- Панель Нижние Карты с Редактором таблиц данных для просмотра/редактирования карт и кнопка Сохранение изменений

Этот пользовательский интерфейс может сильно отличаться в других задачах управления внешними базами данных.

3. Привязка интерфейса к базе данных

На этом этапе нужно сделать так, чтобы наша пользовательская форма могла загружать и сохранять данные из/в нашу таблицу базы данных Карт.

У нашего виджета есть только две привязки 1295.

КНОПКА ПОИСКА ПРИВЯЗОК

Первая привязка работает при щелчке по кнопке Поиск (как указано активатором `form/searchButton:click@`):

Цель	`form/cards:dataTable`
Выражение	`addColumns(adjustRecordLimits(callFunction("users.admin.devices.cardsDatabase", "executeQuery", "select * from cards where id like ? and name like ?", false, table("<<cardId><S>><<cardholderName><S>>", "%" + {form/cardId:text} + "%", "%" + {form/cardholderName:text} + "%")), 0, 1000000000), "<originalId><S>", "{id}")`
Активатор	`form/searchButton:click@`
Условие	

Опции On Event

Привязка загружает данные карты путем вызова функции executeQuery из устройства Базы данных SQL (указанного путем контекста users.admin.devices.cardsDatabase) и помещает их в таблицу cards (как указано целью form/cards:dataTable).

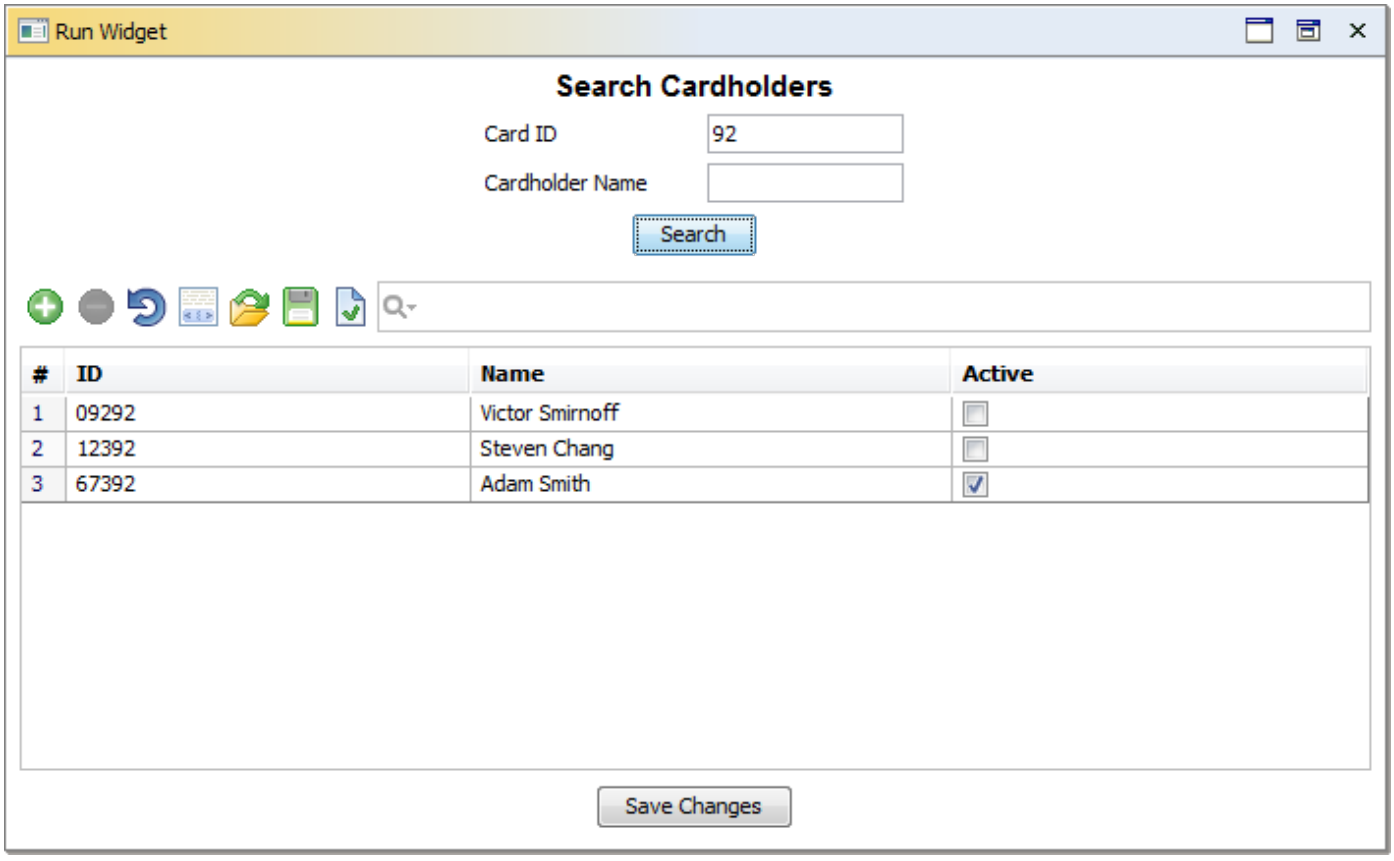
Текст запроса - select * from cards where id like ? and name like ?. Второй параметр false функции выполнения запроса говорит о том, что мы выполняем запрос выбора (не обновления). Третий параметр - это таблица в две колонки, сформированная "на лету" при помощи функции table(). Имя ее поля (cardId и cardholderName игнорируются, поскольку значение первой колонки используется для замещения первого параметра ? в тексте запроса и т.д.).

Конструкция "%" + {form/cardId:text} + "%" добавляет к началу и концу строки поиска символ %, чтобы активировать нестрогое соответствие для SQL-оператора **like** .

adjustRecordLimits(*table* , 0, 1000000000) расширяет минимальное и максимальное количество записей для таблицы, возвращенной запросом, чтобы позволить операторам удалять записи из таблицы карт и добавлять к ней новые записи.

И, наконец, addColumn(*table* , "<originalId><S>", "{id}") добавляет к таблице невидимую колонку originalId и задает ее значения, равные значениям в колонке id. Скрытая колонка позволит обновлять записи базы данных, даже если были отредактированы идентификаторы карт.

Вот как выглядит форма с загруженными и отфильтрованными данными:



ПРИВЯЗКА КНОПКИ ЗАПИСИ

Пивязка кнопки записи проста. Она выполняет функцию processChanges() в компоненте Редактор таблицы данных. См. в документации Обработка Функции Изменений[98] для получения более подробной информации.

Свойства привязки:

Цель form/cards:processChanges()

Выражение

Активатор form/saveButton:click@

Условие

Опции On Event

4. Конфигурирование обработки записей Добавлено/Изменено/Удалено

Функция Изменение Процесса Редактора Таблиц Данных просматривает списки добавленных, измененных и удаленных записей и оценки специального выражения для каждой такой записи.

Таким образом, необходимо сконфигурировать три свойства нашего компонента Редактор Таблиц Данных Виджета (таблица Карты):

- Выражение обработки добавленных записей
- Выражение обработки измененных записей
- Выражение обработки удаленных записей

ОБРАБОТКА ДОБАВЛЕННЫХ ЗАПИСЕЙ

Вот **Выражение обработки добавленных записей** нашего виджета:

```
cell(callFunction("users.admin.devices.cardsDatabase", "executeQuery", "insert into cards (ID, Name, Active) values (?,?,?)", true, dt()), "rows") > 0
```

Это выражение вызывает ту же функцию `executeQuery`. Однако текст запроса другой: `insert into cards (ID, Name, Active) values (?,?,?)`

Второй параметр `true` функции `executeQuery` говорит о том, что мы пытаемся обновить запрос.

Третий параметр - это данные добавленной записи, "завернутые" в таблицу с одной строкой и указанные функцией `dt()` (поскольку это таблица по умолчанию 120 во время расчета **Выражения обработки добавленных записей**).

Таким образом, значение из первой колонки, т.е. идентификатор карты, будет использоваться для замены первого параметра `?` и т.д.

Конструкция `cell(executeQueryResult, "rows") > 0` гарантирует, что хотя бы одна запись была вставлена запросом. Обрабатываемая в данный момент запись удаляется из списка добавленных записей в этом случае.

ОБРАБОТКА УДАЛЕННЫХ ЗАПИСЕЙ

Вот **Выражение обработки удаленных записей**:

```
cell(callFunction("users.admin.devices.cardsDatabase", "executeQuery", "delete from cards where ID = ?", true, dt()), "rows") > 0
```

Выражение очень похоже на **Выражение обработки добавленных записей**. В этом случае используется только параметр идентификатора карты.

ОБРАБОТКА ОБНОВЛЕННЫХ КАРТ

Выражение обработки измененных записей несколько более сложное:

```
cell(callFunction("users.admin.devices.cardsDatabase", "executeQuery", "update cards set id = ?, name = ?, active = ? where id = ?", true, dt()), "rows") > 0 ? set(dt(), "originalId", 0, {id}) : false
```

Его первая часть отправляет обновленный запрос в базу данных, используя значение четвертого поля `originalId` как ключ для сравнения. Это необходимо, потому что первое видимое поле идентификатора карты может редактироваться операторами и все же должно будет точно соответствовать записям.

Часть `set(dt(), "originalId", 0, {id})` рассчитывается, только если хотя бы одна запись обновляется. В этом случае мы берем потенциально оновленное значение поля `id` и снова сохраняем его в поле `originalId`, чтобы можно было дальше менять записи.

Заключение

Форму, подобную вышеобозначенной, можно создать приблизительно за час. Однако она обеспечивает функциональность комплексного просмотра и управления для любой пользовательской таблицы, используя полностью настраиваемый пользовательский интерфейс.

18.24 Сохранение истории во внешней базе данных

Иногда необходимо сохранить историю переменных во внешей базе данных. Вот несколько шагов, позволяющих осуществить это: установите желаемую базу данных с таблицей для хранения значений, подключите эту базу данных к AtomMind Server и создайте Модель, которая будет слушать изменения переменных и хранить их в этой таблице.

1. Установка внешней базы данных

Мы не будем описывать аспекты управления базой данных, предполагая, что вы с этим знакомы.

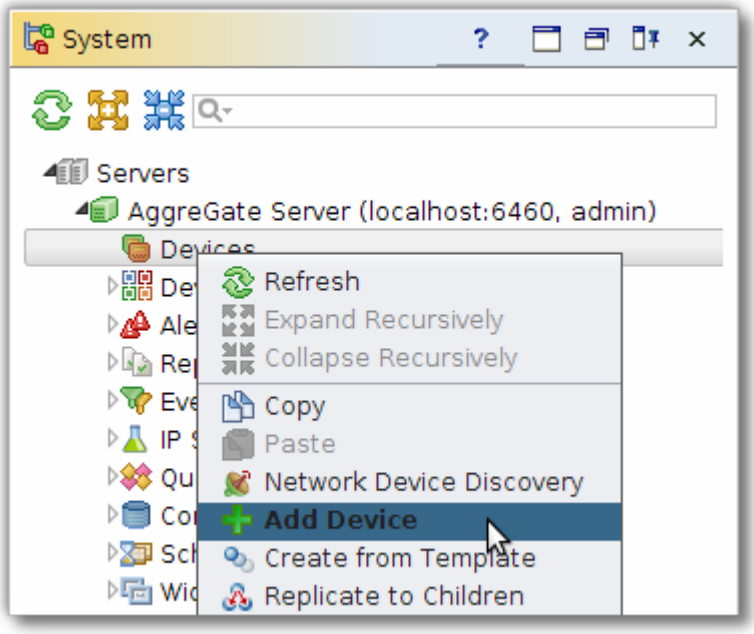
Сначала нужно получить базу данных. Можно создать новую базу данных или использовать существующую. Предположим, что имя базы данных - `historyDB`. Затем необходимо создать таблицу, которая будет сохранять историю переменных. Пусть у таблицы будет имя `variable_history` и следующие поля:

Имя	Тип	Цель
id	BIGINT	Используется для хранения идентификатора событий изменения переменных. Должен быть типом "число", может содержать длинные значения.
name	CHAR/VARCHAR	Содержит имя пепременной.
time	TIME/TIMESTAMP	Время изменения переменной.
value	FLOAT/DOUBLE	Само значение переменной.

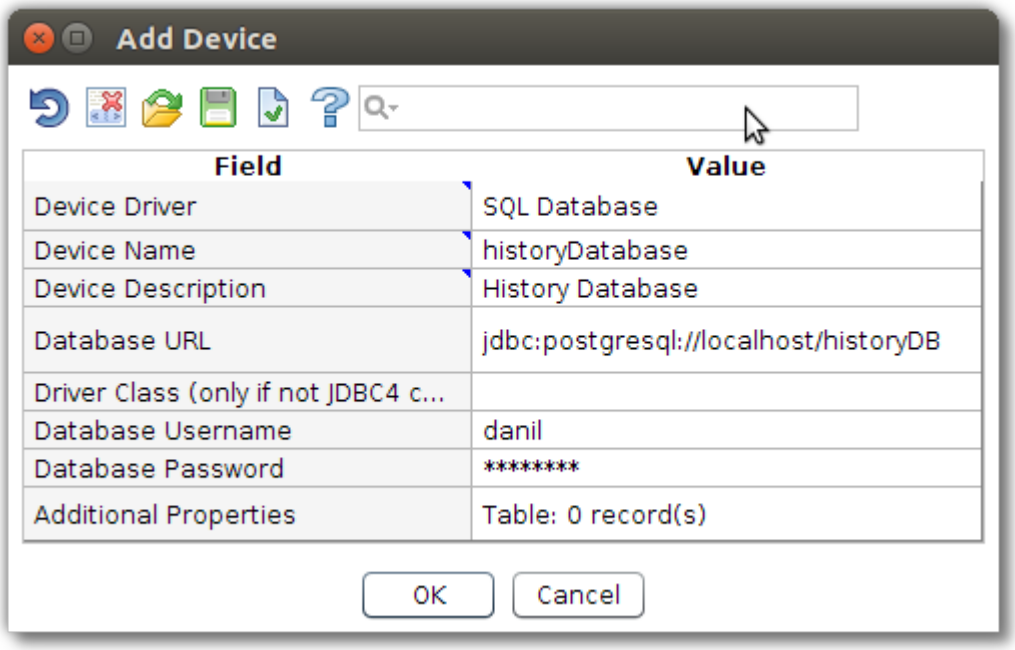
Предлагаются все типы полей, нужно определить соответствующий тип базы данных для каждого поля в соответствии с его целью.

2. Подключение к внешней базе данных

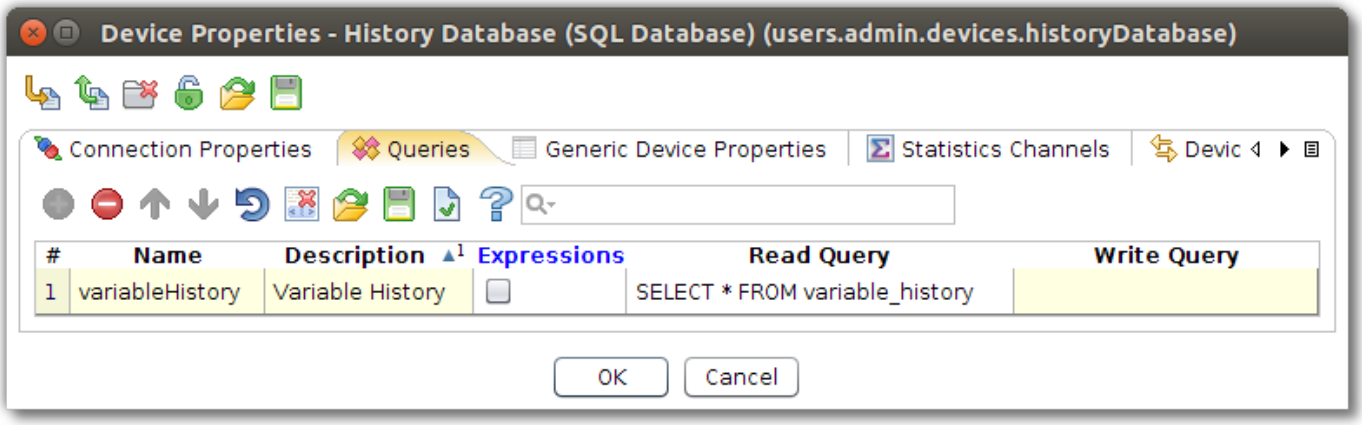
Теперь давайте подключим базу данных к AtomMind Server. Для этого нужно создать для нее учетную запись устройства[497]. Откройте всплывающее меню для узла `Device` в `System Tree` и кликните по пункту `Add Device`.



В диалоге `Add Device` выберите `База данных SQL`[647] как `Device Driver`. Должны появиться следующие поля:



Введите Name и Description для учетной записи устройства, как показано выше. Затем правильно заполните Database URL, эта строка подключения JDBC должна указывать на вашу базу данных. Строка подключения зависит от вашей DBMS. Можно начать редактирование строки, используя один из шаблонов в комбинированной ячейке поля. Для большинства современных баз данных можно пропустить поле Driver Class. Введите учетные данные Username и Password базы данных и нажмите OK. Если параметры верны и соединение установлено, в системе появится учетная запись устройства History Database и отобразится ее диалог Device Properties:

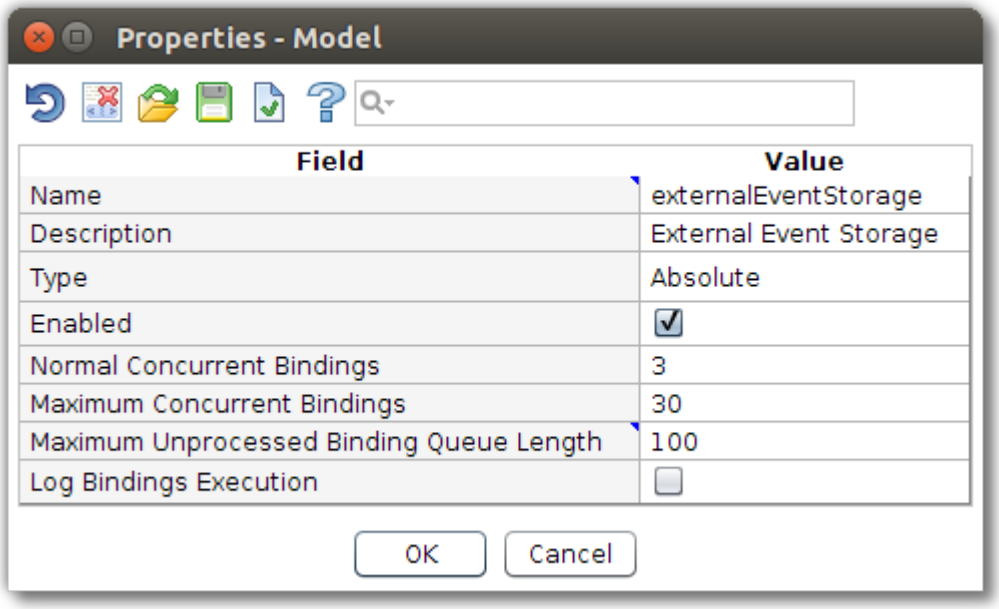


Теперь перейдите во вкладку Queries и добавьте новую запись путем нажатия кнопки Add Row ( ). Заполните все поля, как показано выше. После сохранения изменений при помощи кнопки OK наше устройство History Database получает настройку variableHistory, которая показывает содержание таблицы variable_history. Вы можете посмотреть значение variableHistory в окне настроек устройства. Оно открывается двойным щелчком по узлу устройства или путем выбора пункта Manage Device во всплывающем меню.

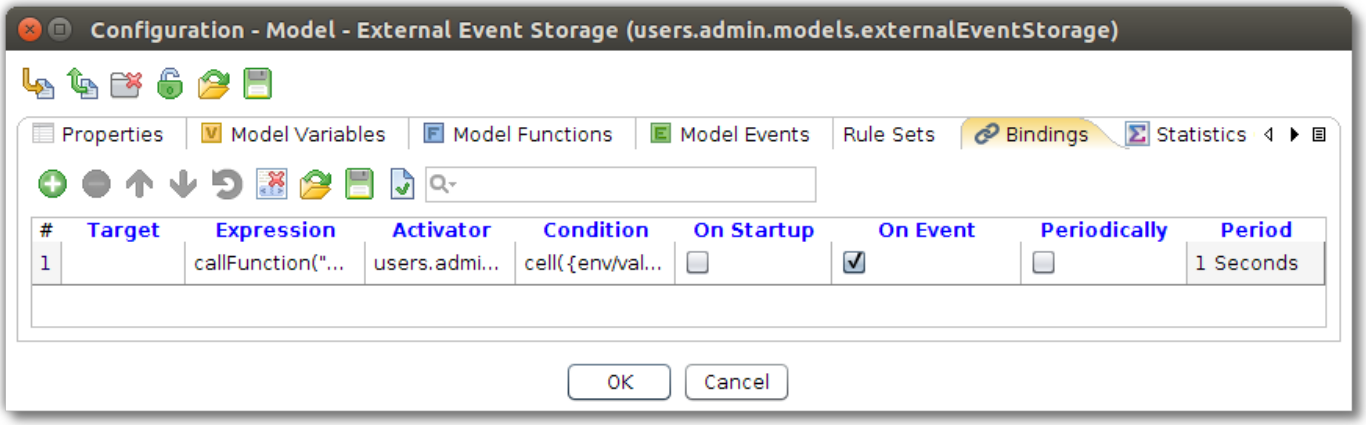
После создания и установки устройства можно взаимодействовать с базой данных, выполняя запросы для учетной записи устройства History Database.

3. Создание модели для извлечения данных

Мы будем использовать модель для прослушивания изменений переменных и записывать данные о событиях в базу данных. Поэтому нужно создать Модель. Откройте всплывающее меню для узла Models и выберите пункт Create, подобный созданию Device Account. Введите Name и Description в диалоге Properties, как показано ниже, и нажмите OK:



Откроется диалог `Configuration`. Перейдите во вкладку `Bindings` и нажмите кнопку `Add Row` (). К таблице Привязки[736] будет добавлена новая привязка. Теперь нужно настроить поля привязки:



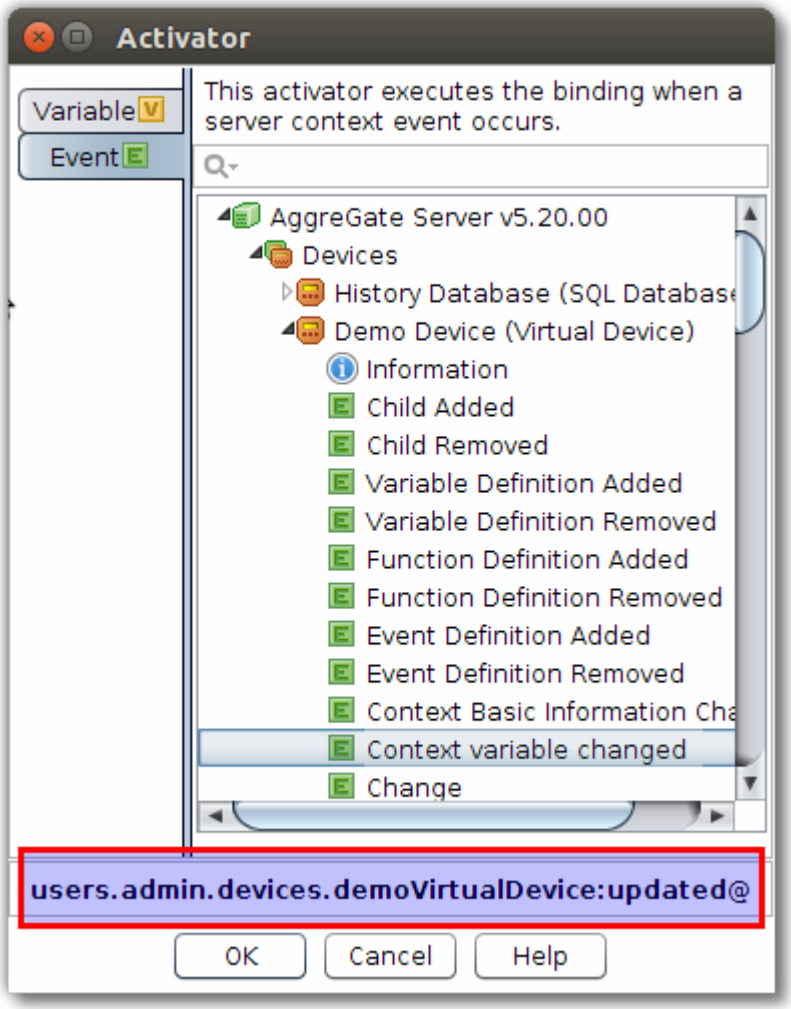
Установите ее поля `On Startup`, `On Event` и `Periodically`, как показано выше.

ЦЕЛЬ

Оставьте это поле пустым. Мы не хотим добавлять данные события к любой цели[737]. Вместо этого мы просто выполняем запрос обновления базы данных в поле `Expression`.

АКТИВАТОР

Сделаем так, чтобы наша привязка прослушивала событие изменения переменных. Щелкните по кнопке `[...]` поля Активатор[739]. В открывшемся диалоге `Activator` откройте `Event tab`, затем локализуйте желаемое устройство для прослушивания и дважды щелкните на событии `Context variable changed`. Внизу должна появиться ссылка на это событие, например: `users.admin.devices.demoVirtualDevice:updated@`. Теперь сохраните изменения путем нажатия кнопки `OK`.



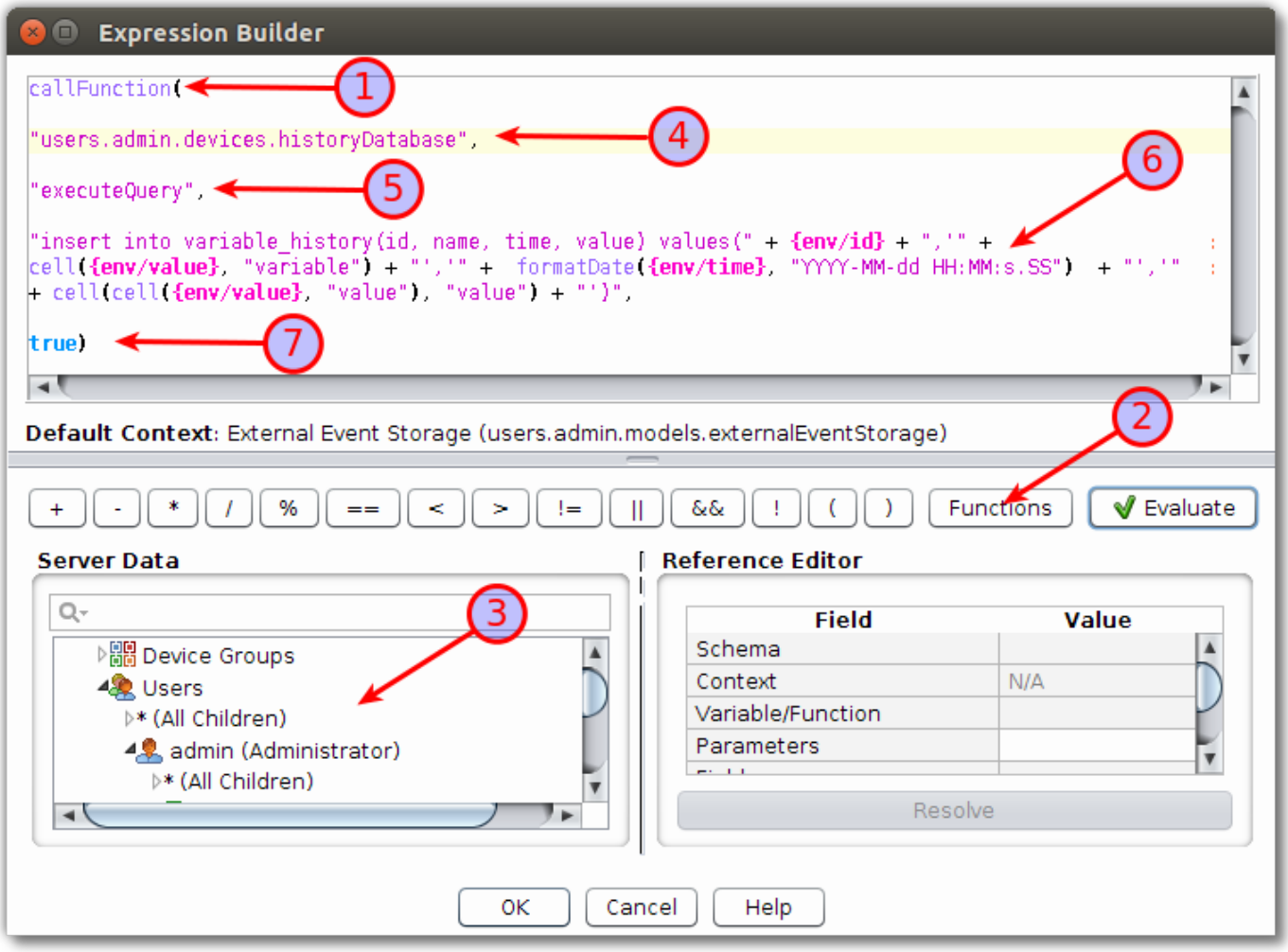
После этого наша привязка будет выполняться каждый раз при изменении любых переменных в `Demo Device`. Для расчета нашей привязки по конкретным изменениям нужно задать условие.

УСЛОВИЕ

Условие 740 будет проверять каждое событие изменения переменных. Когда условие `false`, событие будет игнорироваться. Поэтому нужно настроить выражение условия на `cell({env/value}, 'variable') == 'random'`. Это означает, что выражение привязки будет рассчитываться, только когда событие изменения переменных принадлежит переменной `random` в `Demo Device`.

ВЫРАЖЕНИЕ

Выражение 738 рассчитывается каждый раз при вызове привязки и выполнении условия. Откройте Редактор выражений 404 этого поля путем щелчка по кнопке `[...]`. В открывшемся диалоге `Expression Editor` задайте выражение для выполнения запроса обновления базы данных. Он выполняется путем вызова функции `executeQuery` нашей учетной записи устройства `History Database`. Запрос добавит новую запись к таблице `variable_history`.



Мы будем использовать функцию языка выражений 124 `callFunction(String context, String function, Object parameter1, Object parameter2, ...)` для вызова функции `executeQuery` устройства (1) `History Database`. Можно найти шаблон этой функции в диалоге, который открывается кнопкой (2) `Functions`. Параметры следующие:

- `context` - путь строки к контексту, откуда вызывается функция. Можно локализовать устройство `History Database` в дереве (3) `Server Data`, затем открыть всплывающее меню в соответствующем узле и выбрать действие `Copy`. После этого можно вставить путь устройства в текст выражения. Поэтому путь к устойству `History Database` должен быть `users.admin.devices.historyDatabase` (4)
- `function` - имя вызывающей функции. Оно должно быть `executeQuery` (5). Эта функция 70 принадлежит базе данных SQL `History Database` context 41.
- `parameter1 (query)` - первый параметр вызывающей функции. Для вызывающей функции `executeQuery` это строка запроса. Поэтому наш запрос должен быть следующий: `insert into variable_history(id, name, time, value) values(..., ..., ..., ...)`. Реальные значения должны быть введены вместо `'...'`. Мы получаем реальные значения из среды выражения 123. В нашем случае он будет хранить параметры события измененной переменной, потому что мы настраиваем эту привязку для расчета при событии. `{env/id}` ссылается 117 на идентификатор события. `cell({env/value}, "variable")` получает имя переменной из данных событий. `formatDate({env/time}, "YYYY-MM-dd HH:MM:s.SS")` берет форматы и время событий в приемлемую временную строку базы данных . `cell(cell({env/value}, "value"), "value")` получает измененное значение из данных событий, берет таблицу, которая является измененной переменной `value` и получает поле `value` значения переменной (6).
- `parameter2 (update)` - второй параметр `executeQuery` указывает, является ли этот запрос запросом обновления. В нашем случае это `true`, т.е. запрос должен возвращать любое значение (7).

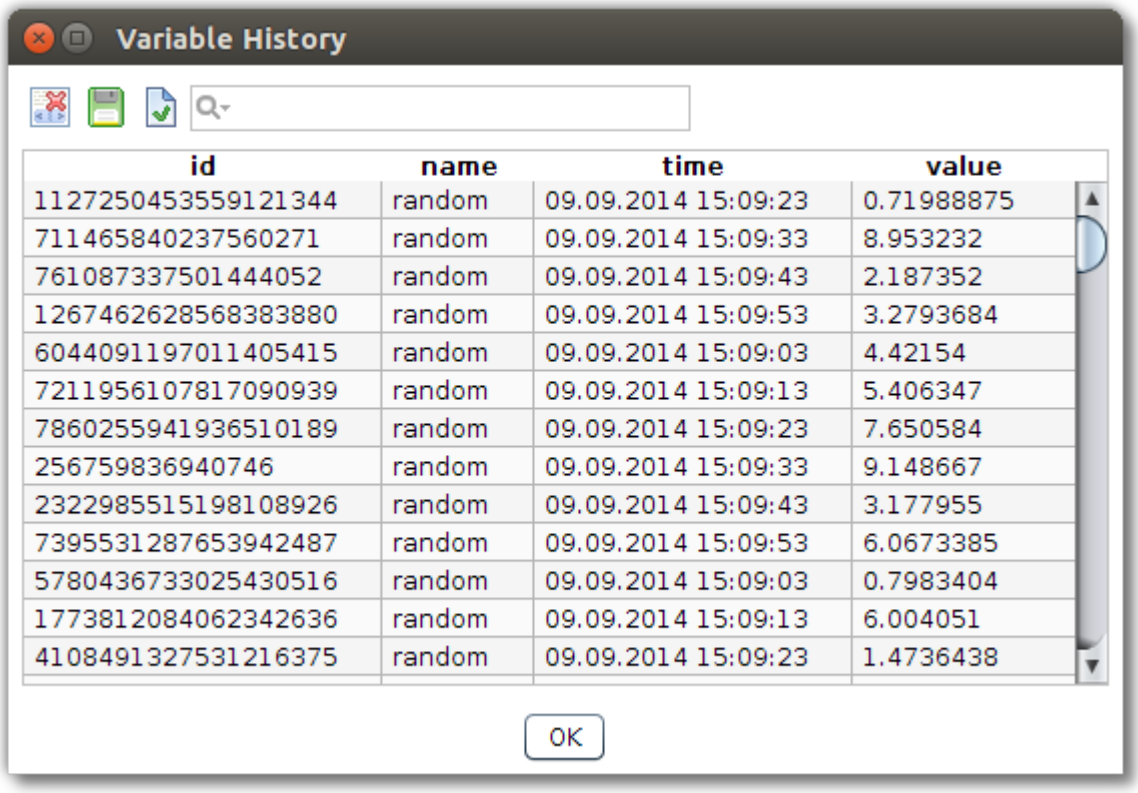
Примените выражение путем нажатия кнопки `OK` и нажмите кнопку `OK` в предыдущем диалоге `Model Configuration` для сохранения наших изменений привязок в модели.

Теперь, если все сделано правильно, изменения переменной `Demo Device's random` начнут сохраняться в таблице `value_history` базы данных `historyDB`.

Можно проверить, если ли ошибки выполнения привязки, просматривая относящиеся к модели события External Event Storage. Просто выберите Monitor Related Events из всплывающего меню модели в System tree. Открывшийся Журнал событий можно использовать для отладки в режиме реального времени.

4. Просмотр результатов

Наконец, вы можете изучить свои результаты, открывая настройки History Database. Дважды щелкните на узле History Database в System Tree. Настройка Variable History должна содержать примерно следующие данные:



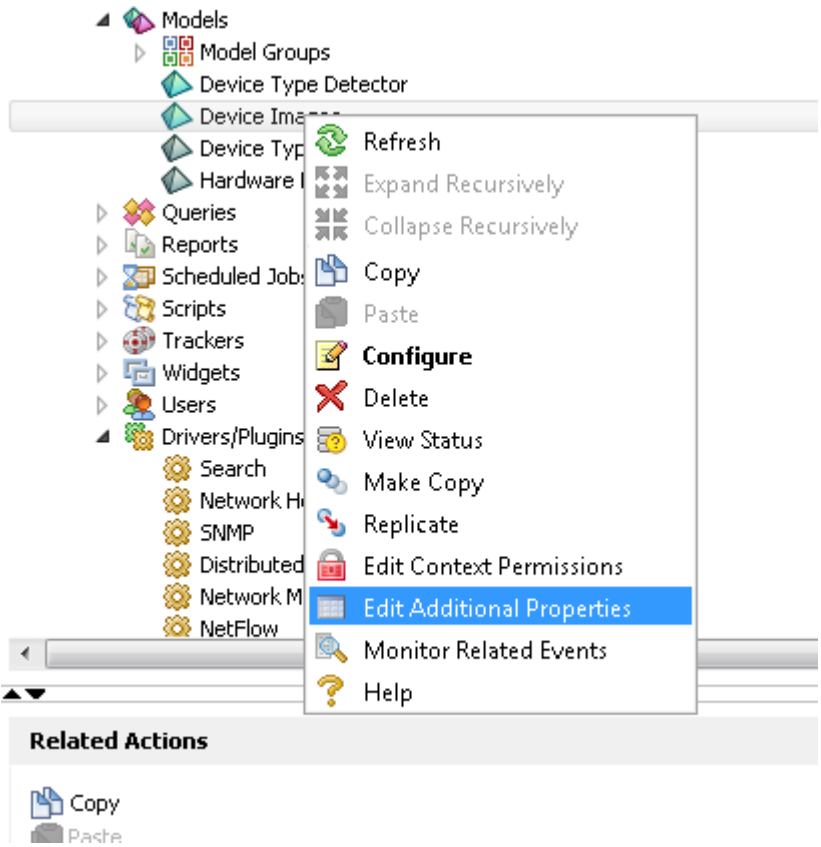
18.25 Использование собственных изображений на кар

AtomMind Server поддерживает добавление ваших собственных изображений на карты. Эта статья описывает, как добавить собственное SVG изображение и настроить виджет для использования изображений на карте.

1. Добавление собственных нестандартных изображений в модель изображений устройств

Изображения устройств - это стандартная модель, используемая для хранения изображений и их отображения на картах. Если вы не можете найти ее в контейнере Модели, тогда создайте ее из Ресурсов 208.

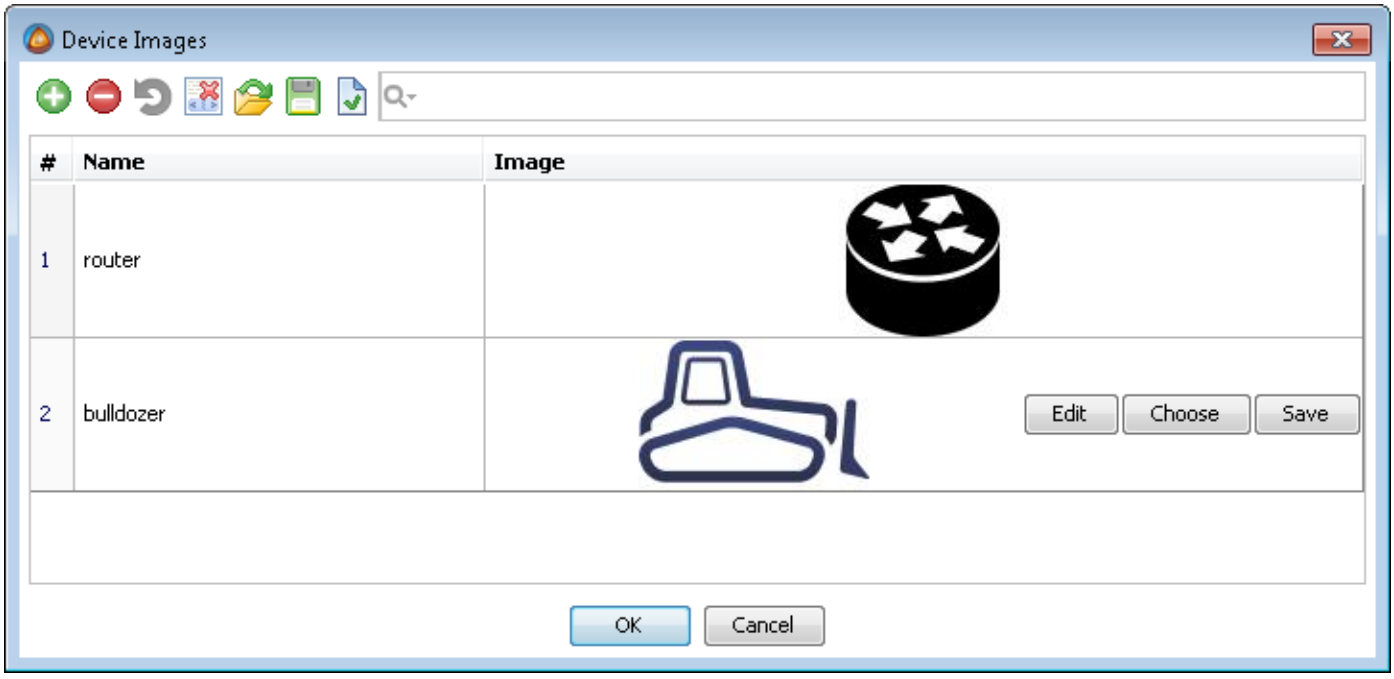
Разверните контейнер Модели в Системном дереве. Кликните правой кнопкой по модели Изображения устройств, выберите Редактировать дополнительные свойства.



Отключите режим "только для чтения" при помощи кнопки `Переключить режим "Только для чтения"`.

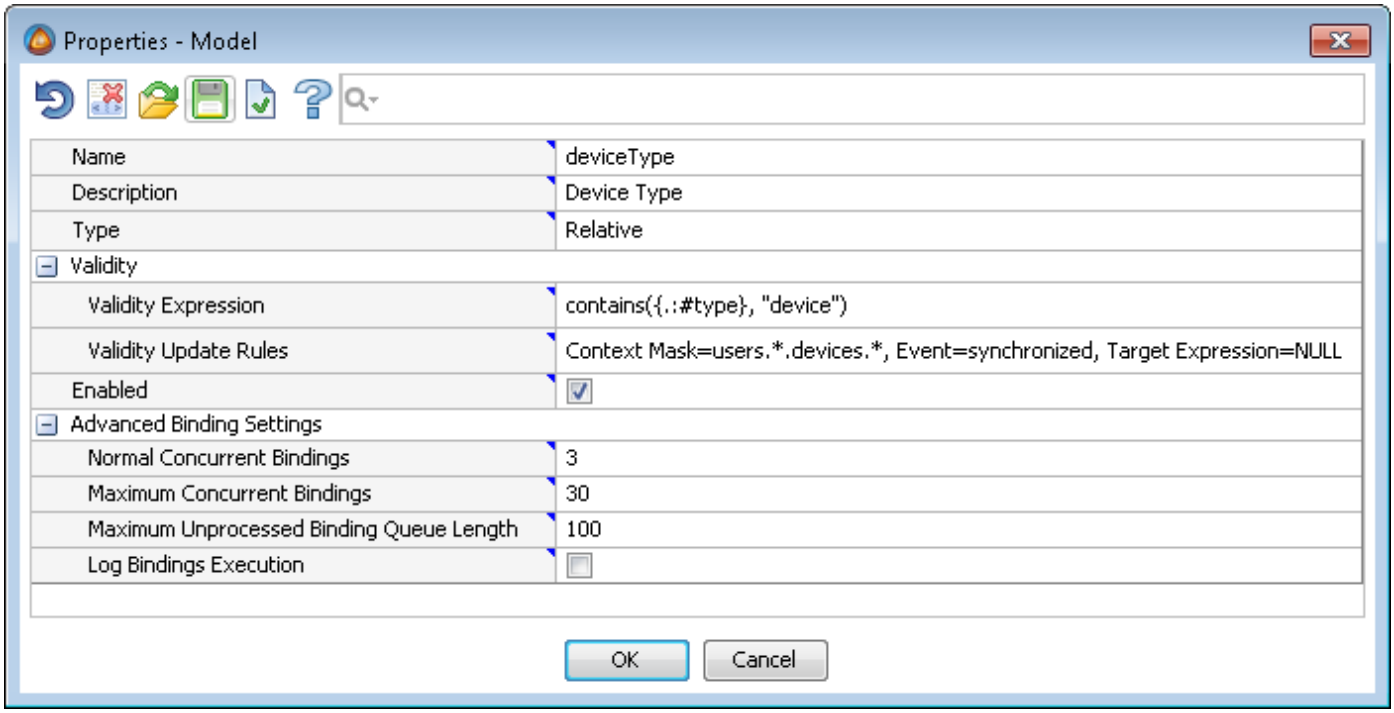
Затем нажмите на кнопку `Кликните для открытия...` и добавьте собственное SVG изображение, используя кнопку `Добавить строку` (⊕), и задайте имя изображения.

Добавьте изображение с именем `Бульдозер`.



2. Создание относительной модели для определения пользовательского типа устройства

Дважды кликните по контейнеру Модель для создания новой модели. Затем заполните поля, как показано ниже и нажмите кнопку OK.



Теперь перейдите во вкладку Переменные модели и добавьте новую переменную, нажав кнопку Добавить строку ( ). Введите Имя и Описание новой переменной, выберите флажок Доступно для записи и укажите формат переменной, кликнув по полю Формат.

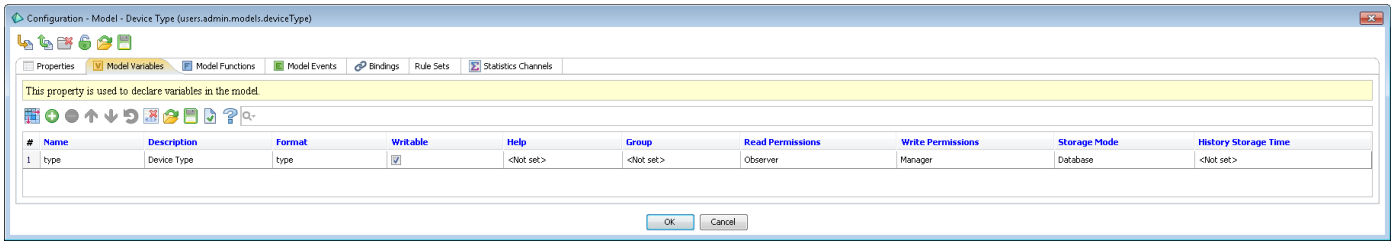
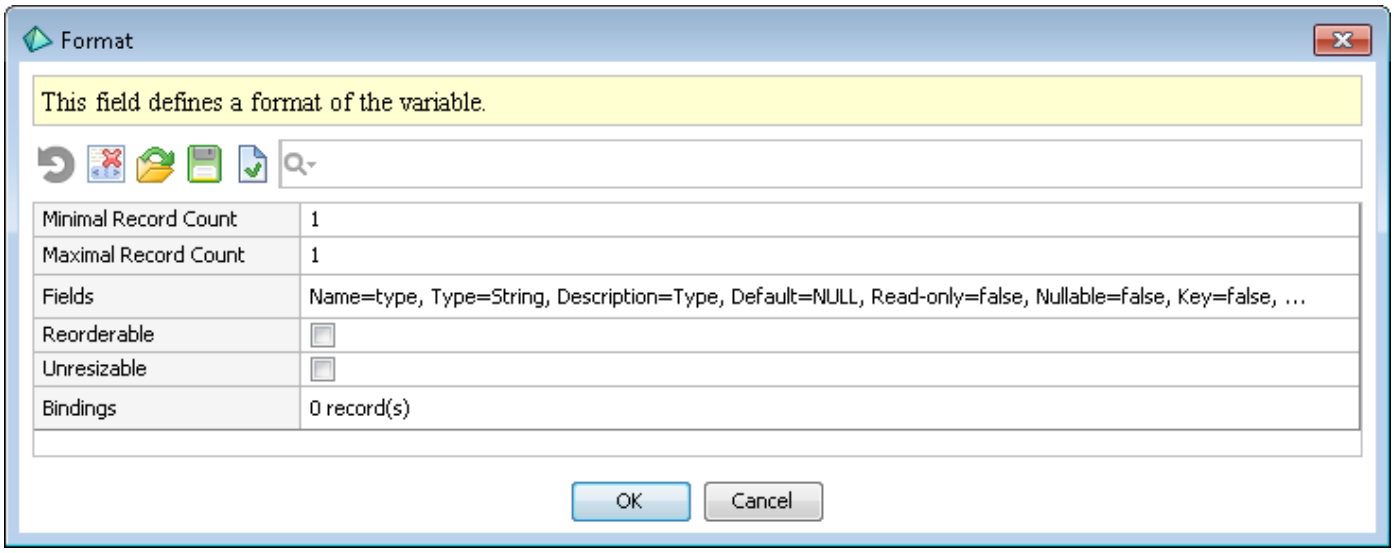
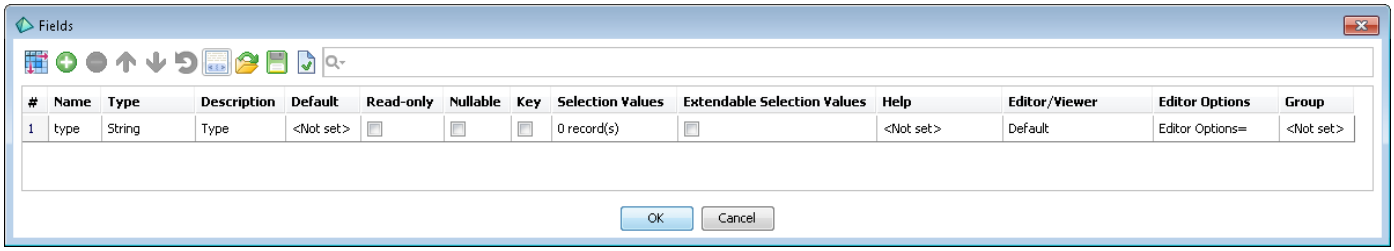


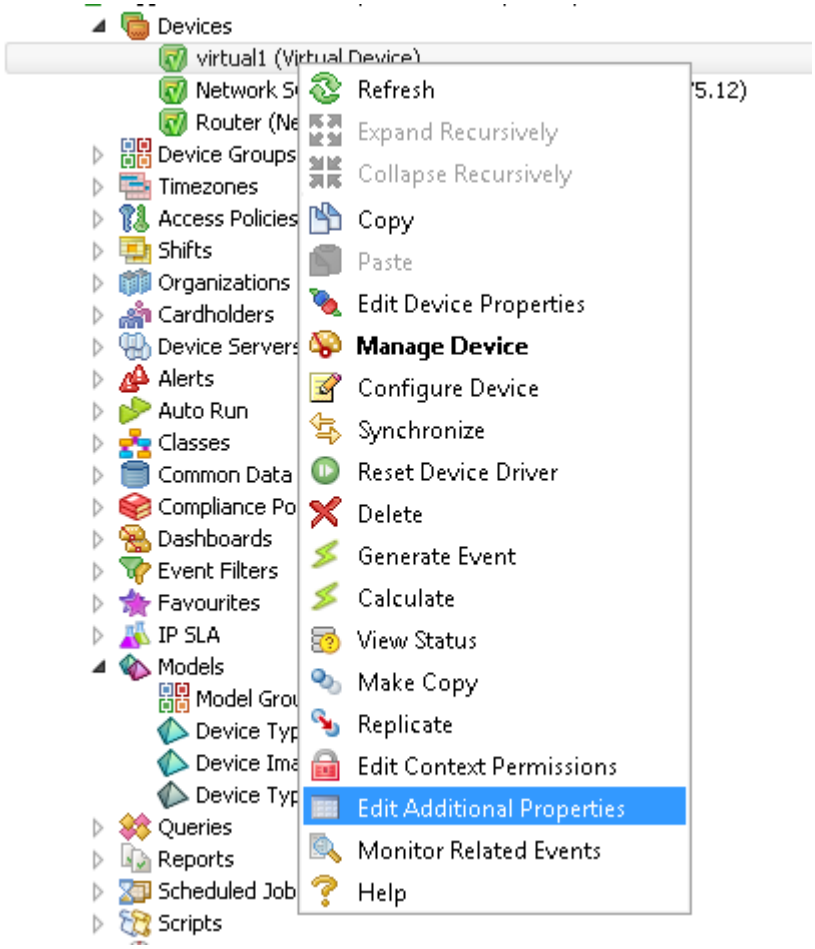
Таблица переменной должна иметь одну строку и один столбец. Задайте Минимальное количество записей и Максимальное количество записей равные одному. Нажмите на ячейку Поля.



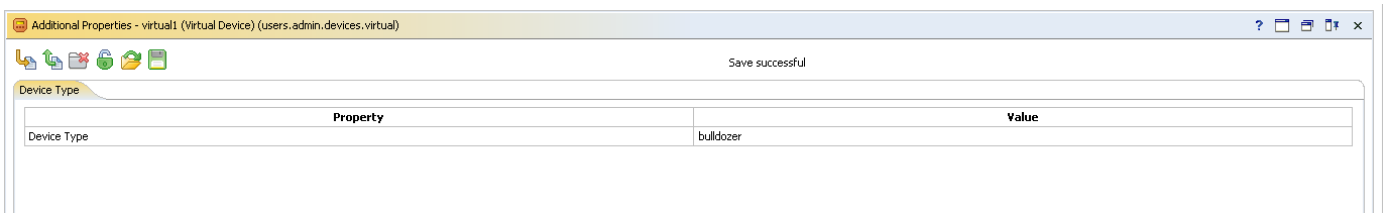
Нажав на кнопку Добавить строку (⊕), создайте поле и введите тип, Строка и Тип в соответсвующих полях Имя, Тип и Описание.



Переменная тип должна иметь устройство на сервере AtomMind Server. Определите тип устройства. Найдите свое устройство в Системном дереве, кликните по нему правой кнопкой и выберите Редактировать дополнительные свойства.

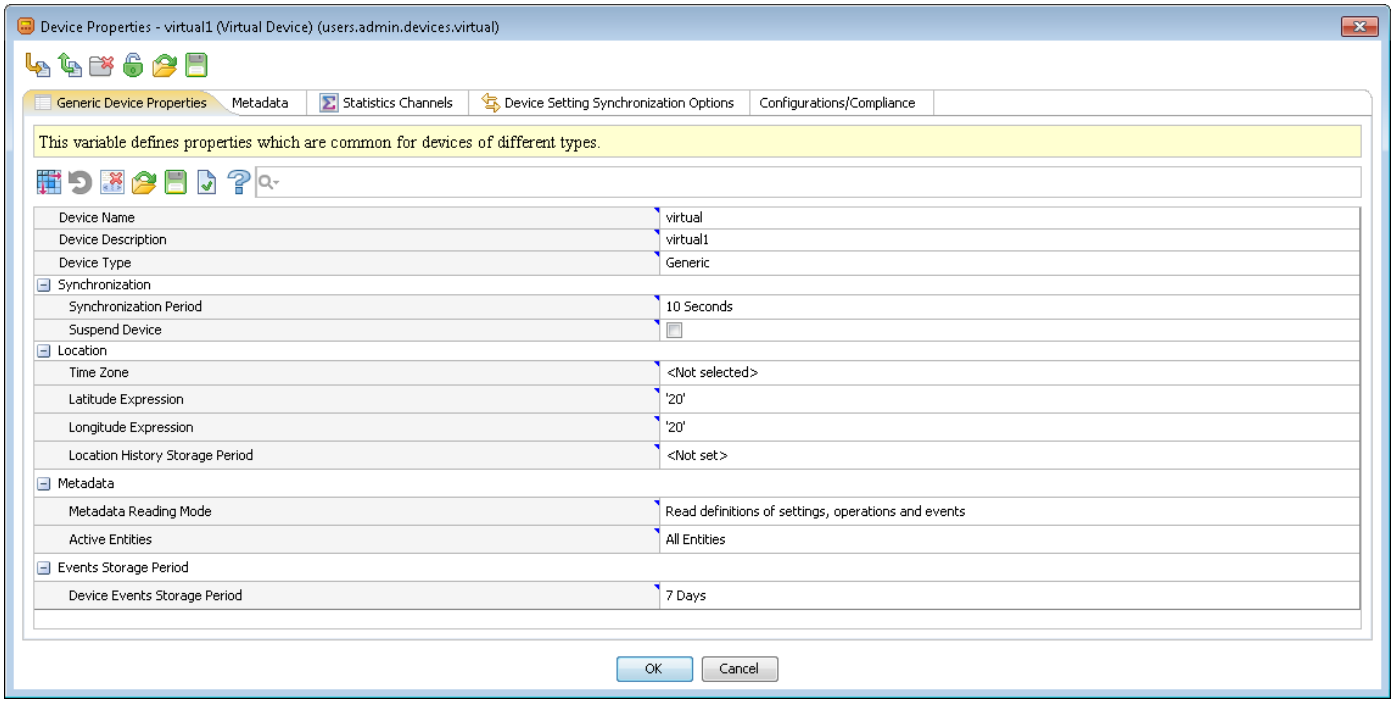


Отключите режим "только для чтения" при помощи кнопки Переключить режим "только для чтения" и заполните поле Тип устройства. У него должно быть то же имя, что имя изображения в модели Изображения устройства.



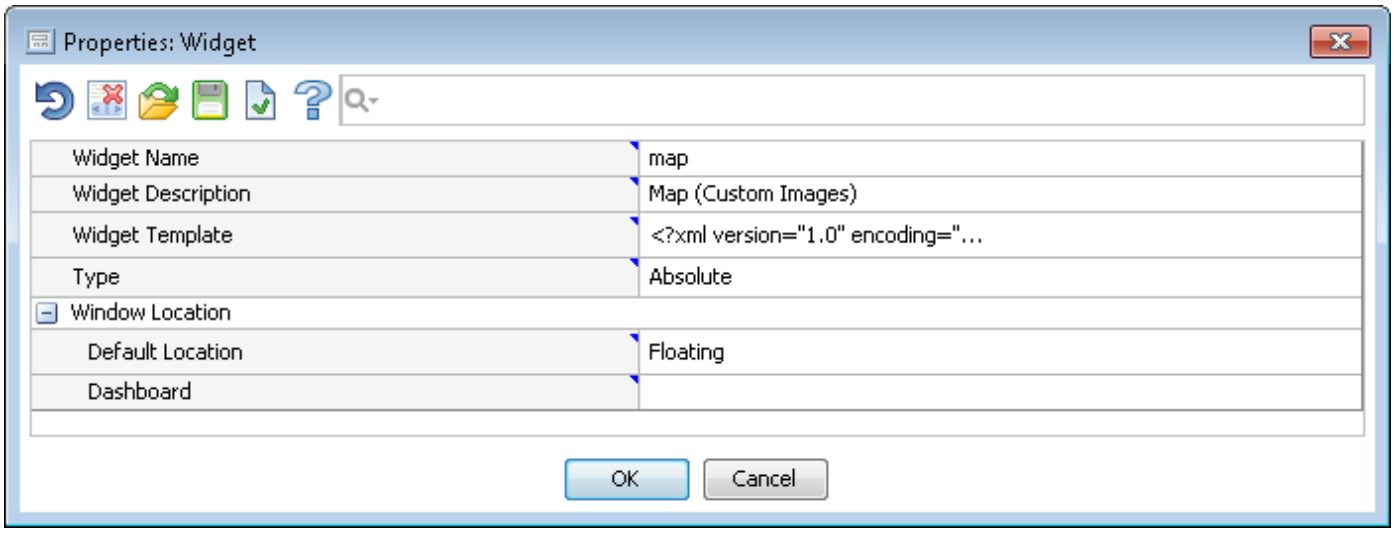
3. Определение координат устройства

Для отображения устройства на карте задайте его широту и долготу. Кликните правой кнопкой по устройству и выберите Редактировать свойства устройства, затем введите Выражение широты и Выражение долготы во вкладке Общие свойства устройств. Нажмите кнопку OK для сохранения изменений. Эту операцию нужно повторить для каждого устройства, чтобы они могли отображаться на карте.

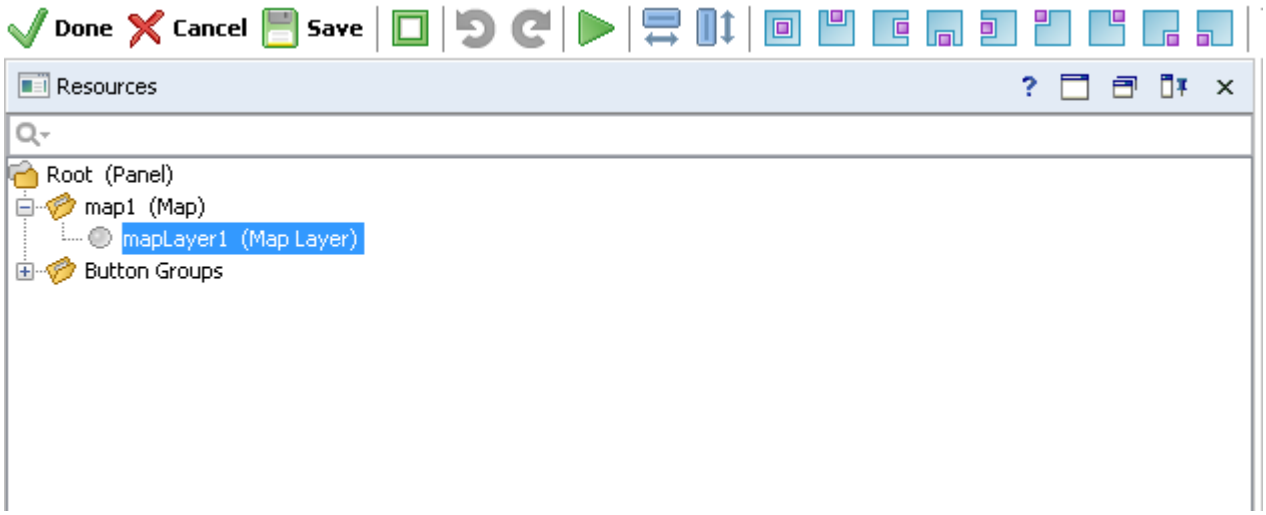


4. Создание виджета с компонентом "Карта"

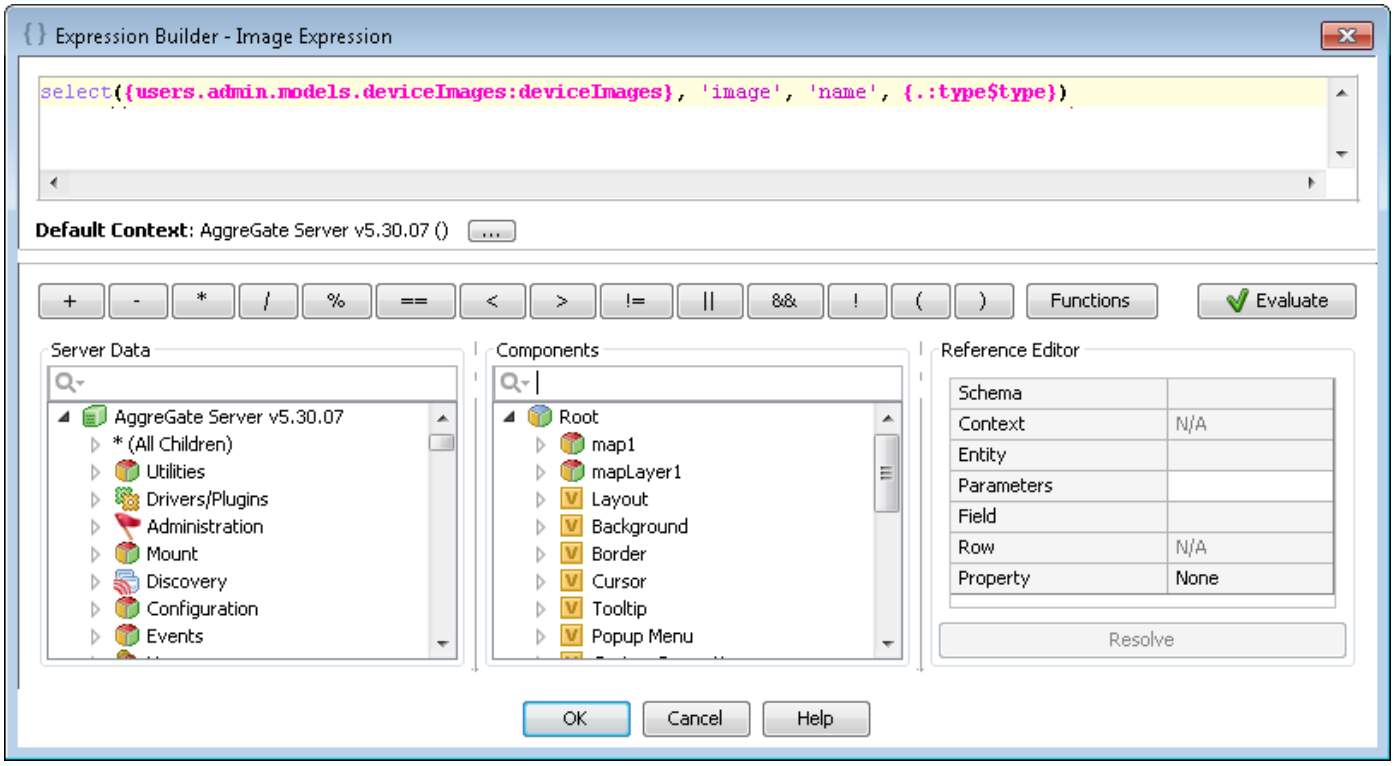
Дважды кликните по узлу Виджеты, заполните поля Имя и Описание и кликните OK.



В Редакторе виджетов перетащите компонент Карта в панели Корень. Выберите узел Слой карты в окне Ресурсы.



Затем выберите вкладку Топология в свойствах Слои карты и измените свойство Выражение изображения в соответствии с приведенным ниже:



Для корректировки размера изображения на Карте используйте свойство Пропорция маркера во вкладке Топология.

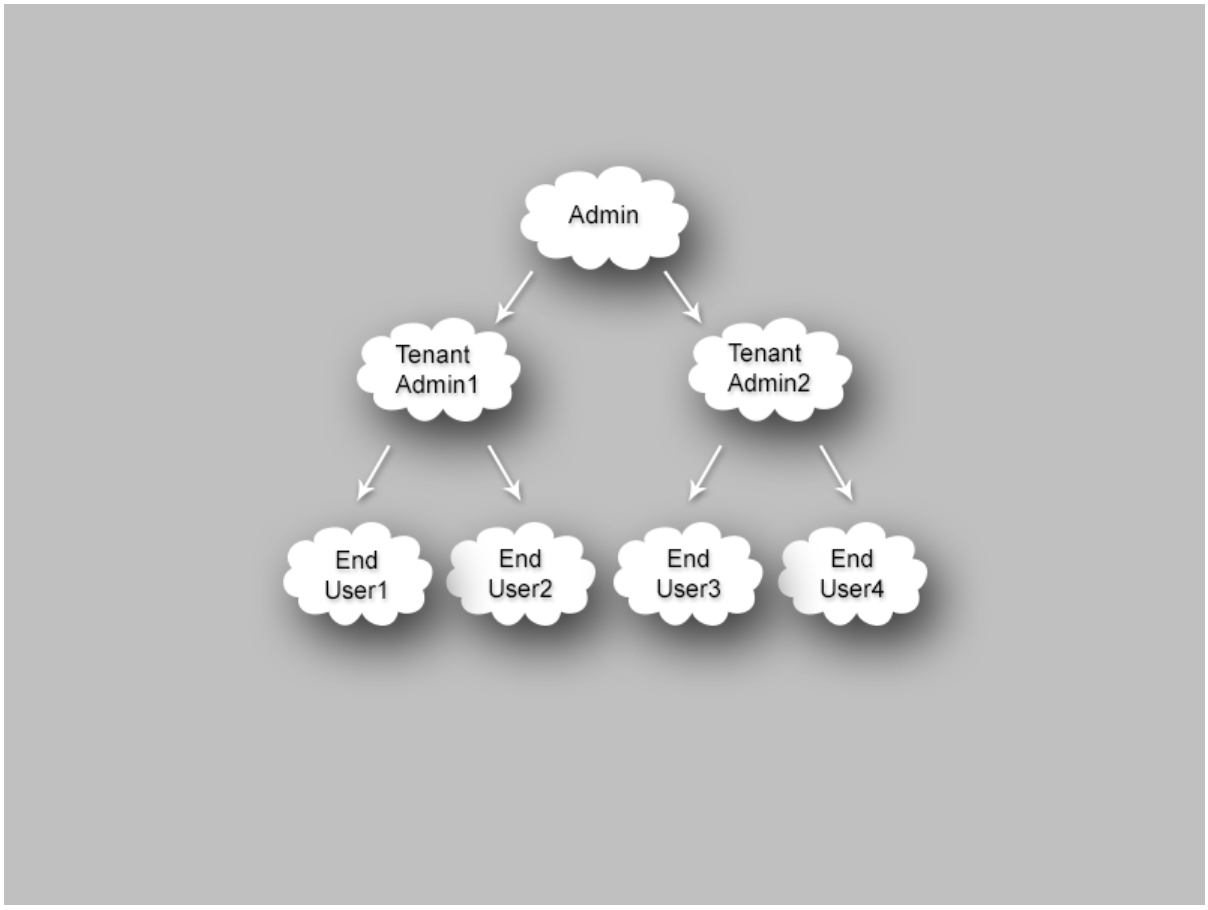
Сохраните все, нажав кнопку Готово в правом верхнем углу окна.

5. Просмотр результата

Наконец, запустите виджет. Он должен быть похож на тот, что представлен на рисунке ниже.

18.26 Многопользовательский контроль доступа

Каждую большую систему необходимо администрировать и контролировать на должном уровне. Существует множество решений и моделей, показывающих, как это сделать разными способами. Сейчас мы обратим внимание на модель, которая помогает объединить пользователей с разным уровнем доступа в "дерево". Корнем этого "дерева" является пользователь Администратор, у которого есть разрешение на выполнение любого действия. На других уровнях находятся такие пользователи, как администратор решения и конечный пользователь с меньшими правами доступа, так называемые "ветви дерева". Самая значимая возможность этой схемы в том, что вы можете добавлять неограниченное количество уровней администрирования и создавать "дерево" настолько сложным, насколько понадобится. В этом практическом уроке мы будем создавать модель многопользовательского контроля доступа со структурой, изображенной ниже:

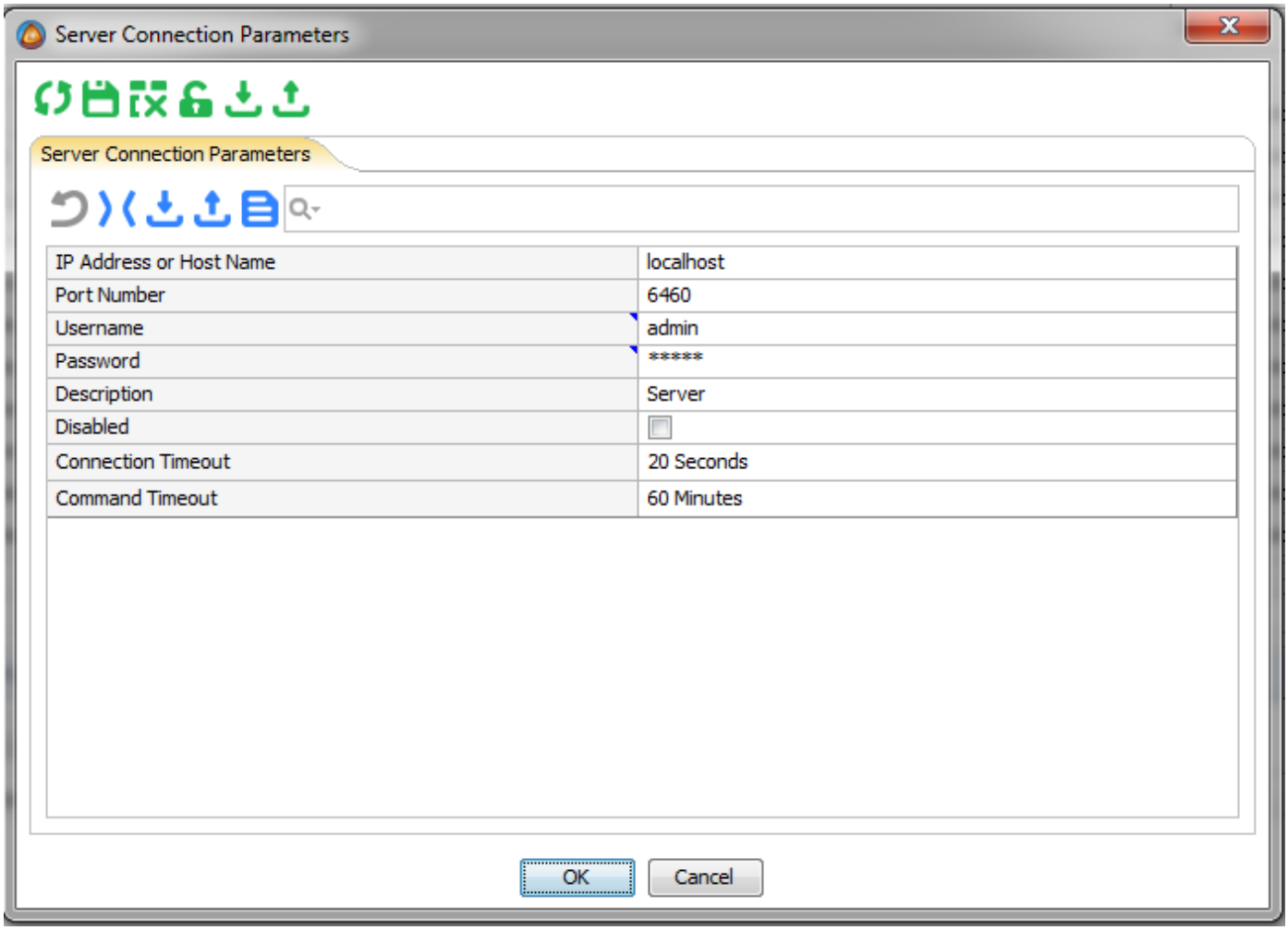


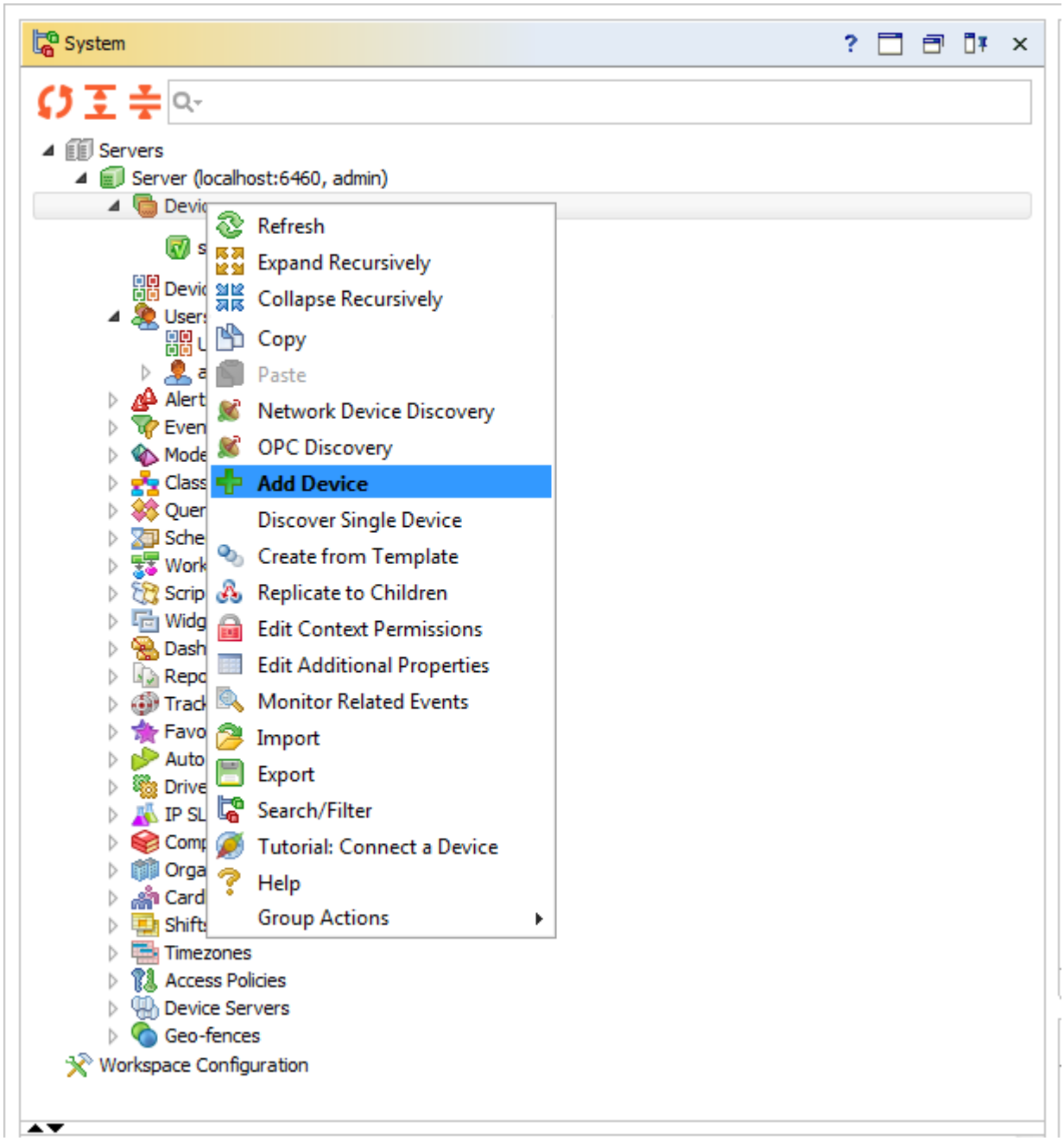
Эта схема показывает отношения между тремя типами пользователей: Администратор, у которого есть полный доступ к пользователям с более низким уровнем доступа, администратор решения, у которого есть тот же доступ, применимый к конечным пользователям. И конечные пользователи, которые администрируются пользователями с более высокими правами доступа. Каждая "ветвь" изолирована друг от друга (Tenant Admin1 может управлять только своими конечными пользователями и "не знает" об **Администраторе** и других администраторах решений. Конечные пользователи могут видеть только свои ресурсы).

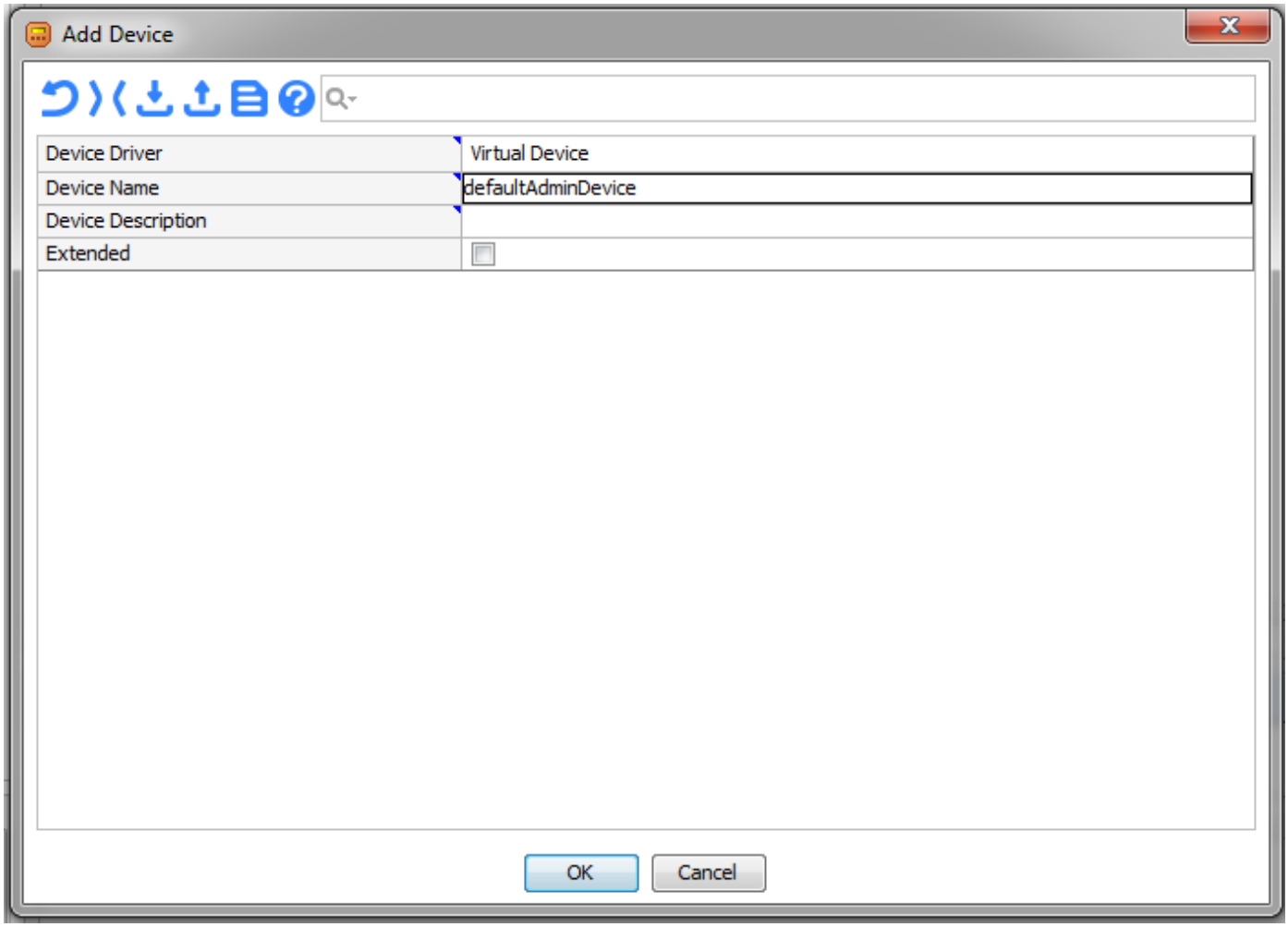
Теперь нужно проверить, как это реализовать на платформе ТВЭЛ AtomMind.

1. Создание тестового ресурса

Прежде всего вам необходимо подключиться к AtomMind Server под учетной записью **администратора** и создать любой тестовый ресурс, например `Virtual Device`, и назвать его `defaultAdminDevice`.

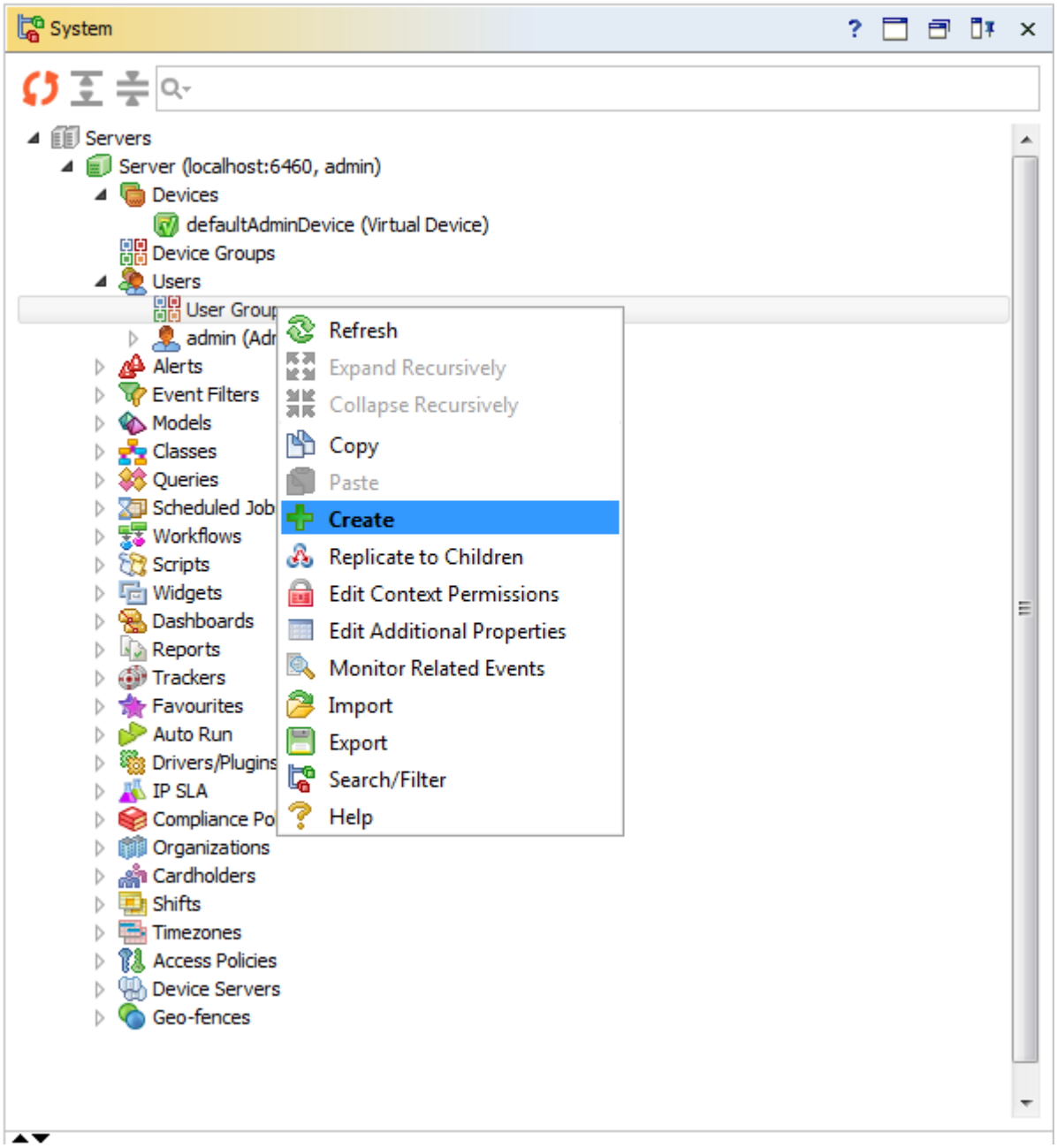


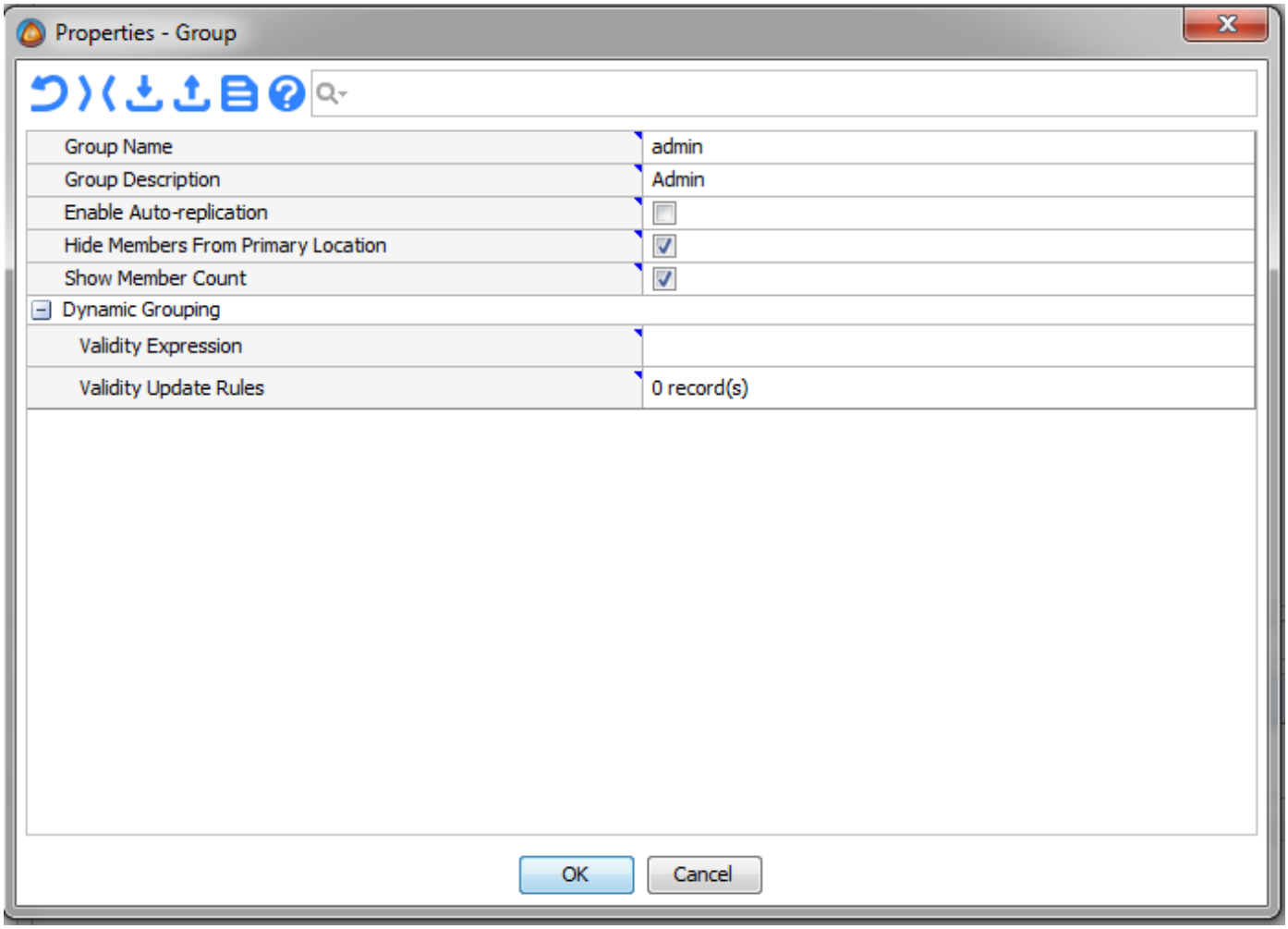




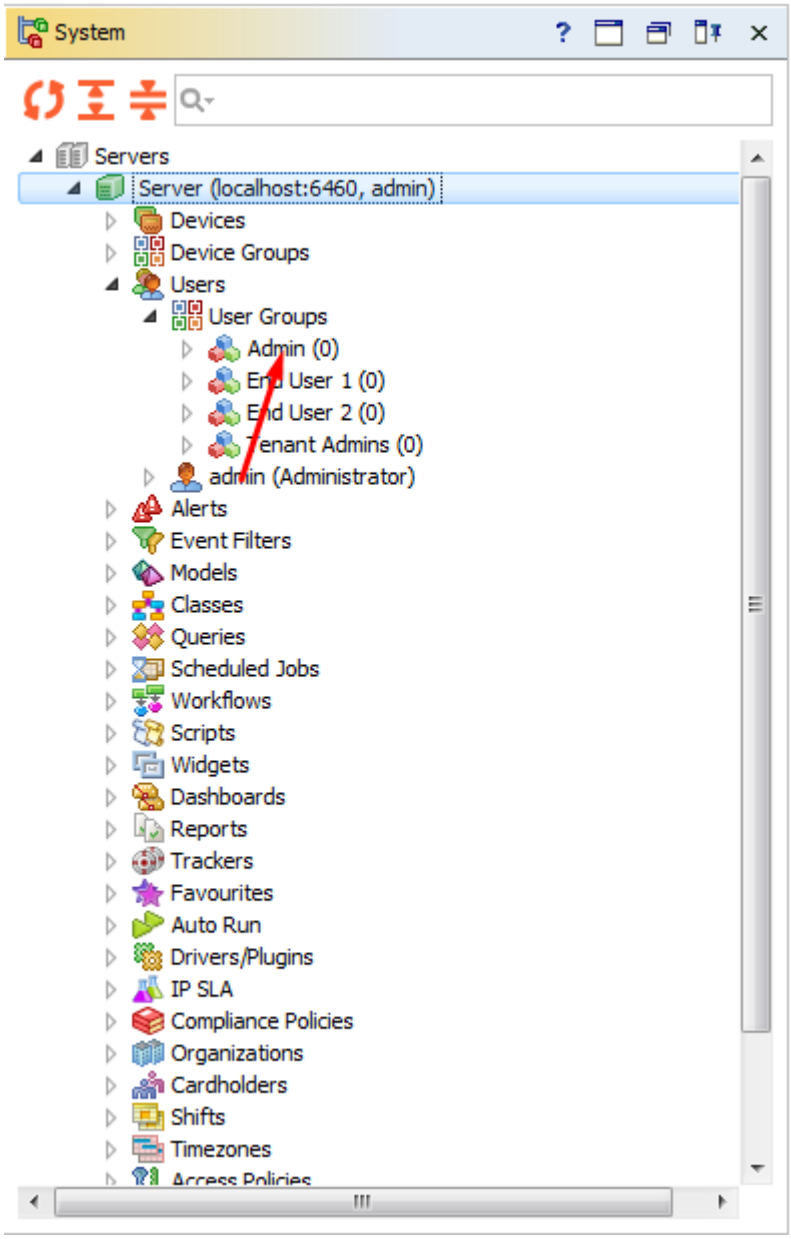
2. Создание групп пользователей

Создайте четыре группы с именами: `Admin, Tenant Admins, End Users 1, End Users 2`, используя опцию **Скрывать членов группы в их основном местоположении**, как показано ниже:



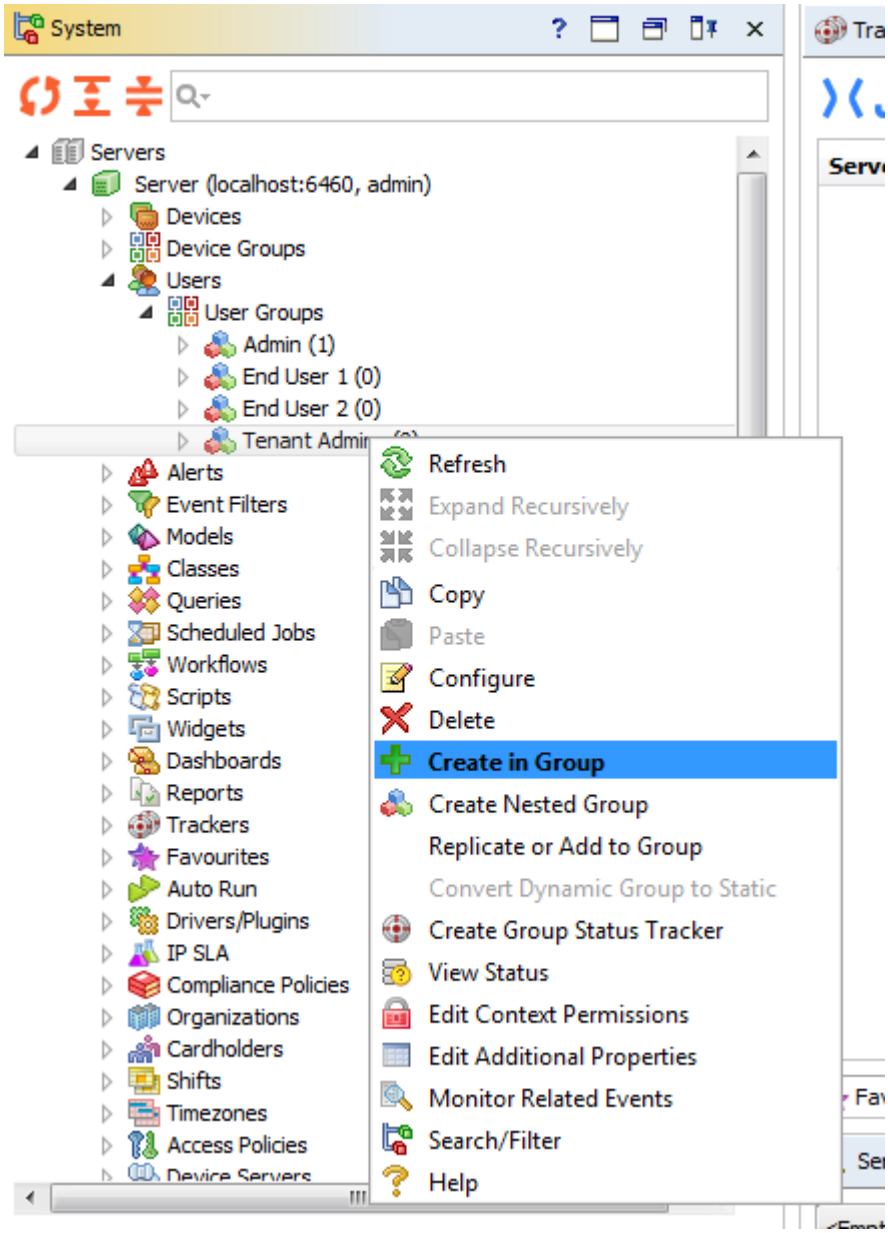


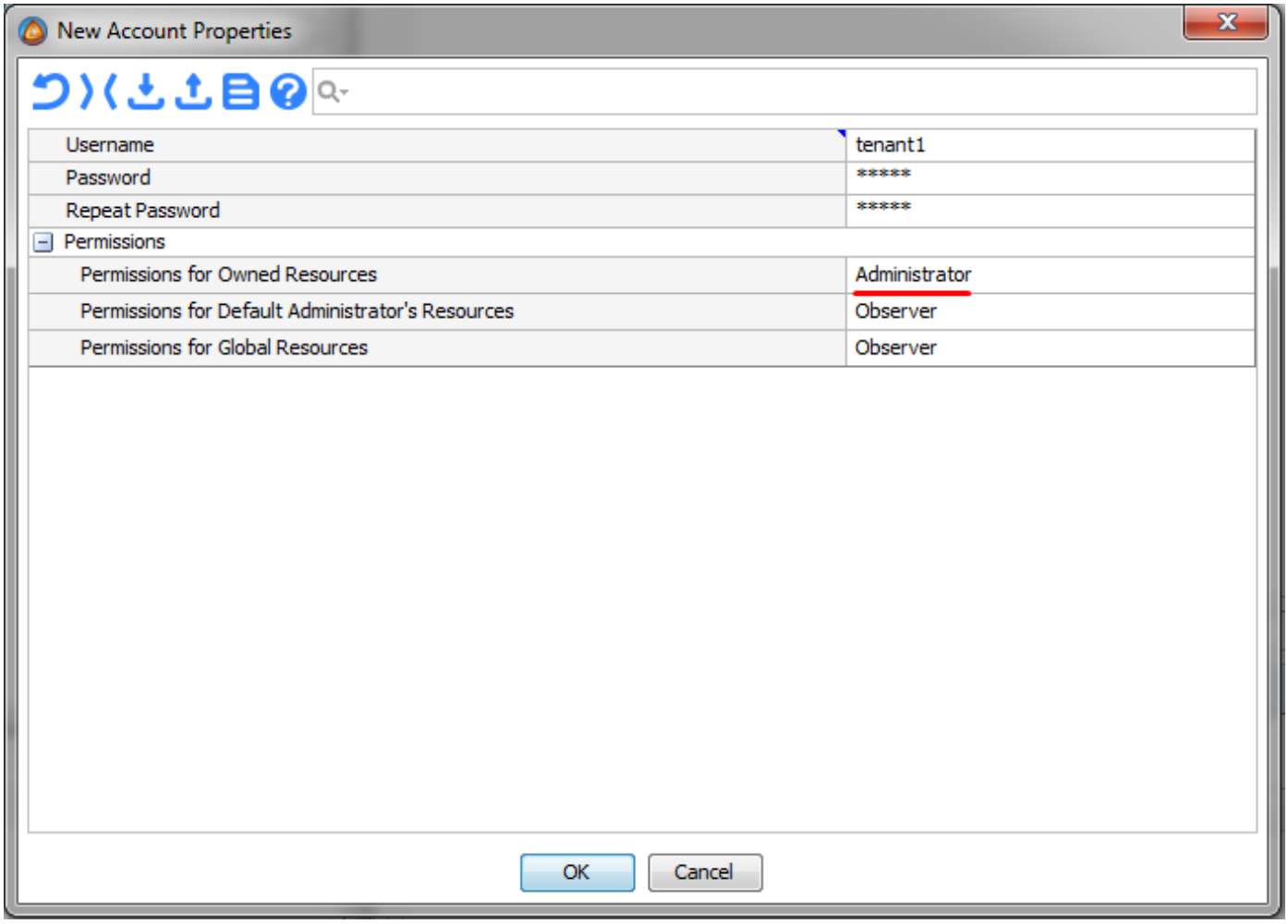
Перетащите пользователя **Администратор** в соответствующую группу.



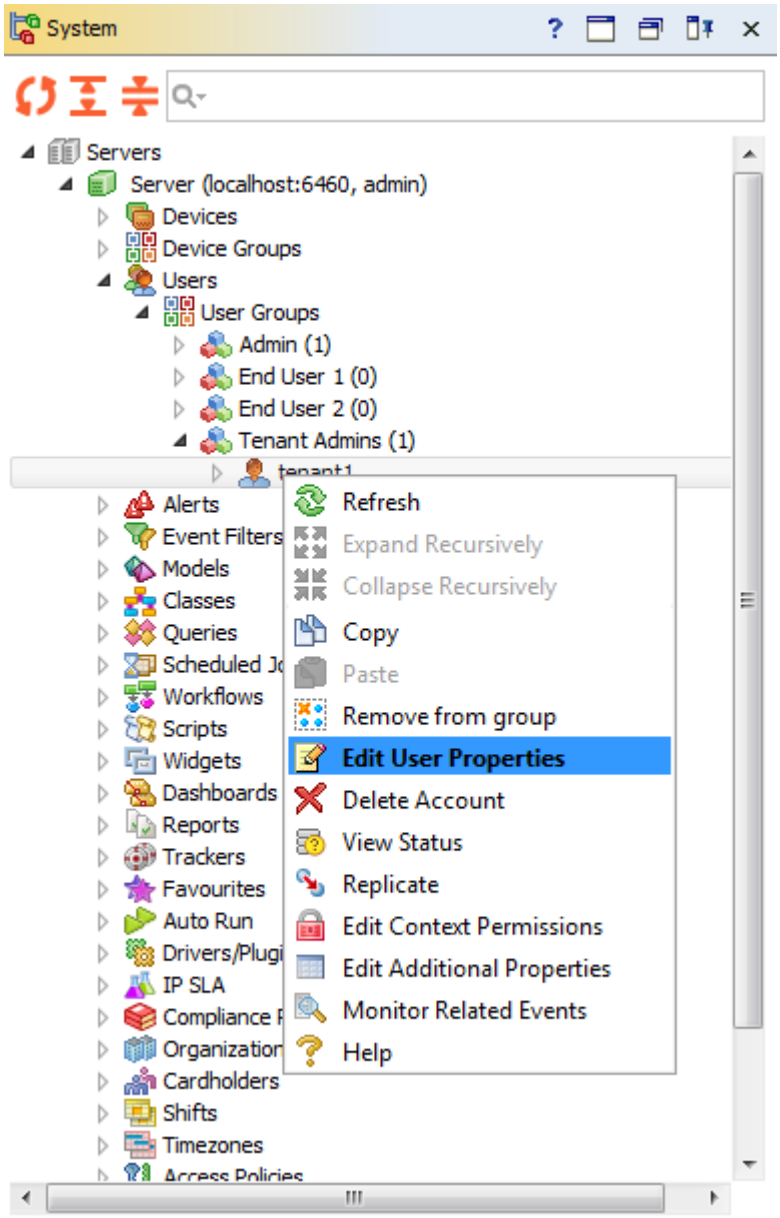
3. Создание пользователя администратор решения и его доступ

Теперь вам необходимо создать администратора решения с именем `tenant1` в группе `Tenant Admins` и установить параметр **Права доступа к собственным ресурсам** на уровне **Администратор**.

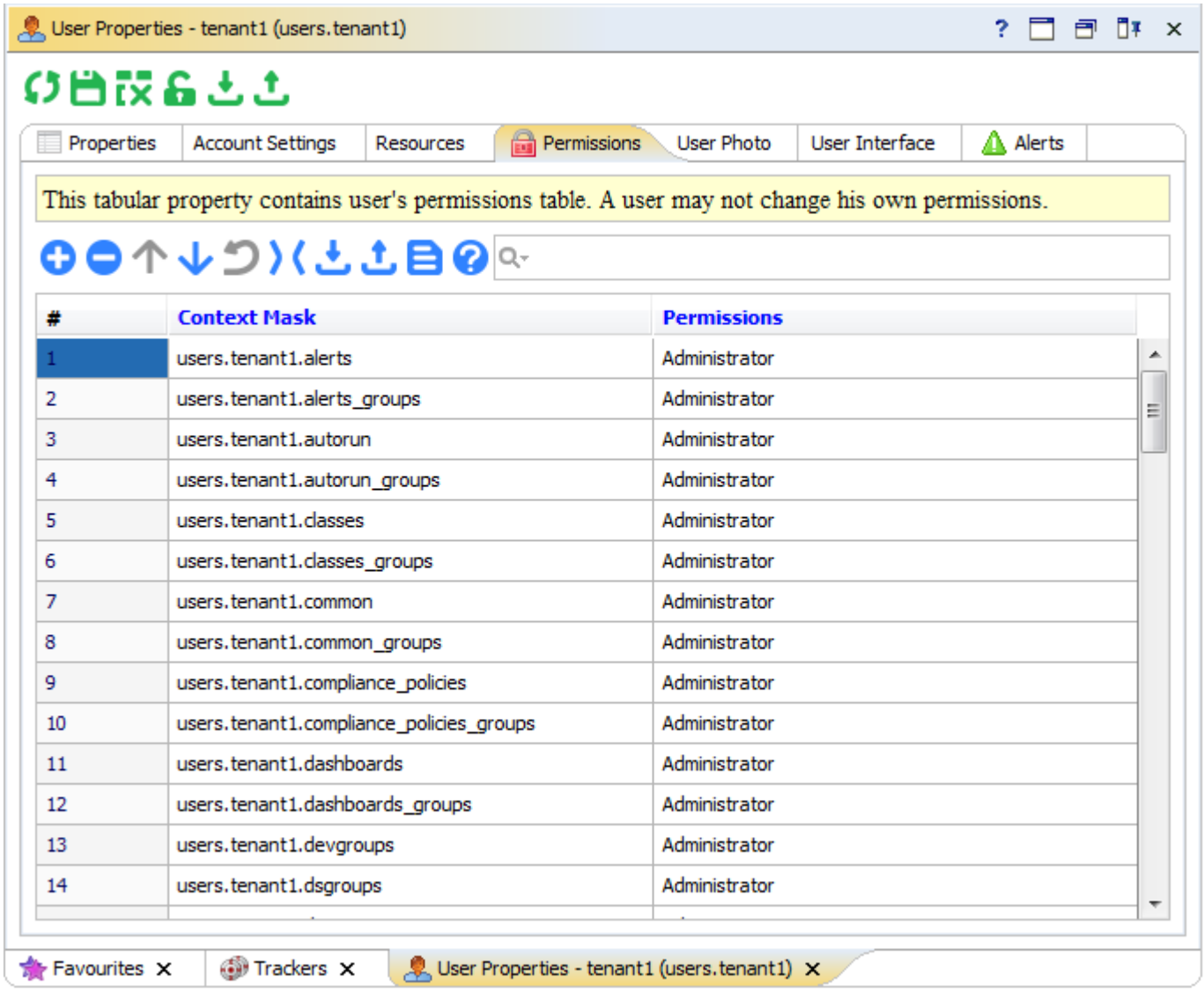




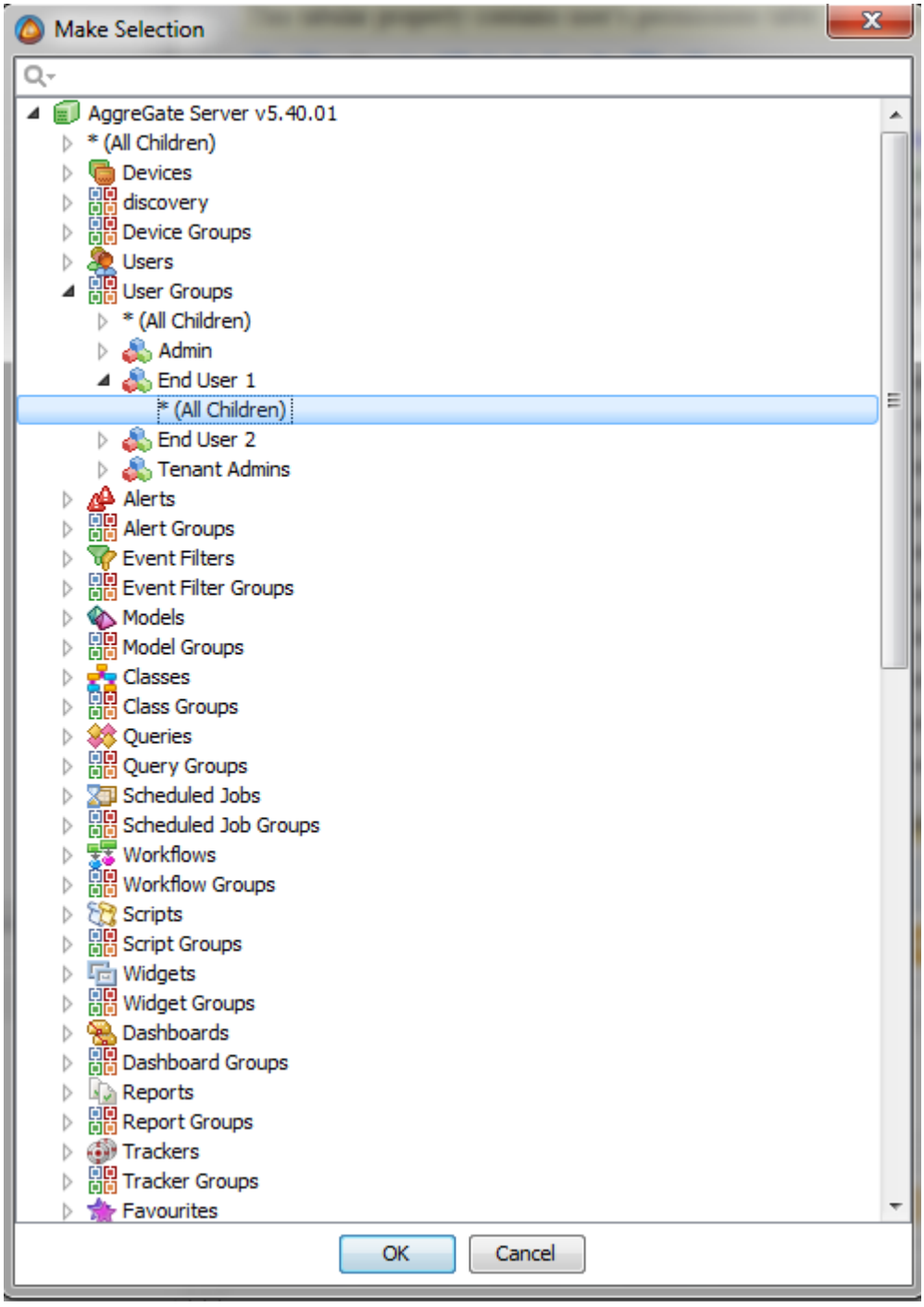
После этого вы можете открыть **Настройки аккаунта**, закладку **Права доступа**.



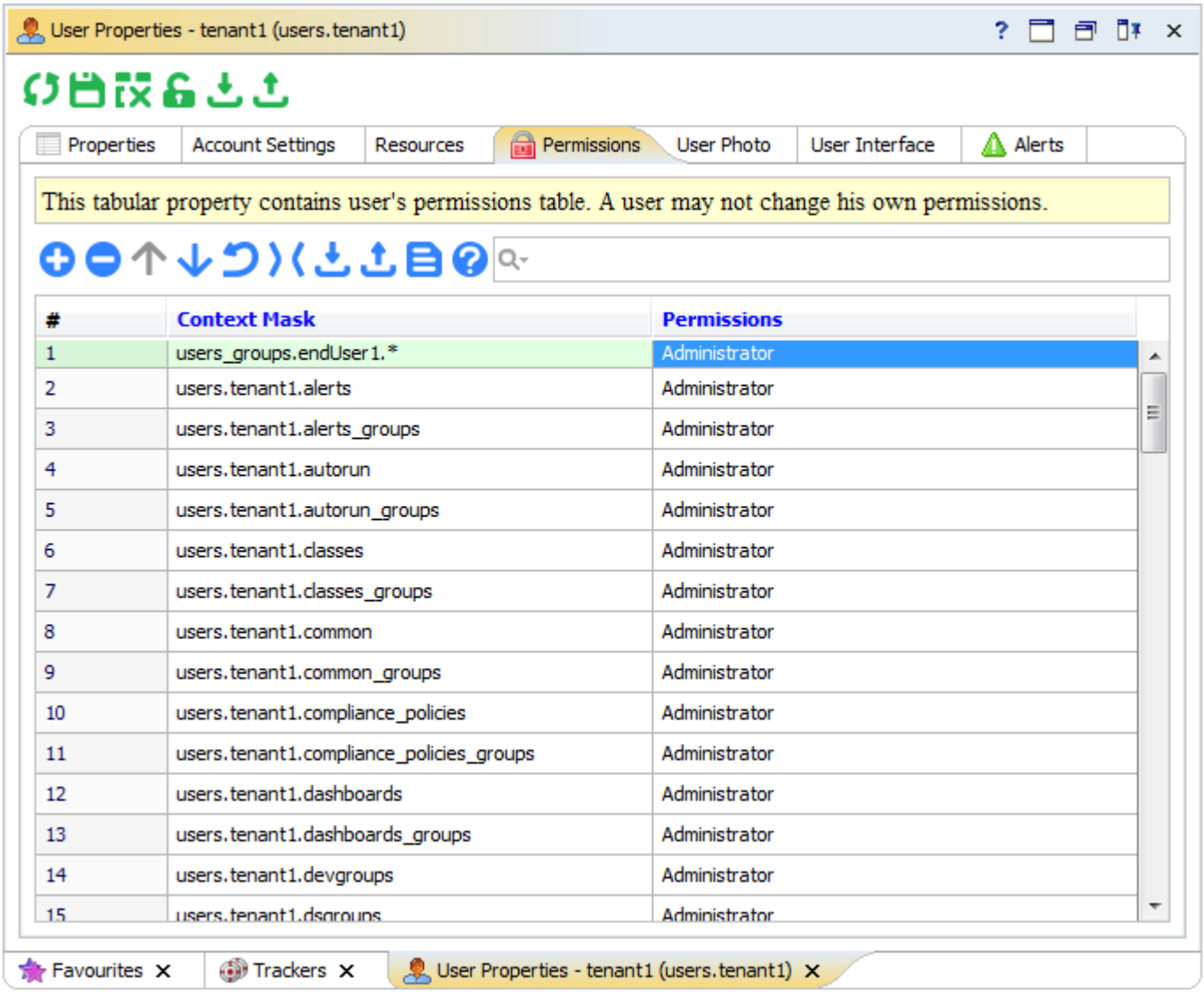
Выберите первый ряд и создайте новую запись `users_group.endUser1.*` в самом начале списка нажатием кнопки **Добавить запись** ( ).



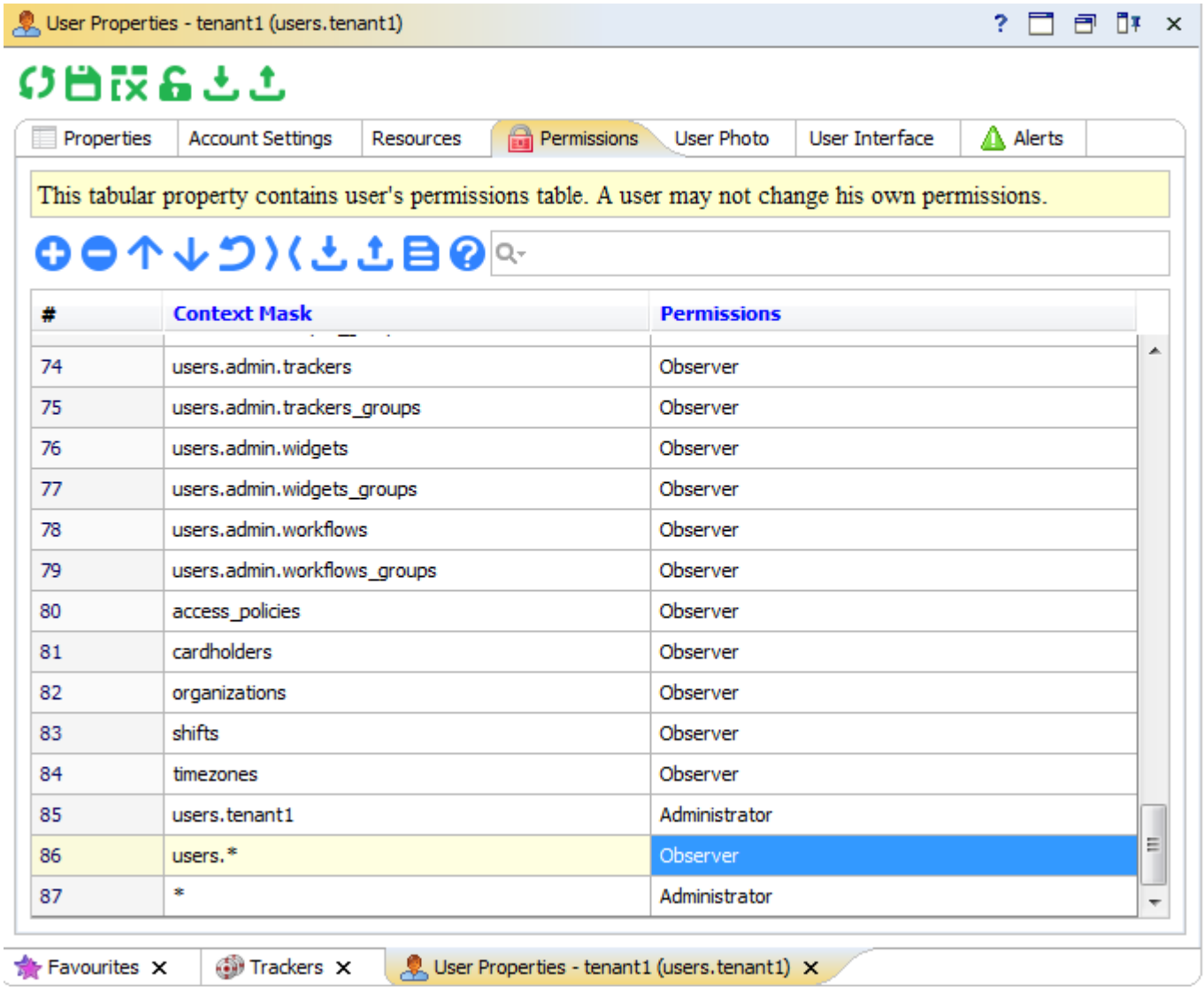
#	Context Mask	Permissions
1	users.tenant1.alerts	Administrator
2	users.tenant1.alerts_groups	Administrator
3	users.tenant1.autorun	Administrator
4	users.tenant1.autorun_groups	Administrator
5	users.tenant1.classes	Administrator
6	users.tenant1.classes_groups	Administrator
7	users.tenant1.common	Administrator
8	users.tenant1.common_groups	Administrator
9	users.tenant1.compliance_policies	Administrator
10	users.tenant1.compliance_policies_groups	Administrator
11	users.tenant1.dashboards	Administrator
12	users.tenant1.dashboards_groups	Administrator
13	users.tenant1.devgroups	Administrator
14	users.tenant1.dsgroups	Administrator



Установите права доступа на уровень **Администратор**.

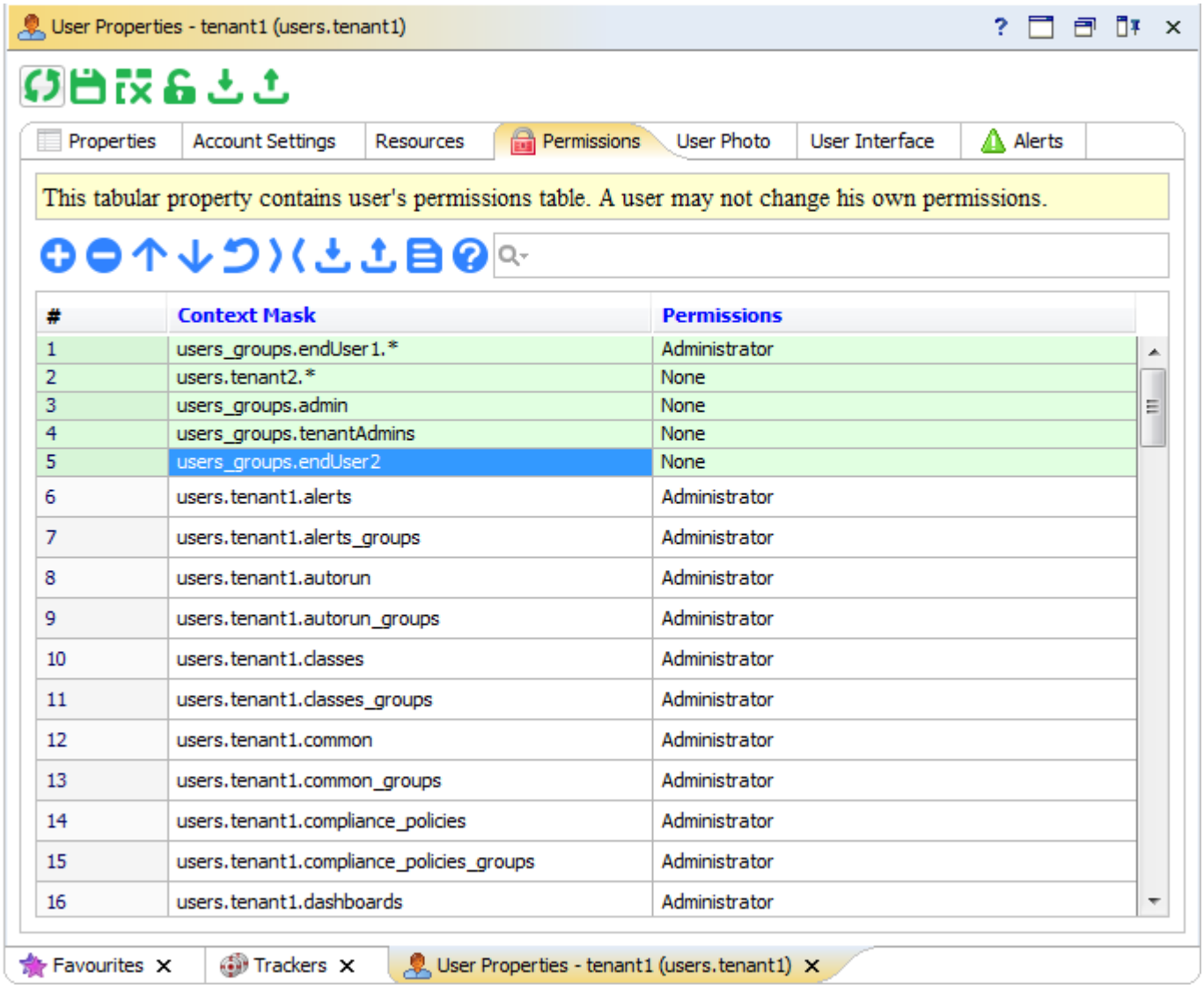


Установите права доступа для записи `users.*` на уровень **Наблюдатель**.



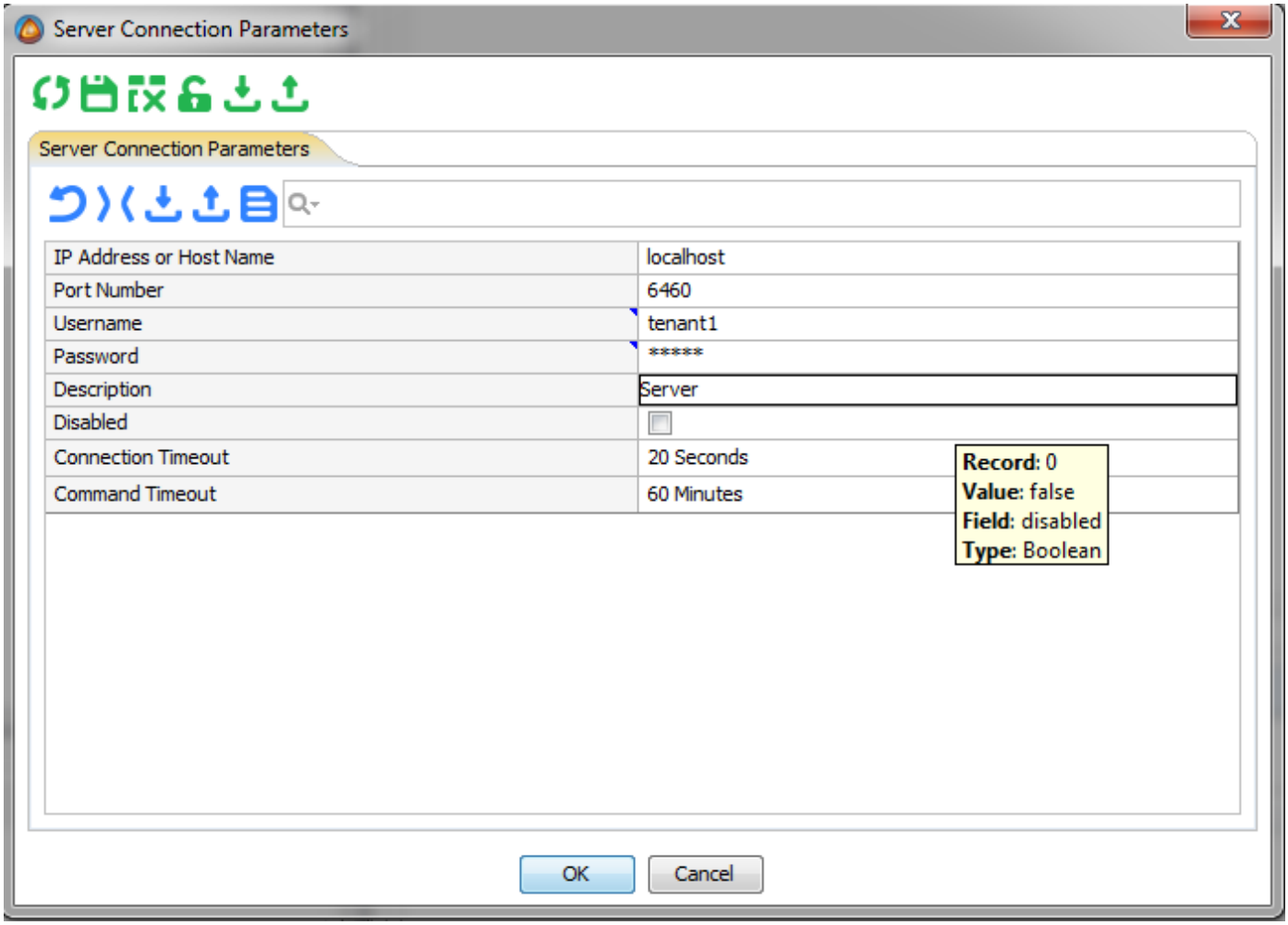
Эти шаги помогают настроить права доступа пользователя `tenant1` для администрирования группы `endUser1` и получить возможность видеть ресурсы пользователя **Администратор**.

Перед тем, как перейти к следующему шагу, вам необходимо запретить пользователю `tenant1` видеть пользователя **Администратор** и других администраторов решений с их группами конечных пользователей. Чтобы это сделать, добавьте четыре записи: `users.tenant2.*`(эта запись необходима, потому что в последующих шагах мы будем создавать второго администратора решений с таким именем), `users_groups.admin`, `users_groups.tenantAdmins`, `users_groups.endUser2`, устанавливая права доступа на уровне **Нет прав**. Не забудьте сохранить ваши настройки, нажав кнопку **Сохранить свойства** ( ).

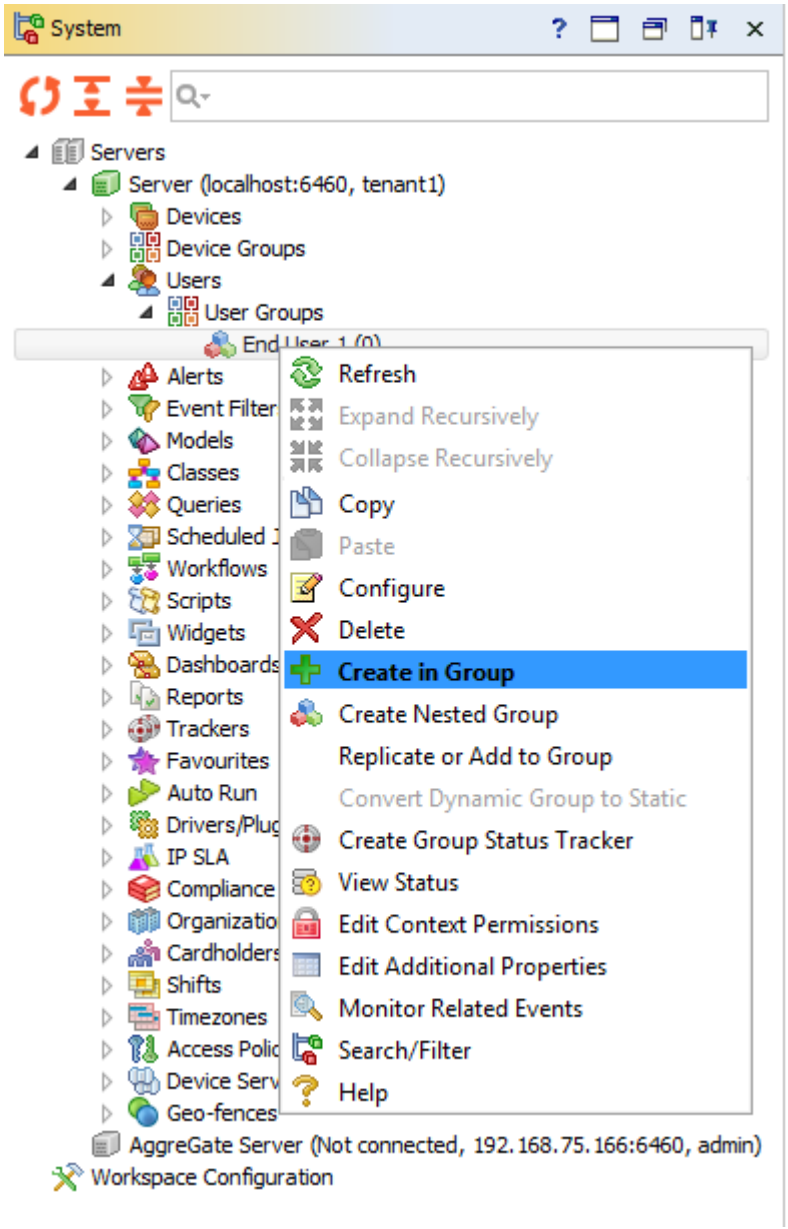


4. Создание тестого устройства на уровне конечного пользователя и администратора решения

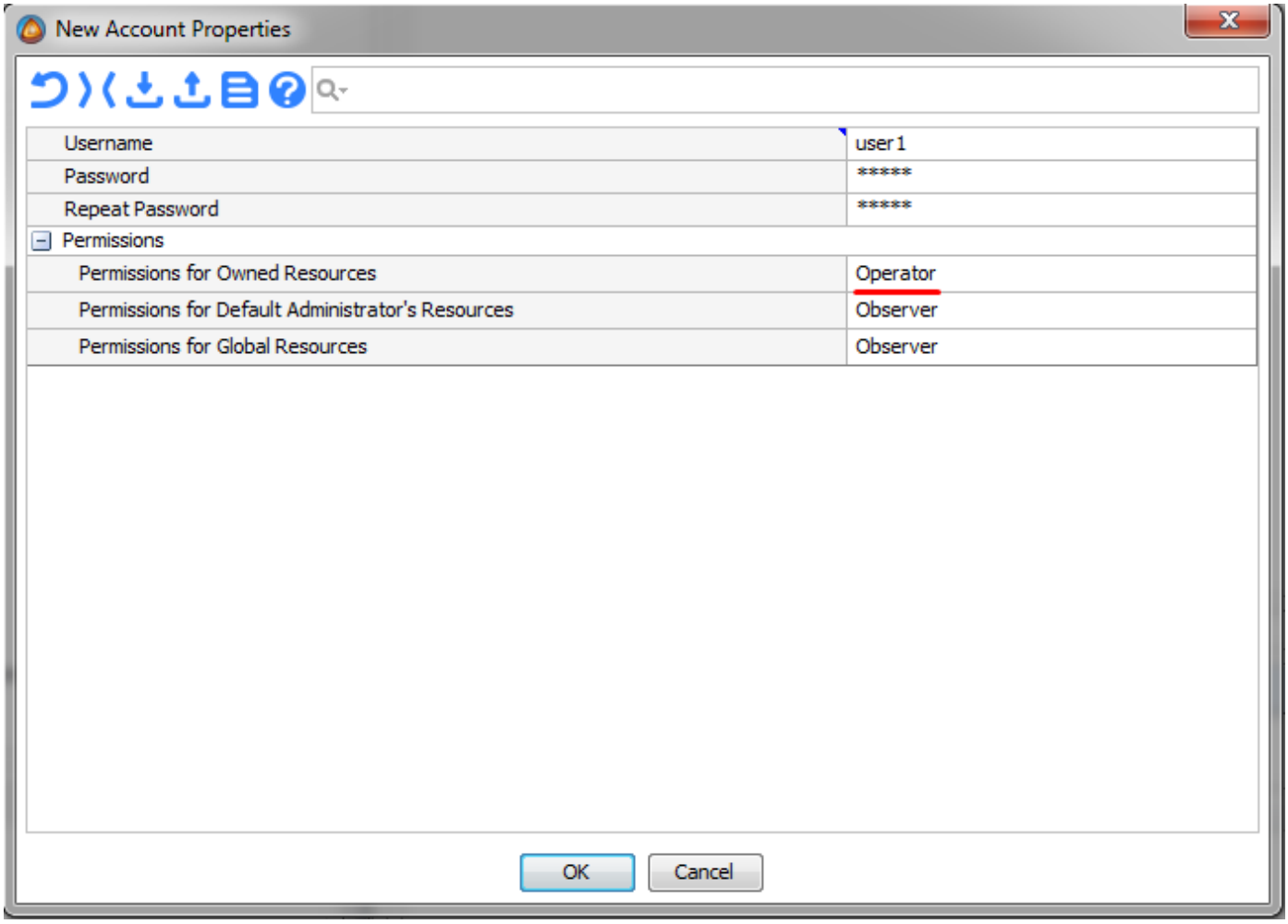
Все готово к созданию конечного пользователя. Зайдите на сервер под учетной записью `tenant1` и создайте тестовое устройство, как описано в Шаге №1. Назовите его `defaultTenant1Device`.



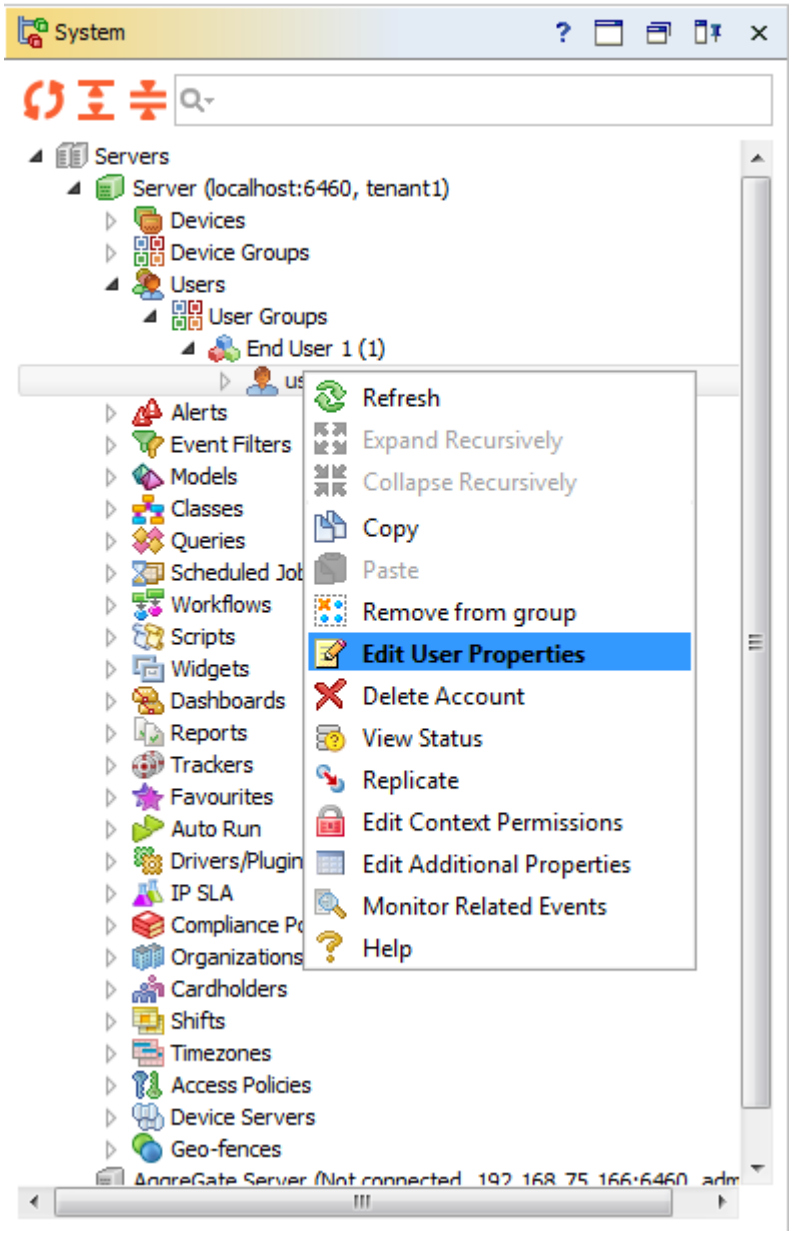
Создайте пользователя `user1` в группе `End User 1`.



Установите уровень **Оператор** в поле **Права доступа к собственным реурсам**.

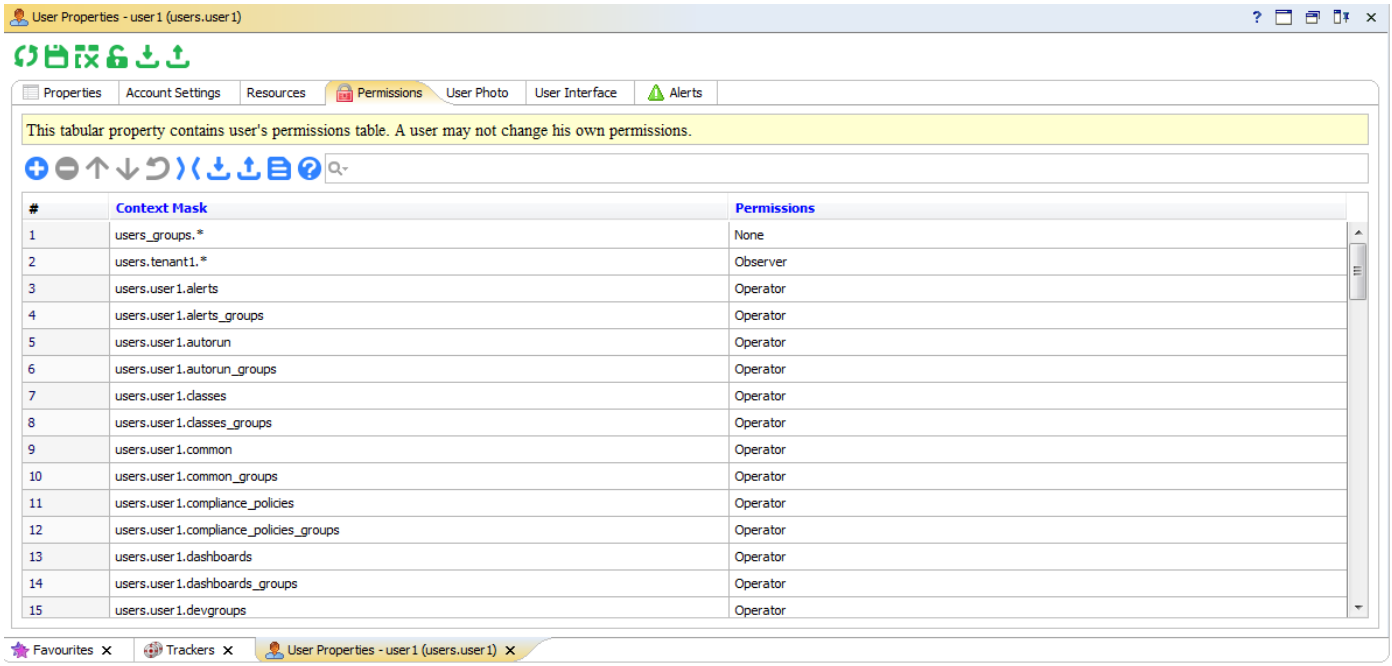


Выберите пункт **Редактировать настройки аккаунта** в контекстном меню пользователя `user1`, откройте вкладку **Права доступа**.



Добавьте первую запись с маской контекстов `users_groups.*` и правами доступа **Нет прав**, чтобы запретить `user1` видеть информацию про всех остальных пользователей.

Добавьте далее запись с маской контекстов `users.tenant1.*` и правами доступа **Наблюдатель**, чтобы пользователь `user1` мог видеть все родительские ресурсы.



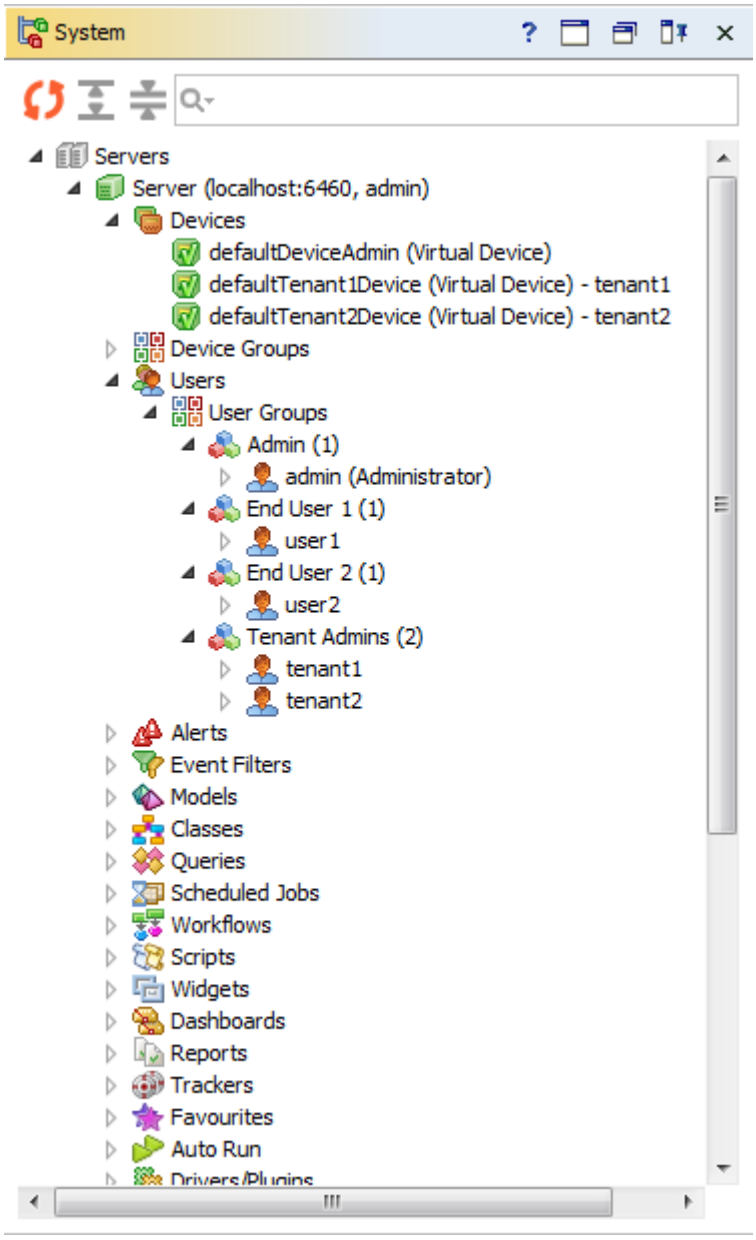
#	Context Mask	Permissions
1	users_groups.*	None
2	users.tenant1.*	Observer
3	users.user1.alerts	Operator
4	users.user1.alerts_groups	Operator
5	users.user1.autorun	Operator
6	users.user1.autorun_groups	Operator
7	users.user1.classes	Operator
8	users.user1.classes_groups	Operator
9	users.user1.common	Operator
10	users.user1.common_groups	Operator
11	users.user1.compliance_policies	Operator
12	users.user1.compliance_policies_groups	Operator
13	users.user1.dashboards	Operator
14	users.user1.dashboards_groups	Operator
15	users.user1.devgroups	Operator

5. Последний шаг настройки и результат

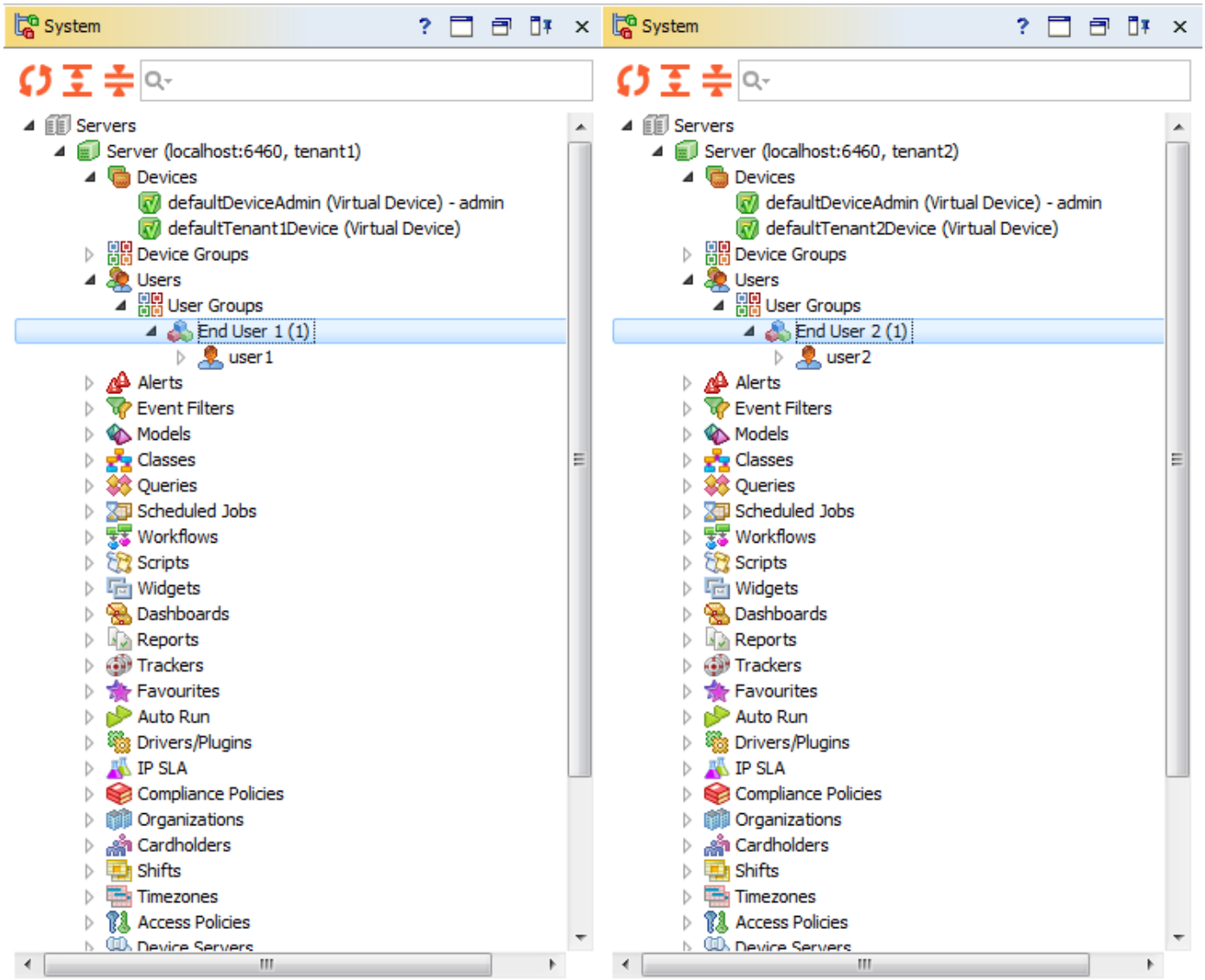
Теперь, когда вы знаете, как создать администраторов решений и конечных пользователей, вы можете шаг за шагом добавлять вторую ветвь дерева для второго администратора решений (назовите его `tenant2`) и его конечного пользователя (назовите его `user2`). Будьте внимательны при настройке доступа для всех новых пользователей. Используйте корректные имена контекстов, когда вы захотите разрешить или запретить доступ. Соблюдайте указанный выше порядок правил на вкладке **Права доступа**.

Если вы выполните все действия без ошибок, вы получите результат, показанный ниже.

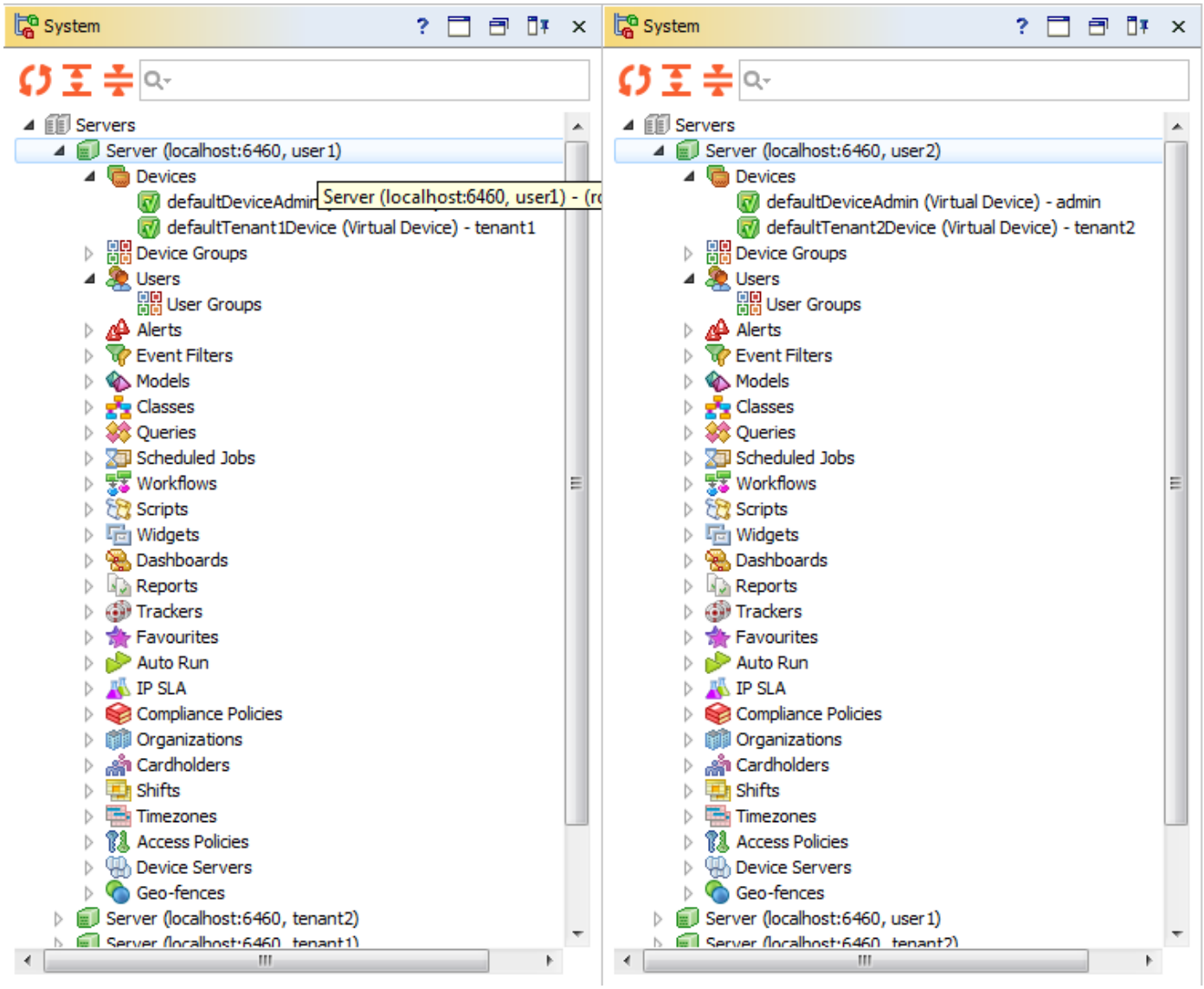
На уровне доступа пользователя **Администратор** вы увидите все ресурсы остальных пользователей, также будете иметь полный доступ к ним.



На уровне доступа администратора решений вы увидите только ресурсы текущего администратора решений, а также будете иметь полный доступ к его конечному пользователю. Ресурсы пользователя **Администратор** доступны только для чтения.



Конечному пользователю доступны для чтения все ресурсы пользователей более высокого уровня его ветви.



18.27 Работа с обучаемым модулем

Обучаемые модули 1599 являются потомками контекста машинного обучения 865. Цель обучаемых модулей - решение конкретных задач машинного обучения.

Урок объясняет, как создать, сконфигурировать и использовать обучаемый модуль на примере задачи регрессии. В качестве иллюстрации мы будем использовать популярный набор данных *Boston House Prices* (Harrison, D. and Rubinfeld, D.L. 'Hedonic prices and the demand for clean air', J. Environ. Economics & Management, vol.5, 81-102, 1978), который часто применяют как ориентир для задач регрессии.

Набор данных содержит 14 столбцов -13 признаков и целевую переменную:

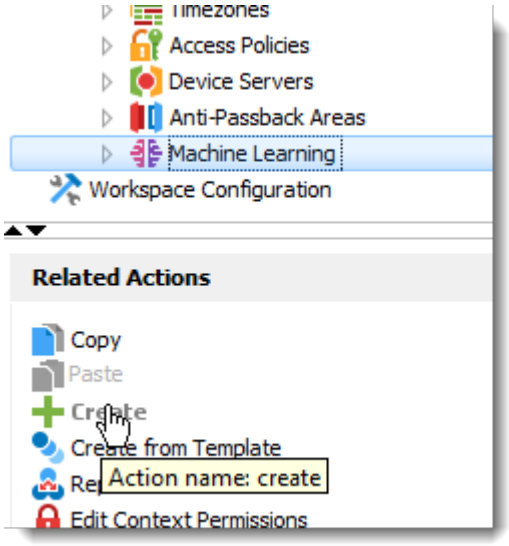
Название столбца	**Описание**
CRIM	Уровень преступности на душу населения
ZN	Процент земли, застроенной жилыми домами
INDUS	Процент деловой застройки
CHAS	1 - если участок граничит с рекой, 0 - в противном случае
NOX	Концентрация оксида азота (деленная на 10^7)
RM	Среднее число комнат по домам
AGE	Процент домов, построенных до 1940 года и занимаемых владельцами

DIS	Взвешенное расстояние до пяти деловых центров Бостона
RAD	Индекс удаленности от радиальных магистралей
TAX	Величина налога на $10,000
PTRATIO	Соотношение учеников и учителей по городу
B	1000(Bk - 0.63)^2, где Bk - доля афроамериканцев
LSTAT	Процент жителей с низким социальным статусом
MEDV	Целевая переменная: медианное значение цены строения

Задача: построить модель, которая предсказывает целевую переменную (медианное значение цены строения) на основе данного набора признаков.

1. Создание обучаемого модуля

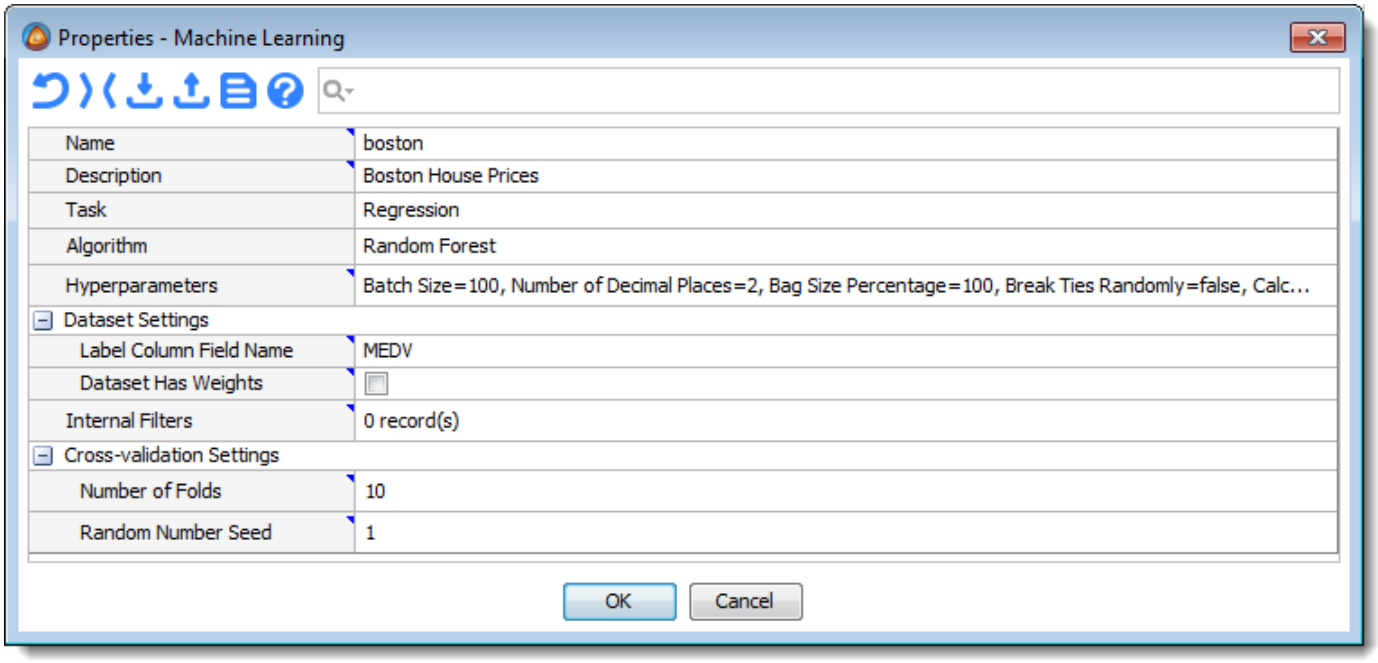
Чтобы создать новый обучаемый модуль найдите узел **Машинное обучение** ( ) в Системном дереве 370.
Нажмите на действие **Создать** ( ) на панели **Действия**:



Можно также дважды щелкнуть мышью на узел **Машинное обучение**, чтобы запустить действие **Создать,** т.к. это действие является действием по умолчанию 88 в контексте Машинное обучение 865.

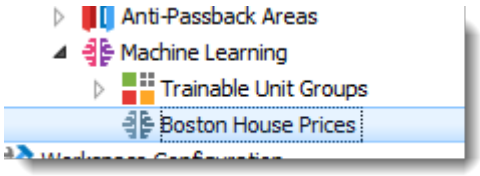
В окне **Свойства обучаемого модуля** 869:

- Установите **Имя** в `boston`
- Установите **Описание** в `Boston House Prices`
- Оставьте типа **Задачи** `Regression`, посколку мы решаем задачу регрессии
- Установите **Алгоритм** в `Random Forest`; это будет наш приоритетный алгоритм для данной бизнес-задачи
- Установите **Имя поля с зависимой переменной** в `MEDV`, так как это имя целевой переменной (зависимая переменная)



При необходимости вы можете изменить гиперпараметры[870] алгоритма, щелкнув на значение свойства **Гиперпараметры**.

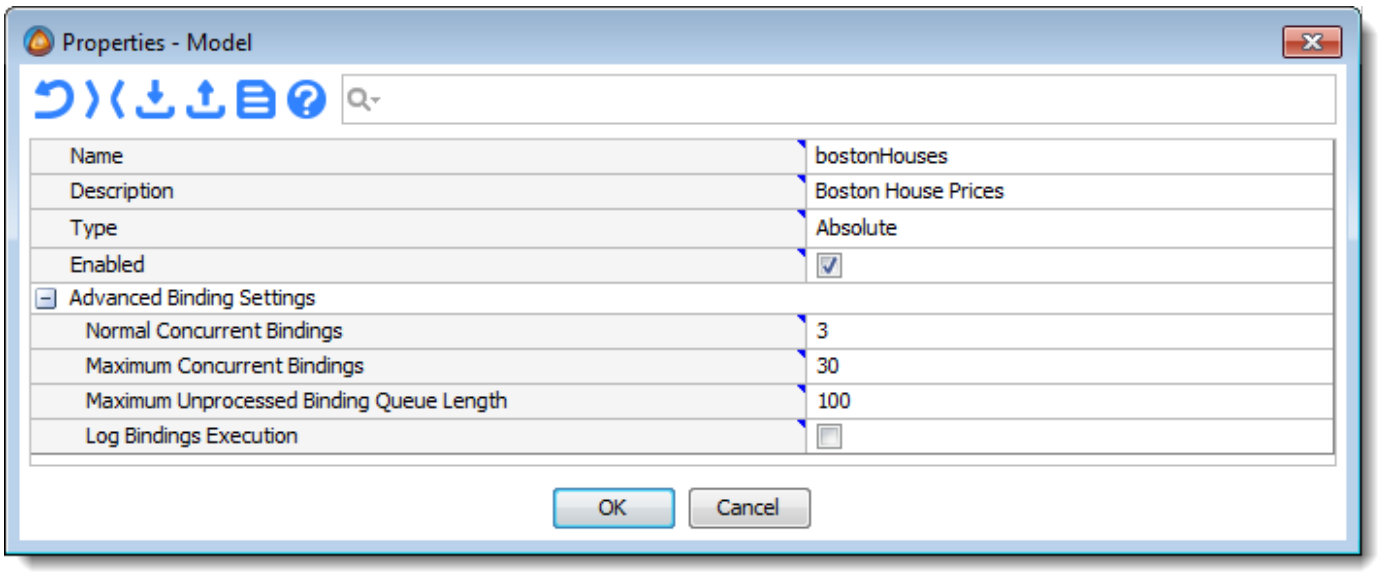
Чтобы продолжить кликните на OK два раза. Обучаемый модуль создан. Он еще не обучен, поэтому его иконка окрашена серо-голубым:



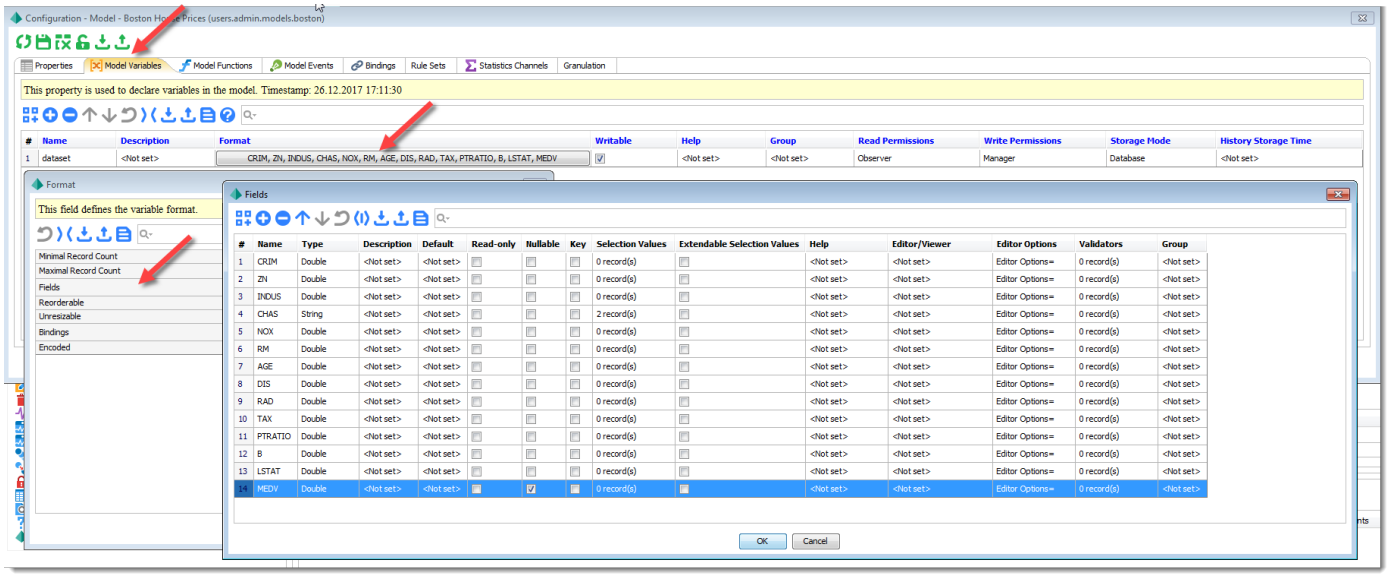
2. Подготовка данных для обучаемого модуля

В данном примере мы создадим модель[810] для хранения данных, необходимых обучаемому модулю. Мы будем также использовать эту модель для вызова функций обучаемого модуля.

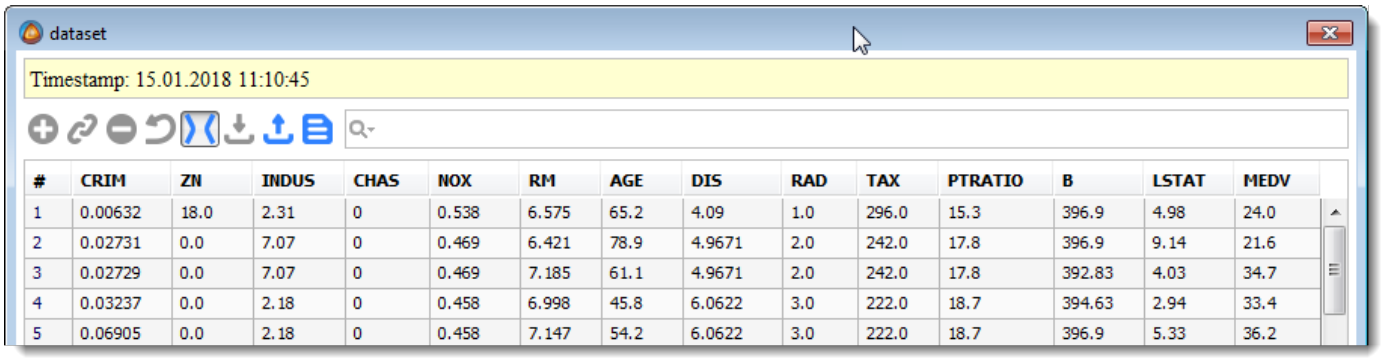
Чтобы создать модель, кликните мышью дважды на узел **Модели** в системном дереве. Сконфигурируйте свойства модели:



А теперь давайте создадим переменную для хранения блока данных. Кликните на иконку добавления ( ) во вкладке Переменные модели. Затем кликните на свойство Формат и добавьте поля, как показано ниже:



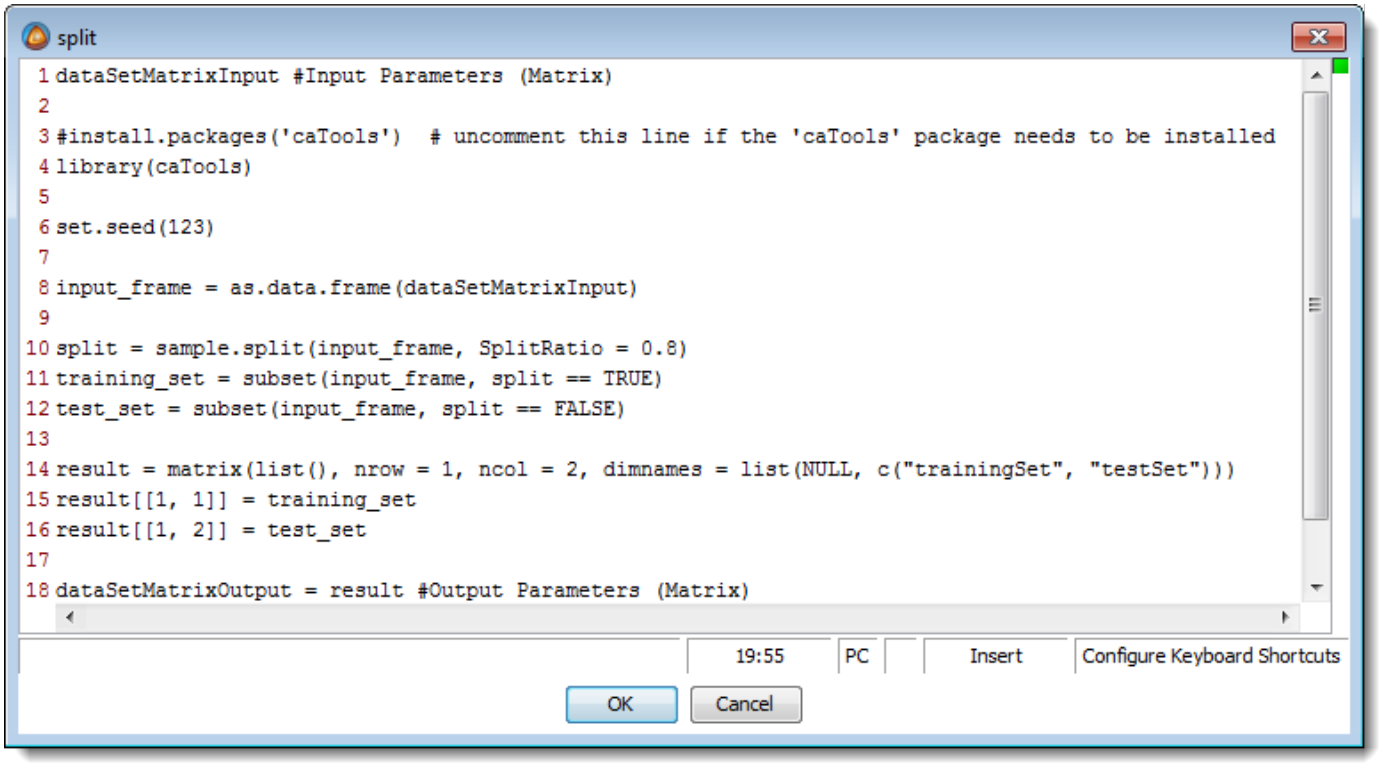
Теперь необходимо импортировать данные из файла. Для этого необходимо создать устройство Локальный файл 576, определить путь к файлу в настройках устройства, прочитать содержание файла, а затем использовать функцию 124 **tableFromCSV,** которая преобразует содержание файла в таблицу данных 49. Функция поддерживает различные разделители, включая знак табуляции. Тем не менее, заголовок файла должен состоять только из одной строки, поэтому все дополнительные строки заголовка нужно удалить из файла. Затем мы сможем использовать привязку 736 в ранее созданной модели, чтобы поместить получившуюся таблицу данных в нашу назначенную переменную. Первые пять экземпляров набора данных выглядят так:



#	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.09	1.0	296.0	15.3	396.9	4.98	24.0
2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.9	9.14	21.6
3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.9	5.33	36.2

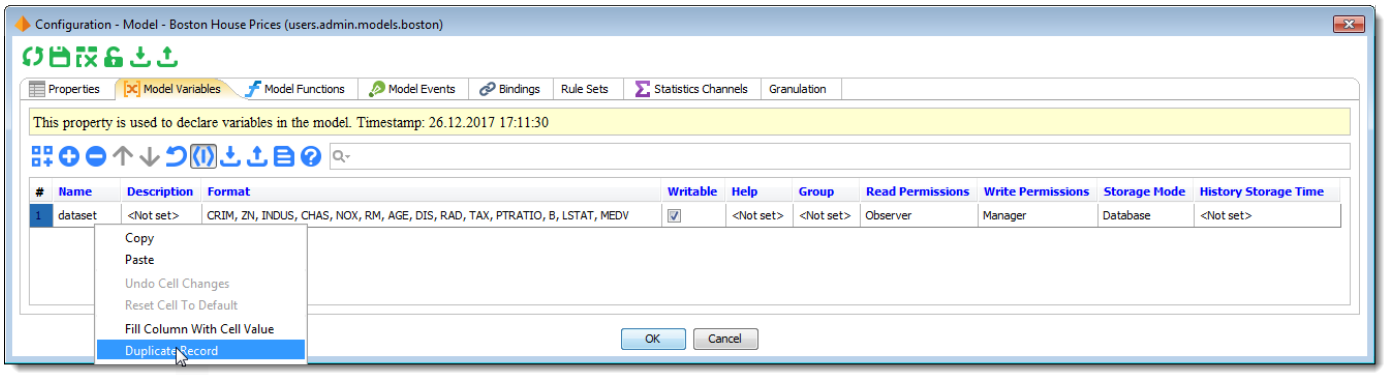
Обратите внимание, что эти действия требовались исключительно для импортирования внешних данных из файла.

Затем мы разобьем набор данных на обучающий набор и тестовый набор. Первый будет использоваться для обучения обучаемого модуля, последний - для оценки качества его работы. Разбивку можно осуществить при помощи скрипта на языке R 882. Давайте создадим скрипт под названием `split`, выберем **Тип языка** `R` и напишем следующий код в поле **Текст скрипта на языке R**:

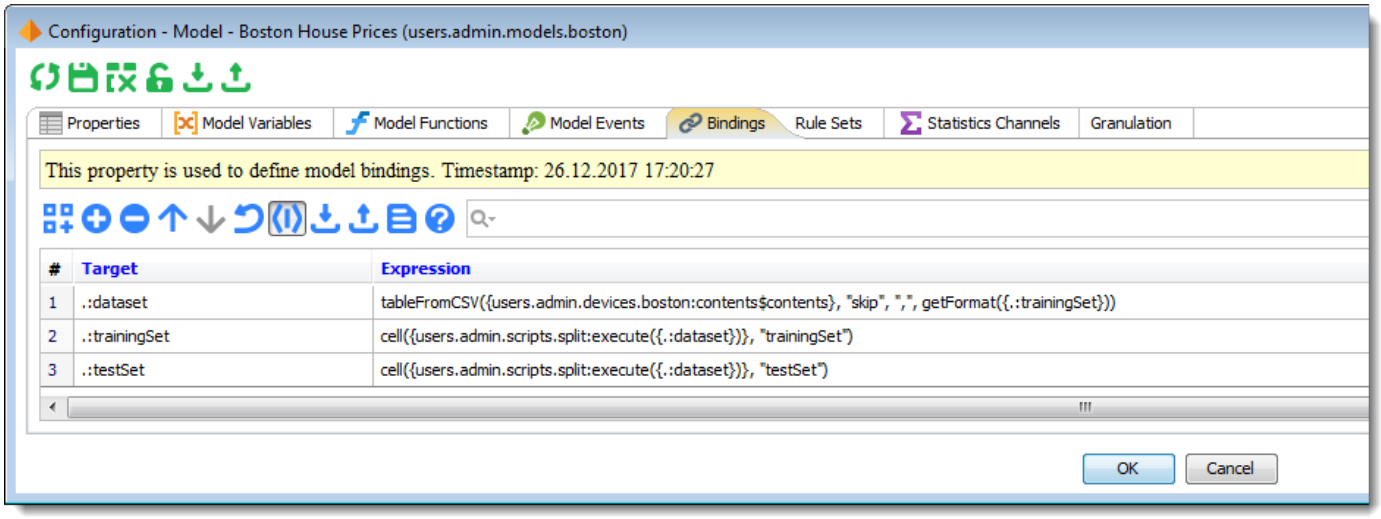


Этот скрипт возвращает таблицу данных, которая состоит из двух вложенных таблиц данных.

Теперь нам надо создать еще две переменные в нашей модели. Поскольку эти переменные имеют тот же формат, как и уже созданная переменная, мы можем просто скопировать ее дважды, а затем дать новым переменным подходящие имена. Чтобы скопировать переменную, кликните правой кнопкой мыши на запись в панели **Переменные модели** и выберите функцию **Копировать запись**:



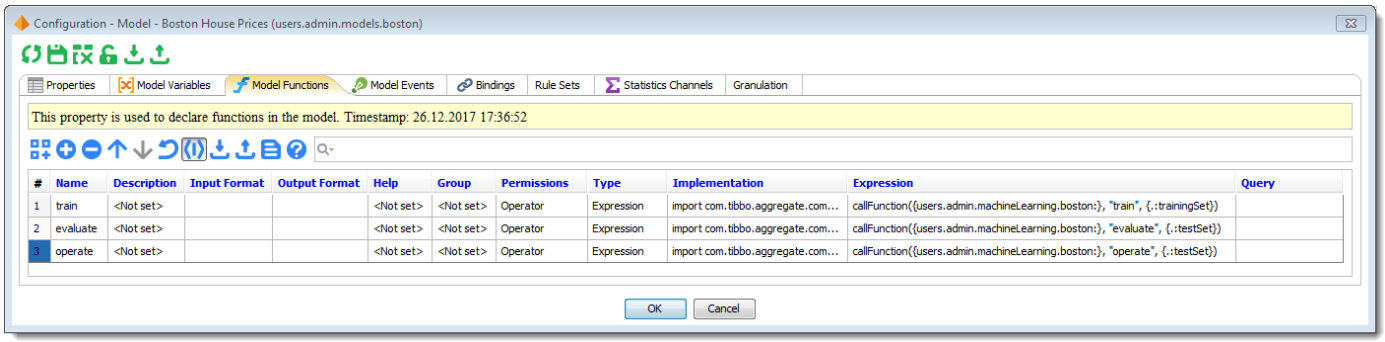
Дайте новой записи имя `trainingSet`. Скопируйте ее еще раз и дайте третьей записи имя `testSet`. Теперь мы можем использовать следующие привязки, чтобы заполнить вновь созданные переменные данными с использованием скрипта на языке R 885, который мы создали ранее:



Здесь можно использовать функцию **cell,** чтобы получить необходимые вложенные таблицы данных.

3. Работа с обучаемым модулем

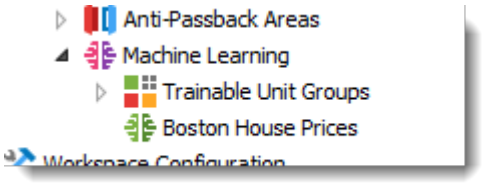
Стадии операции описаны в соответствующем разделе. Мы создадим функции в созданной нами ранее модели для вызова функций контекста нашего обучаемого модуля. Убедитесь, что тип функции установлен в **Выражение**. Затем напишите выражение, как показано ниже:



ОБУЧЕНИЕ

Функция **Обучить** возвращает информацию об обученном модуле в формате String.

После того, как обучаемый модуль обучен, его иконка поменяет цвет:



ОЦЕНКА КАЧЕСТВА

Функция **Оценить** возвращает набор оценочных метрик:

evaluate - Boston House Prices

Correlation Coefficient	0.9355489218731846
Mean Absolute Error	2.358273725271642
Root Mean Squared Error	3.626837056176194
Relative Absolute Error, %	34.34546774949183
Root Relative Squared Error, %	37.47973558895283
Number of Unclassified Instances	0
Unclassified Instances, %	0.0
Total Number of Instances	108

Можно использовать эти метрики для сравнения различных алгоритмов и поиска оптимальных значений гиперпараметров.

ПРЕДСКАЗАНИЯ

В этом примере мы сделаем предсказания экземпляров обучающего набора. Поскольку мы знаем фактические значения целевой переменной, мы можем напрямую сравнить предсказания, сделанные обучаемым модулем, с фактическими значениями:

operate - Boston House Prices

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV (Actual)	MEDV (Predicted)	Error
0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.9	5.33	36.2	32.48358412698411	-3.7164158730158903
0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5.0	311.0	15.2	396.9	19.15	27.1	18.959321933621936	-8.140678066378065
0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5.0	311.0	15.2	392.52	20.45	15.0	19.095688600288604	4.095688600288604
0.80271	0.0	8.14	0	0.538	5.456	36.6	3.7965	4.0	307.0	21.0	288.99	11.69	20.2	18.79144404761905	-1.4085559523809508
0.85204	0.0	8.14	0	0.538	5.965	89.2	4.0123	4.0	307.0	21.0	392.53	13.83	19.6	18.072097344322348	-1.5279026556776536
0.75026	0.0	8.14	0	0.538	5.924	94.1	4.3996	4.0	307.0	21.0	394.33	16.3	15.6	16.547704761904768	0.9477047619047685
1.38799	0.0	8.14	0	0.538	5.95	82.0	3.99	4.0	307.0	21.0	232.6	27.71	13.2	14.991673015873014	1.7916730158730143
0.06417	0.0	5.96	0	0.499	5.933	68.2	3.3603	5.0	279.0	19.2	396.9	9.68	18.9	21.819767063492076	2.9197670634920776
0.17505	0.0	5.96	0	0.499	5.966	30.2	3.8473	5.0	279.0	19.2	393.43	10.13	24.7	21.435308333333328	-3.264691666666671
0.18836	0.0	6.91	0	0.448	5.786	33.3	5.1004	3.0	233.0	17.9	396.9	14.15	20.0	19.960980952380954	-0.03901904761904618

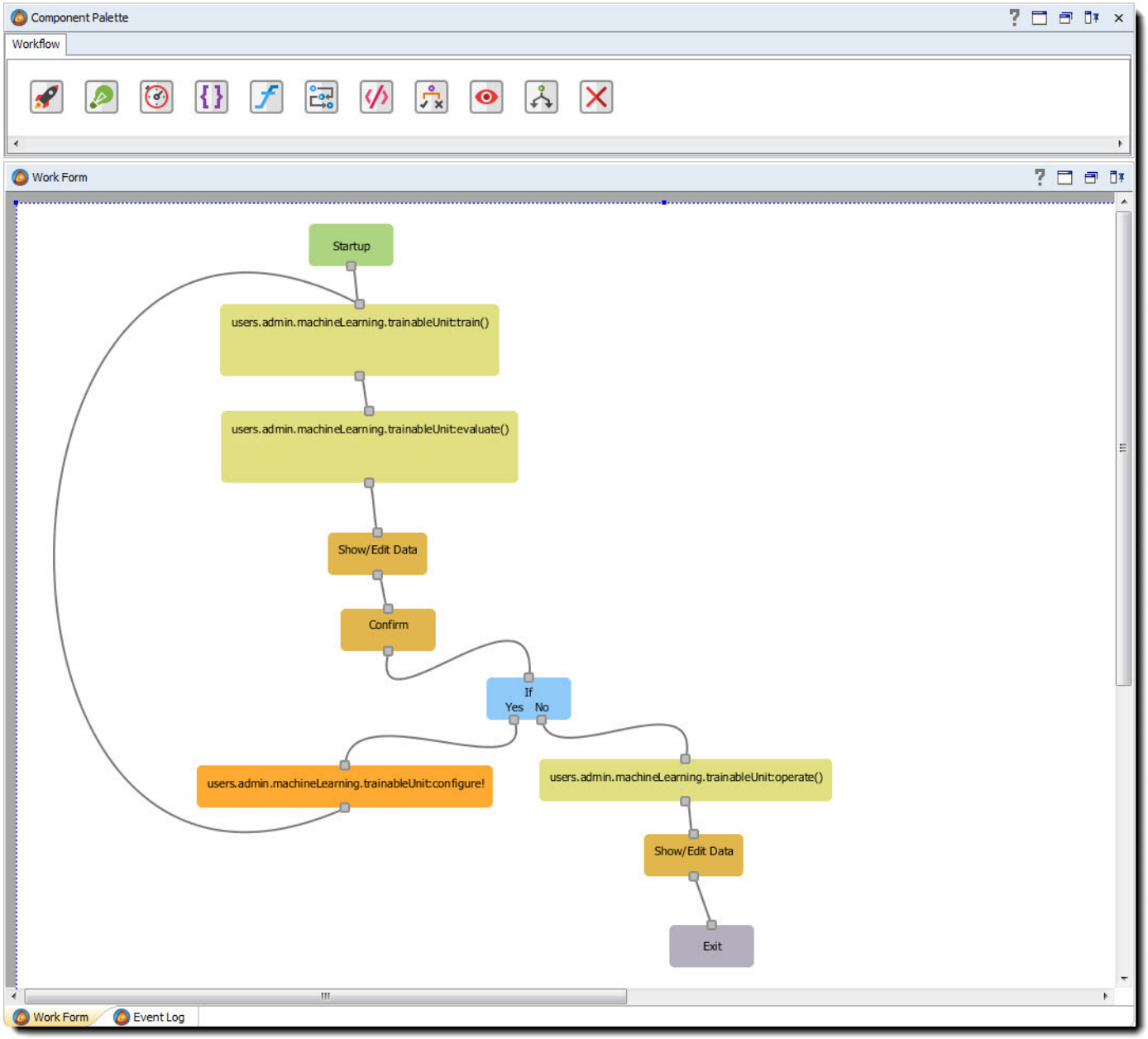
Это может стать еще одним способом проверки правильности нашего обучаемого модуля.

Если бы фактические значения были нам не известны, то есть нам было бы необходимо сделать предсказания на новых экземплярах данных, полученная таблица выглядела бы так:

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV (Actual)	MEDV (Predicted)	Error
0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.9	5.33	N/A	32.48358412698411	N/A
0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5.0	311.0	15.2	396.9	19.15	N/A	18.959321933621936	N/A
0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5.0	311.0	15.2	392.52	20.45	N/A	19.095688600288604	N/A
0.80271	0.0	8.14	0	0.538	5.456	36.6	3.7965	4.0	307.0	21.0	288.99	11.69	N/A	18.79144404761905	N/A
0.85204	0.0	8.14	0	0.538	5.965	89.2	4.0123	4.0	307.0	21.0	392.53	13.83	N/A	18.072097344322348	N/A
0.75026	0.0	8.14	0	0.538	5.924	94.1	4.3996	4.0	307.0	21.0	394.33	16.3	N/A	16.547704761904768	N/A
1.38799	0.0	8.14	0	0.538	5.95	82.0	3.99	4.0	307.0	21.0	232.6	27.71	N/A	14.991673015873014	N/A
0.06417	0.0	5.96	0	0.499	5.933	68.2	3.3603	5.0	279.0	19.2	396.9	9.68	N/A	21.819767063492076	N/A
0.17505	0.0	5.96	0	0.499	5.966	30.2	3.8473	5.0	279.0	19.2	393.43	10.13	N/A	21.435308333333328	N/A
0.18836	0.0	6.91	0	0.448	5.786	33.3	5.1004	3.0	233.0	17.9	396.9	14.15	N/A	19.960980952380954	N/A

ИСПОЛЬЗОВАНИЕ ПРОЦЕССОВ ПРИ РАБОТЕ С ОБУЧАЕМЫМ МОДУЛЕМ

Еще один способ работы с обучаемым модулем - использовать контекст процессов 895. Он предоставляет графические средства для работы с контекстами и их объектами. Следующий пример иллюстрирует типичный процесс машинного обучения:



Сначала мы обучаем обучаемый модуль, затем оцениваем его качество. Результаты функции **Оценить** показываются пользователю. Если пользователь решает, что значение какой-либо оценочной метрики неудовлетворительно, пользователю показываются свойства обучаемого модуля. Пользователь реконфигурирует обучаемый модуль (т.е. меняет некоторые гиперпараметры выбранного алгоритма. после этого процессы обучения и оценки повторяются. Когда пользователь сочтет результаты вычисления удовлетворительными, процесс продолжается функцией **Использовать**. Затем результаты функции демонстрируются пользователю.

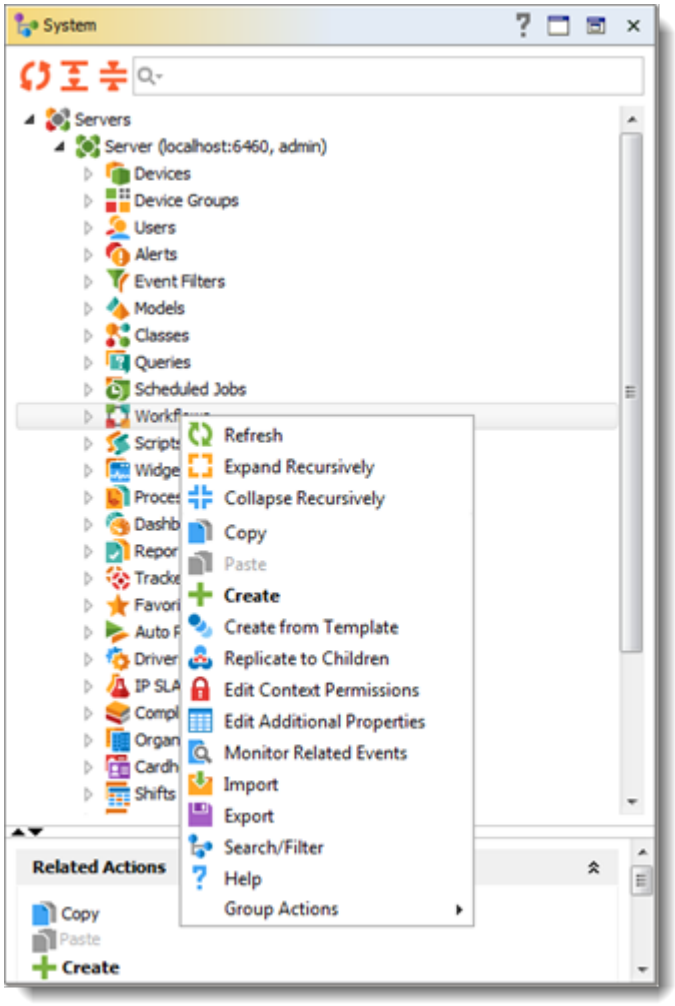
Используя контекст **Процессов**, можно полностью автоматизировать процесс настройки гиперпараметров.

18.28 Создание процесса

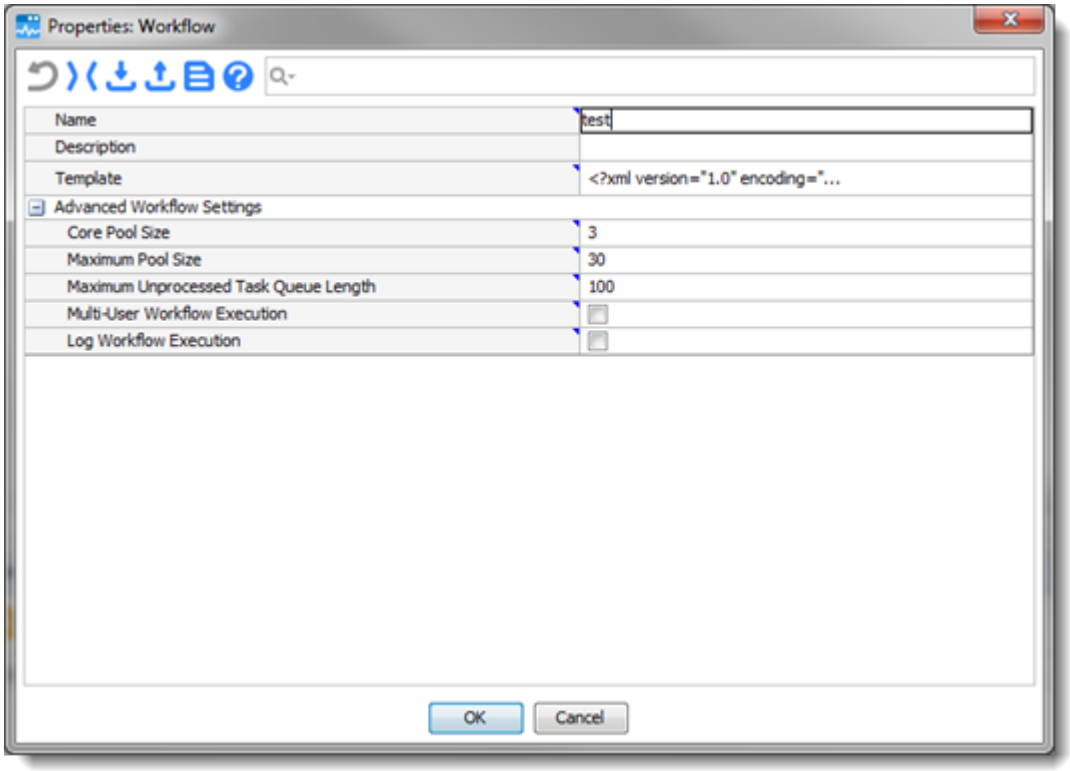
В этом уроке рассказывается, как построить процесс[895], который отоброзит сообщение "Hello, world!" на экране.

1. Начинаем создавать процесс

Перед тем как приступить к созданию процесса, найдите узел Процессы[1629] ( ) в системном дереве[370]. Во всплывающем меню выберите действие **Создать** ( ):



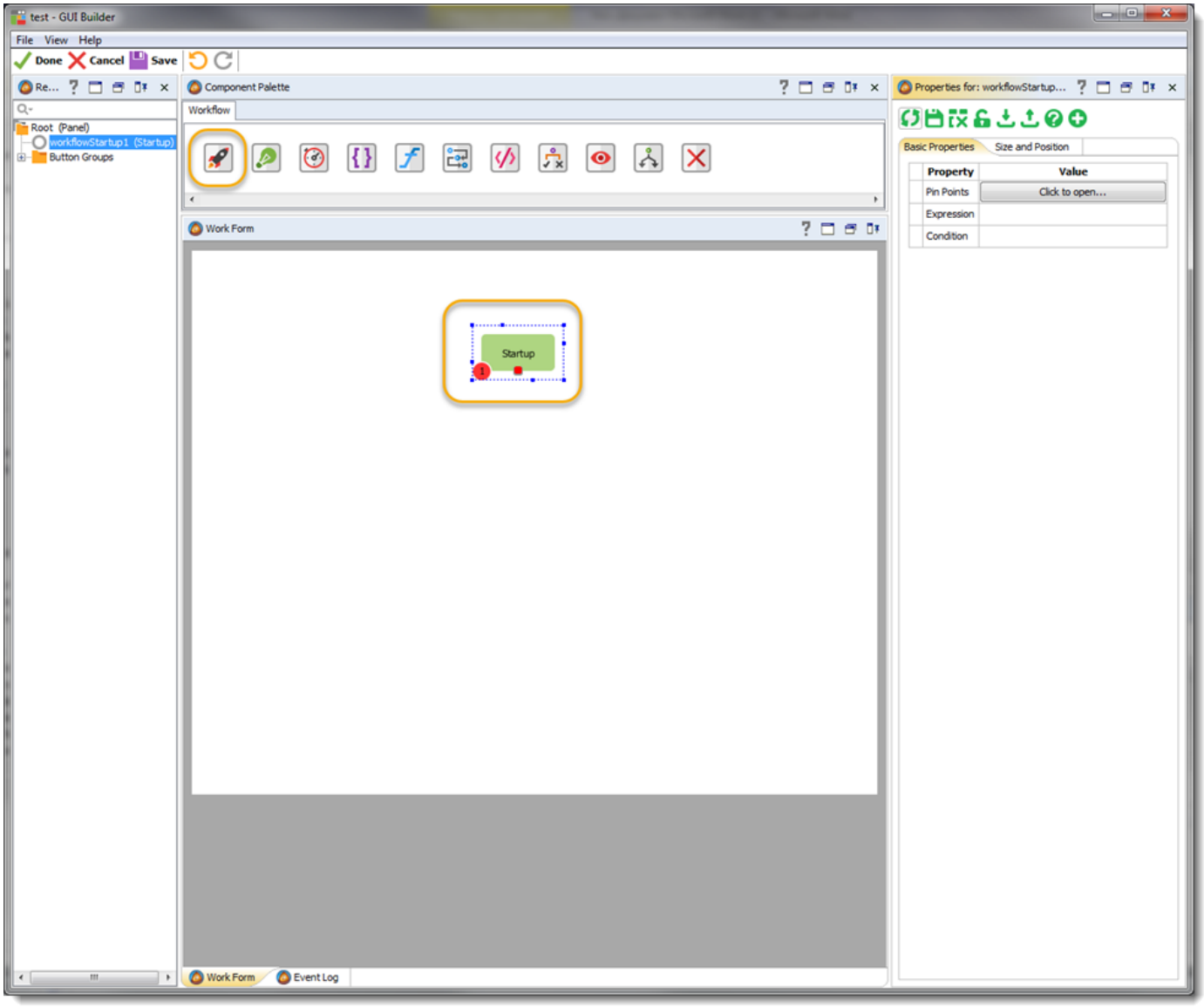
Откроется диалоговое окно свойства процесса[895]. Установите **Имя** как `test` и кликните OK.



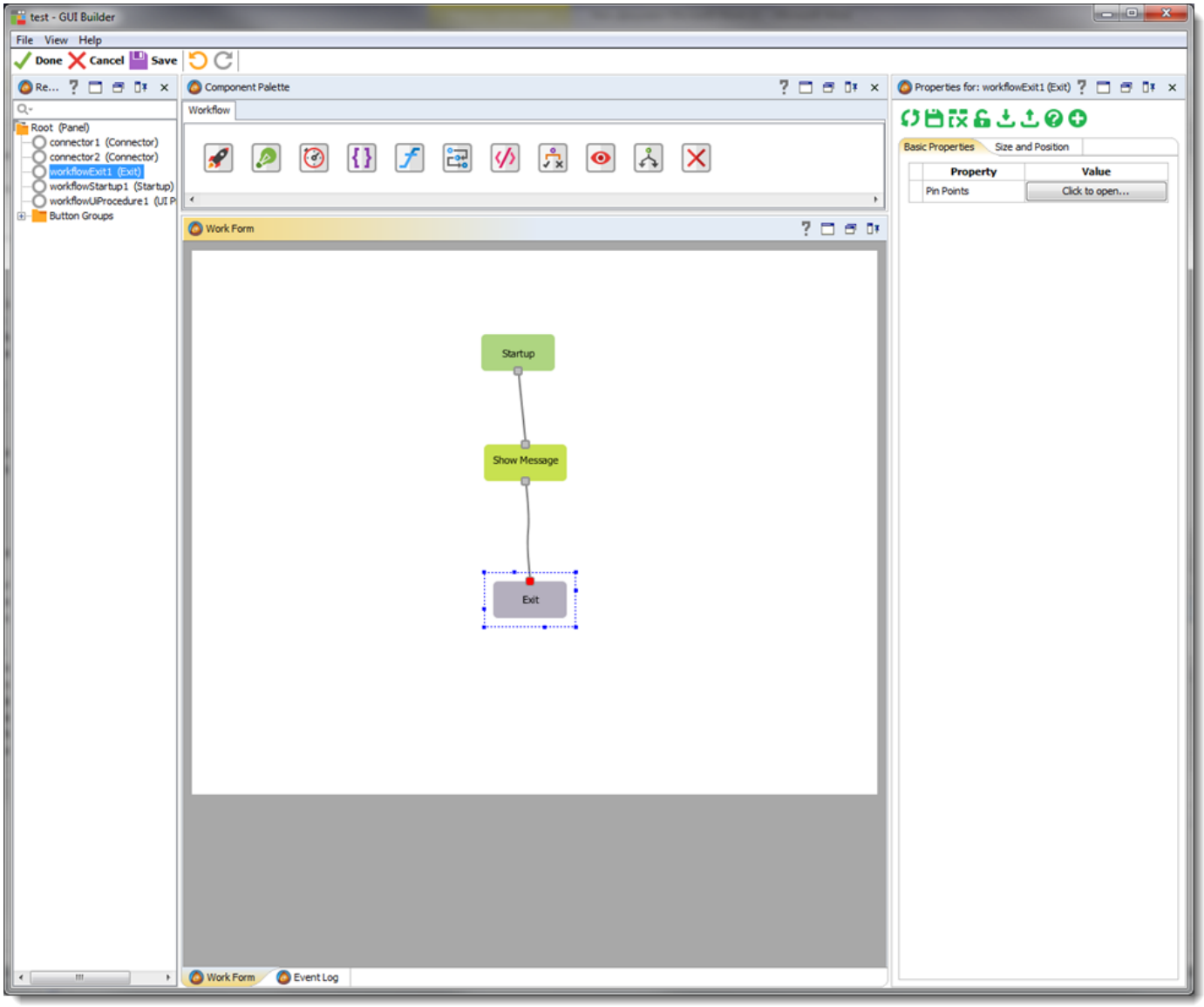
Это запустит редактор виджетов[423] с пустым процессом.

2. Редактирование процесса

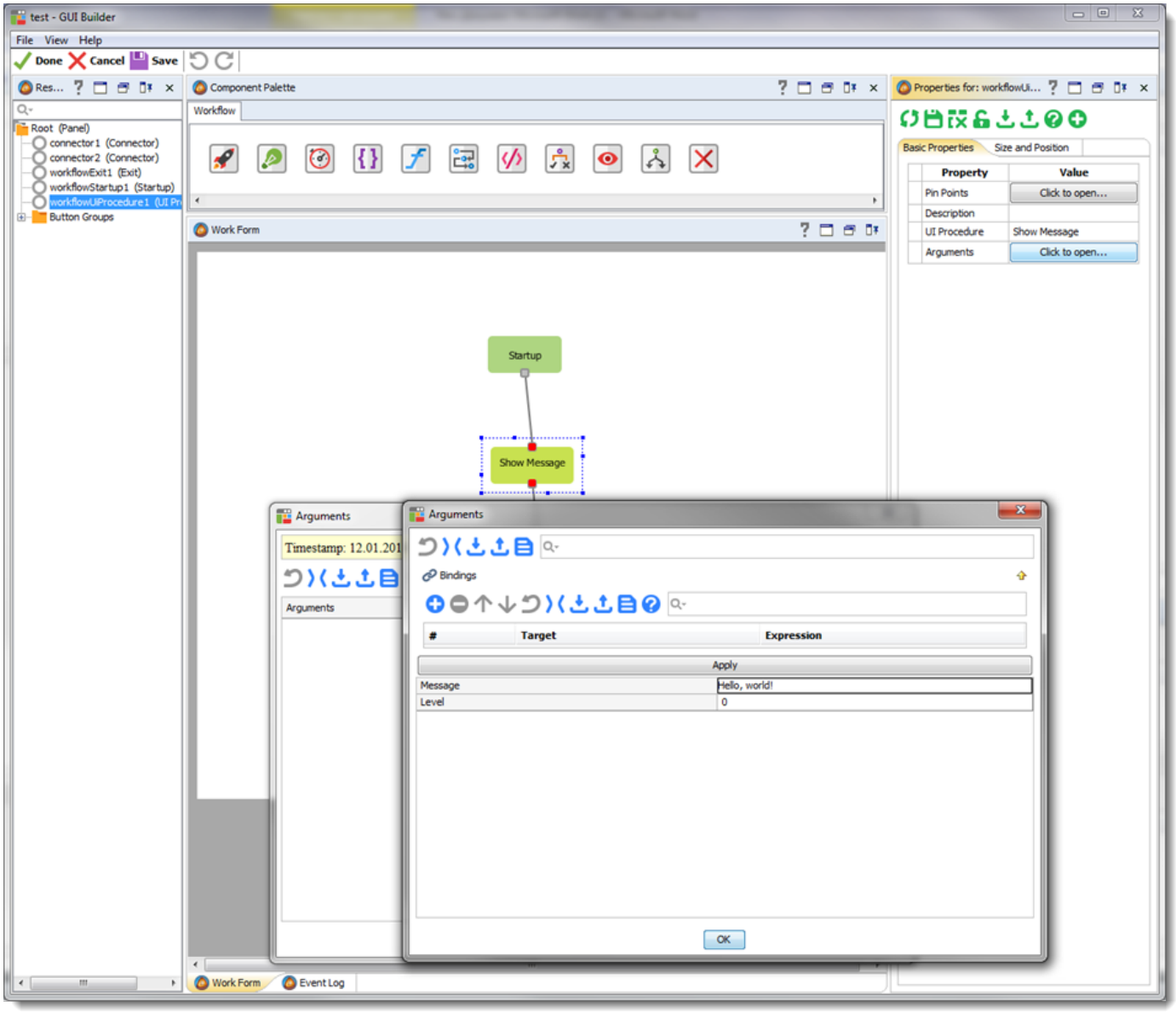
Перетащите компонент Запуск 899 с палитры компонентов 430 на корневую панель виджета. Это добавит новый компонент диаграммы к шаблону виджета 946:



Аналогично перетащите компонент UI процедура 901 и компонент Выход 905 и соедините между собой коннекторами 1273, как показано на рисунке:

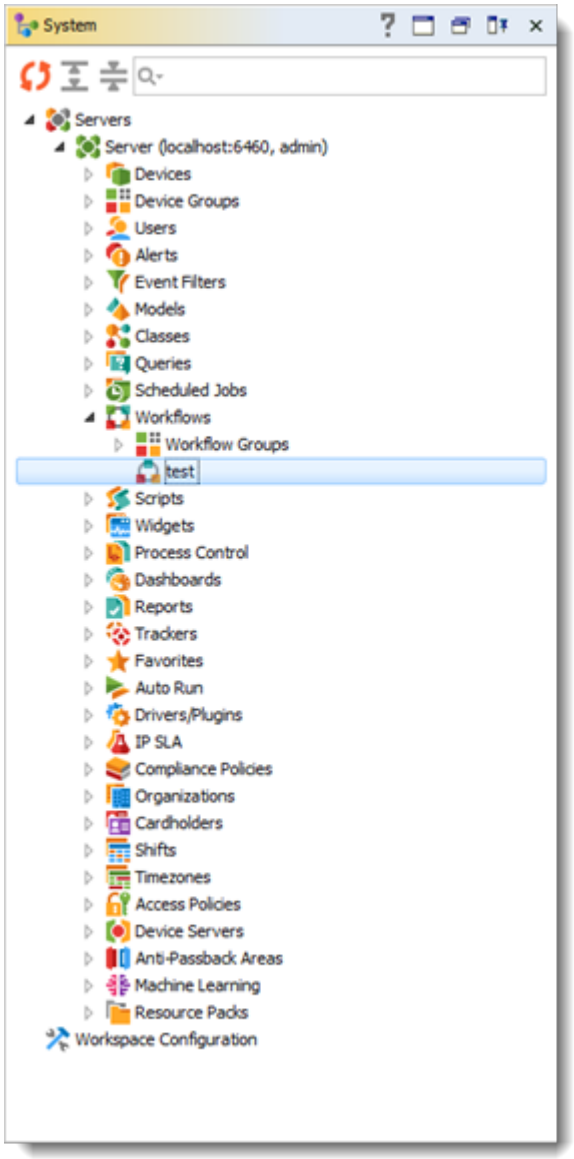


Далее, выберете компонент UI процедура "Show Message" и нажмите на свойство аргументы. В появившемся окне введите сообщение "Hello, world!" и кликнете Ok:



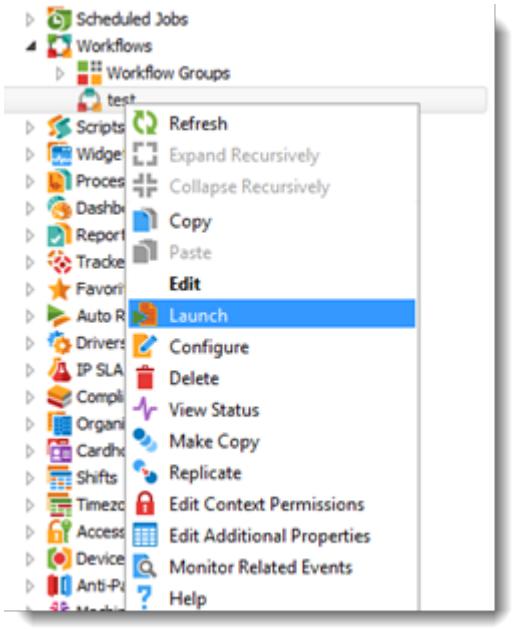
3. Сохранение программы

Шаблон готов. Кликните **Готово** ( ). Теперь должен быть создана новый процесс 1630 ( ), который появится в системном дереве 370:

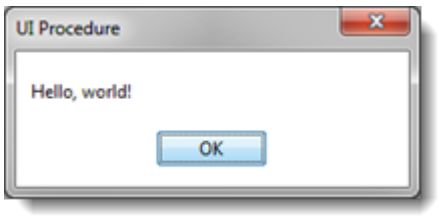


4. Запуск программы

Пора увидеть наш новый процесс в действии. Щелкните правой кнопкой мыши по процессу в системном дереве 370 . Во всплывающем меню выберите действие **Запустить** ( ):



В результате должно появиться диалоговое окно:



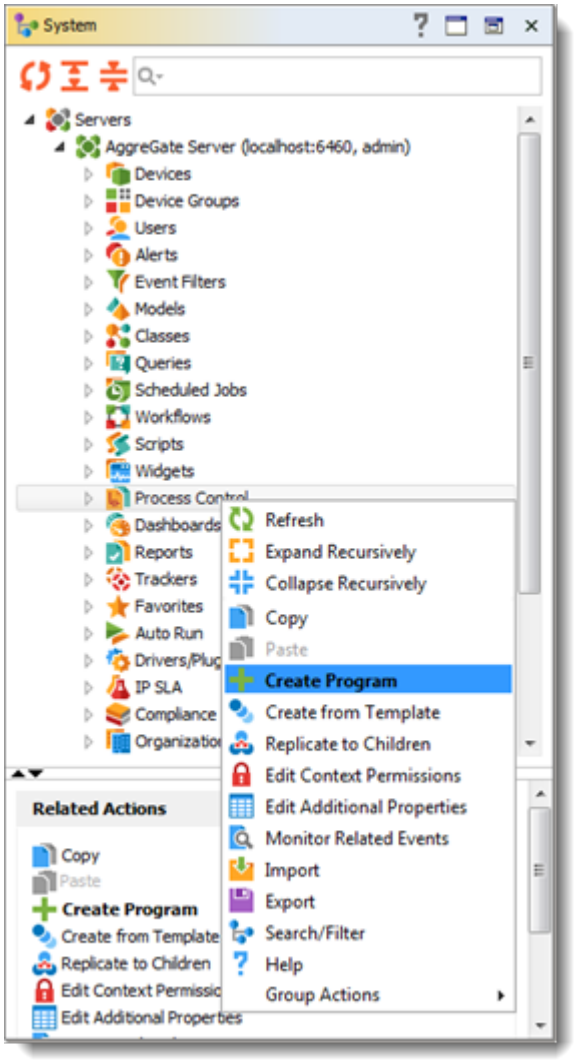
Кликнете Ok и похвалите себя. Вы написали свой первый процесс.

18.29 Построение Управление процессами программы

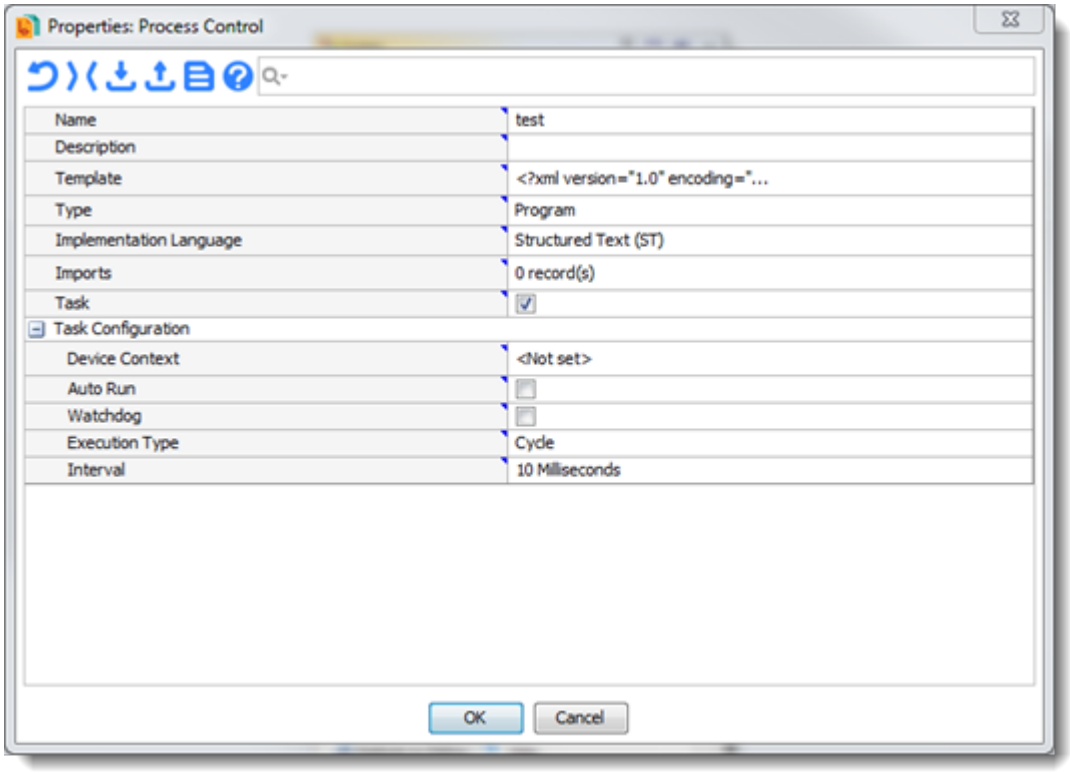
В этом уроке рассказывается, как построить Управление процессами программу.

1. Начинаем создавать программу

Перед тем как приступить к созданию программы, найдите узел Управление процессами () в системном дереве[370]. Во всплывающем меню выберите действие **Создать программу** ():



Откроется диалоговое окно свойства программы[454]. Установите **Имя** как `test`, а **Задачу** как `true` и кликните OK.

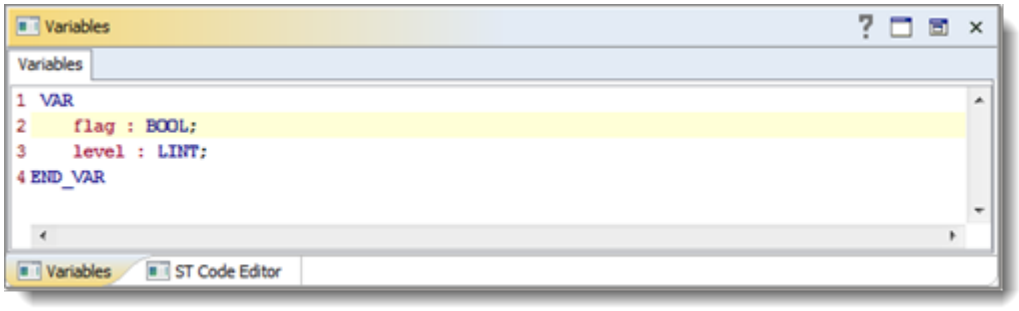


Это запустит AtomMind IDE 443 с пустой программой.

2. Редактирование программы

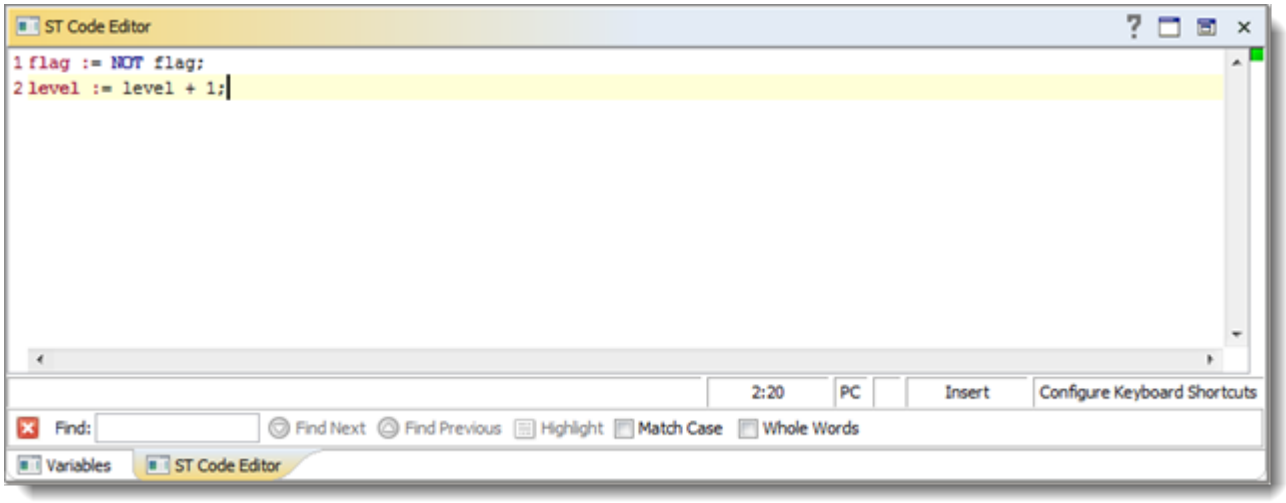
Создадим переменные в области переменных 449 согласно следующему **тексту**:

```
VAR
    flag : BOOL;
    level : LINT:
END_VAR
```



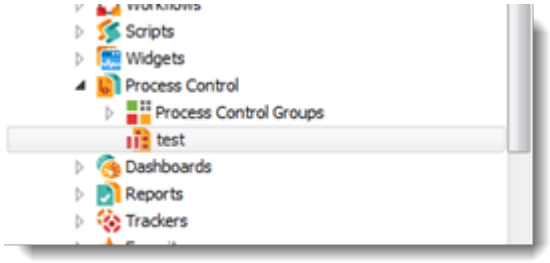
Создадим тело программы в рабочей форме 448 согласно следующему **тексту**:

```
flag := NOT flag;
level := level + 1;
```



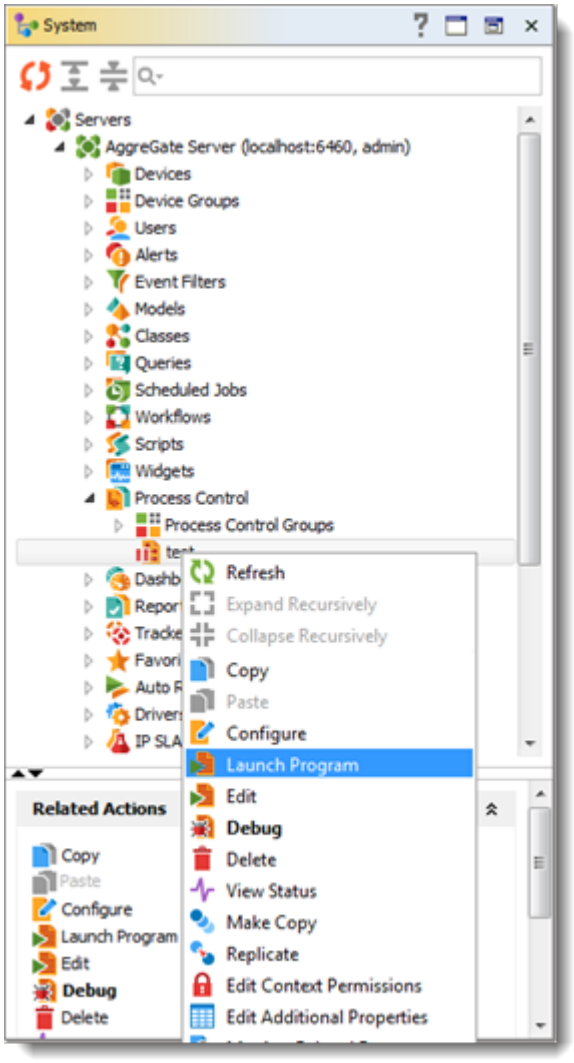
3. Сохранение программы

Шаблон готов. Кликните **Готово** ( ). Теперь должен быть создана новая программа ( ), которая появится в системном дереве 370:



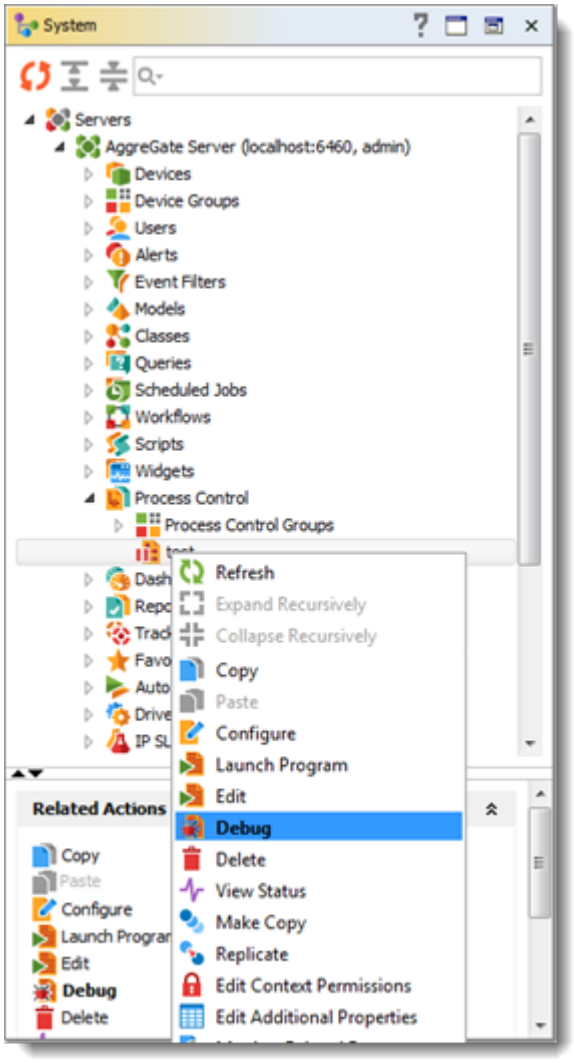
4. Запуск программы

Пора увидеть нашу новую программу в действии. Щелкните правой кнопкой мыши по программе в системном дереве 370. В контекстном меню выберете действие **Запустить программу** ( ):



5. Отладка программы

Для запуска отладки щелкните правой кнопкой мыши по программе в Системном дереве 370. В контекстном меню выберете действие **Отладка** ( ):



18.30 Интеграция с KAFKA

В этом уроке объясняется, как использовать Коррелляторы событий 774 для получения событий из источников данных Kafka.

Зависимости

Для использования Kafka в качестве источников данных, необходимо установить следующие зависимости:

- Плагин Коррелляторы Событий 774
- расширение io-kafka (версия 4.1.2)
- расширение siddhi-map-json
- библиотеку kafka-clients

Все расширения должны быть установлены в директорию расширений, указанную в свойствах плагина 774 Коррелляторы Событий. Библиотека Kafka-clients должна быть установлена в поддиректорую \lib установочной папки AtomMind.

Об источниках данных KAFKA

В Kafka сообщения создаются *продюсерами*. Продюсером может быть, например, устройство, отправляющее данные о температуре. Как только сообщение от продюсера достигает кластера Kafka, оно сохраняется и публикуется в *топике*. *Получатели* могут подписаться на топики, чтобы получать сохраненные сообщения.

В каждом топике есть одна или более *партиций*. Партиция действует как отдельный канал внутри топика и имеет идентификатор для доступа к ней. Партиции нумеруются, начиная с 0. Сообщения присваиваются определенной

партиции на основании значения поля ключа, хранящегося в сообщении. Например, если несколько устройств включат свои идентификаторы в сообщения, каждому устройству может быть назначена отдельная партиция.

Движок коррелятора выступает в роли *потребителя* для топиков Kafka. Поток Kafka может подписаться на топик и получать данные из всех его партиций. Когда поток получает события, его можно использовать как обычный поток коррелятора 775. Например, вы можете отфильтровать входящий поток или обнаружить шаблоны во входящих событиях.

источники данных Kafka

Как и любой другой поток, потоки Kafka используют декораторы для определения параметров потока.

Более подробно о том, как работают потоки коррелятора событий, см. в разделе Скрипты коррелятора событий 775.

Пример входного потока Kafka:

```
@Source(
    type="kafka",
    bootstrap.servers="kafka1.example.com:9091, kafka2.example.com:9092",
    topic.list="example", group.id="correlator",
    threading.option="single.thread",
    @map(type='json'))

define stream InKafkaTemperature (device string, temperature int);
```

Параметры потока Kafka @Source:

Параметр	Опциональный	Описание
type	Нет	Тип входного потока. Для Kafka источников значение всегда `kafka`.
bootstrap.servers	Нет	Список хостов Kafka, разделенных запятой. Движок коррелятора будет получать события от указанных хостов.
topic.list	Нет	Список топиков, разделенных запятой. Поток будет получать события из всех указанных топиков.
group.id	Нет	Имя группы получателей. Если вы хотите разделить сообщения из одного топика между несколькими входящими потоками, используйте то же самое имя группы. Это позволит избежать получения потоками дублей событий.
threading.option	Нет	Определяет способ обработки источника. Возможные значения: • **single.thread**. Источник будет работать на одном потоке. • **topic.wise**. Каждый топик будет работать на отдельном потоке. • **partition.wise**. Для каждой партиции будет использован отдельный поток.
seq.enabled	Да	Опция используется, когда важна последовательность получаемых событий. В этом случае, поток должен определить признак, который будет выступать идентификатором. Параметр необязательный и по умолчанию выставлен на `false`.
is.binary.message	Да	Если события в бинарном формате, этот параметр должен быть выставлен на `true`. Параметр необязательный и по умолчанию выставлен на `false`.
topic.offset.map	Да	Определяет офсеты топиков в формате `<topic> = <offset>, <topic> = <offset>`. Офсет будет определять, сколько сообщений от начала топика будут пропущены. Например, если значение офсета равно `100`, то первые 100 сообщений будут пропущены, когда движок коррелятора будет читать топик. Параметр необязательный, и по умолчанию сообщения не пропускаются.
optional.configuration	Да	Определяет все остальные параметры настроек получателя. Формат `parameter:value, parameter:value`. Параметр необязательный.
@map	Нет	Маппер входных данных. Расширение Kafka поддерживает входные данные в форматах текс, xml, json, а также в бинарном формате. Этот формат можно указать в параметре маппера. Например, для получения данных в формате json укажите маппер `@map(type='json')` для входного потока.

Пример

Следующий скрипт получает данные из семи партиций (0-6) топика *температура*. Поток подключится к двум серверам из кластера Kafka: kafka1.example.com:9091 и kafka2.example.com:9092. Входящий поток имеет два поля: устройство и температура. Для любых событий со значением температуры выше 100, генерируется событие в исходящем потоке.

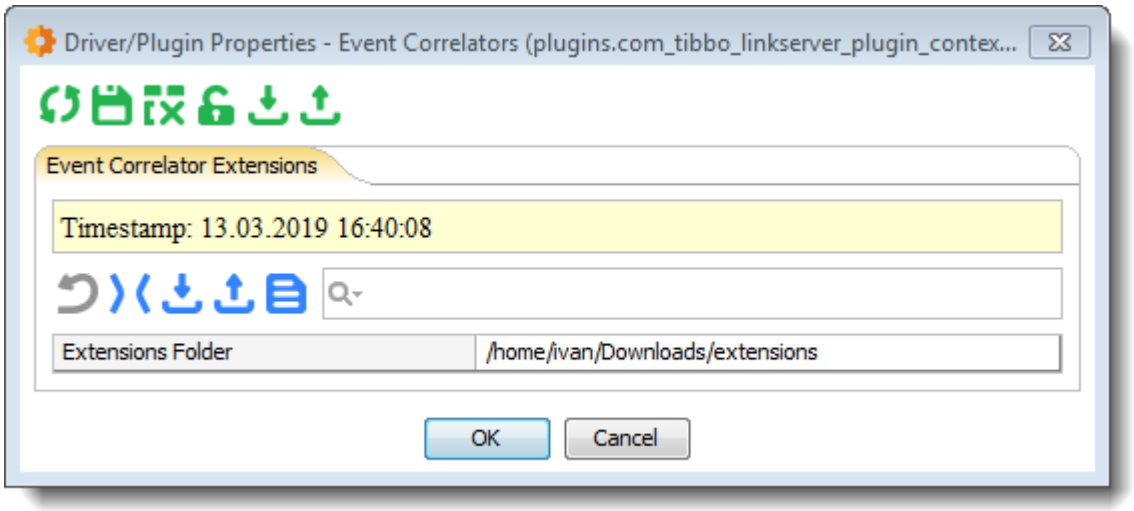
```
@Source(
    type="kafka",
    bootstrap.servers="kafka1.example.com:9091, kafka2.example.com:9092",
    topic.list="temperature", group.id="correlator",
    threading.option="single.thread",
    partition.no.list="0,1,2,3,4,5,6",
    seq.enabled="false",
    is.binary.message="false",
    @map(type='json'))

define stream InKafkaTemperature (device string, temperature int);

 @Sink(type = 'internalEvent', event='test', @map(type='internalEvent'))
define stream OutKafkaTemperature (device string, temperature int);

from InKafkaTemperature[temperature > 100]
select
    device as message,
    temperature as temperature
insert into OutKafkaTemperature;
```

18.31 Интеграция с RabbitMQ

В этом уроке объясняется, как можно установить соединение с брокером сообщений RabbitMQ, используя контекст Коррелятора событий.

1. Настройка папки расширения

Прежде всего, необходимо заполнить поле Extensions Folder контекста plugins.com_tibbo_linkserver_plugin_context_correlators. Все скаченные расширения коррелятора нужно поместить в эту папку.



2. Загрузка расширения RabbitMQ

Скачайте плагин RabbitMQ, используя следующую ссылку: https://github.com/siddhi-io/siddhi-io-rabbitmq. Поместите плагин в папку, заданную на предыдущем шаге.

3. Загрузка расширения мапера

Чтобы правильно осуществлять разбор и обработку входящего потока данных, коррелятору необходим соответствующий плагин мапирования. Список доступных плагинов можно найти по ссылке: https://siddhi-io.github.io/siddhi/extensions/.

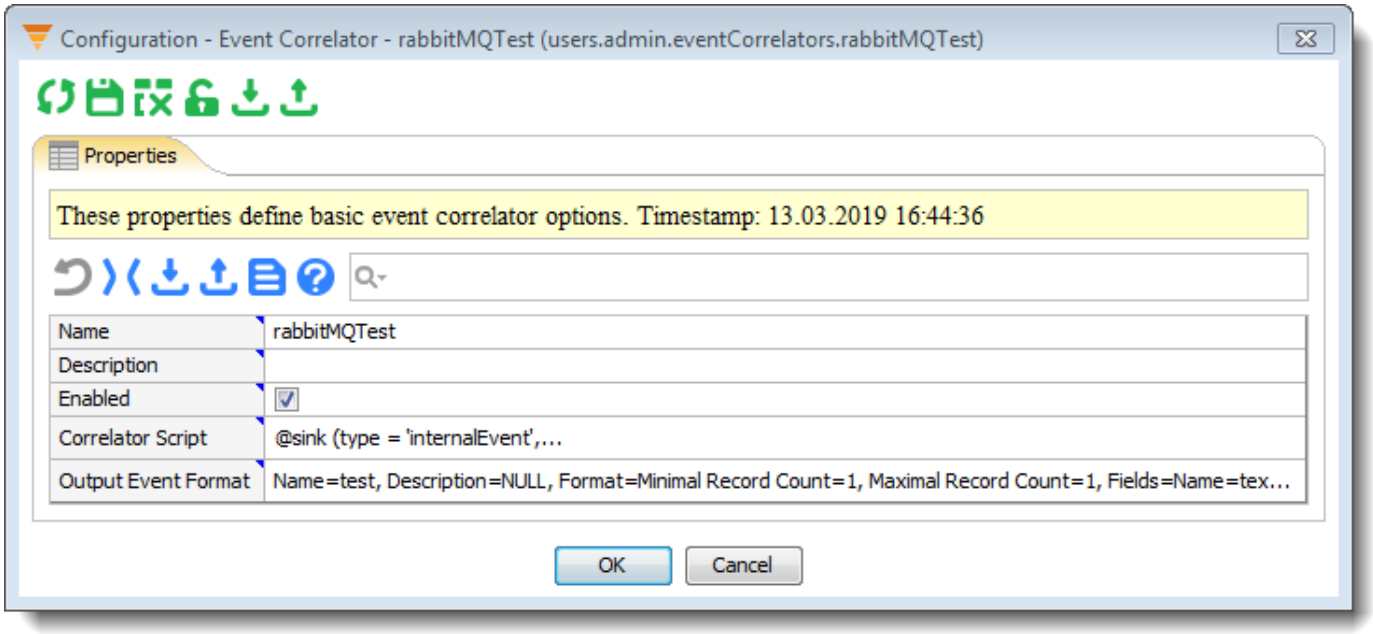
В данном руководстве мы будем работать в простыми текстовыми сообщениями, поэтому вам нужно скачать соответствующее расширение по ссылке: https://wso2-extensions.github.io/siddhi-map-text/.

Поместите плагин в папку, заданную на первом шаге. После этого перезапустите сервер.

После перезапуска сервера RabbitMQ сущности (источник и приемник), а также текстовый маппер, будут доступны в скриптах Коррелятора событий.

4. Настройка коррелятора событий

Теперь мы можем создать и настроить коррелятор событий.

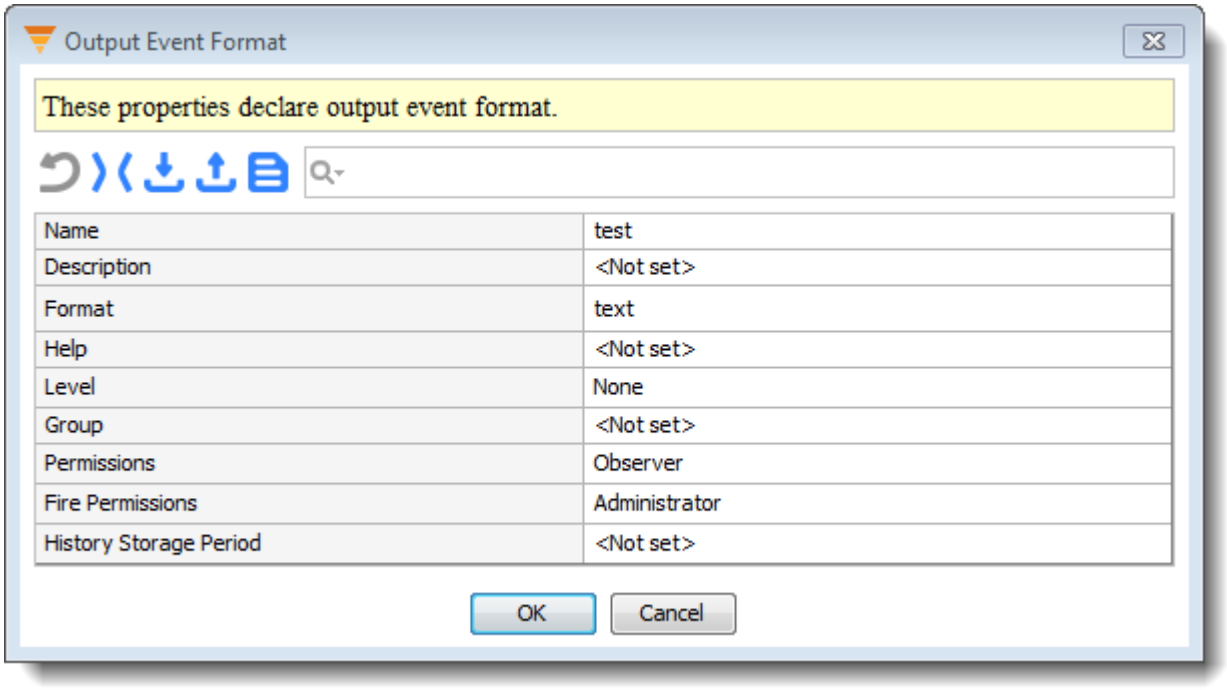


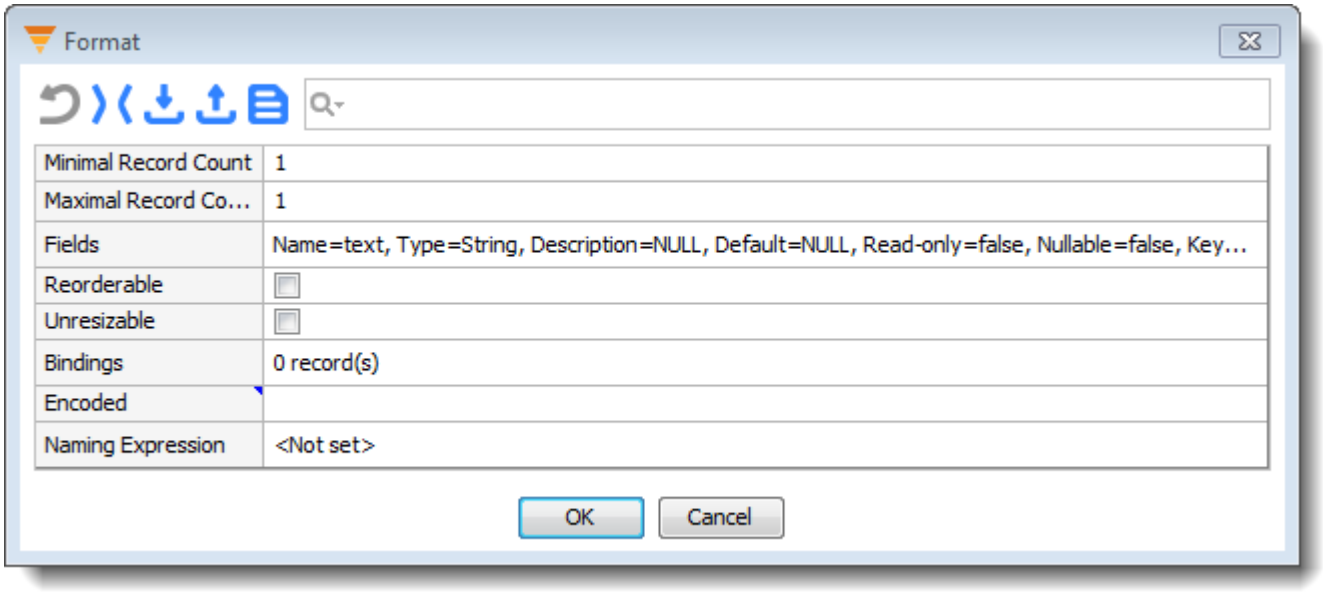
Поместите следующий скрипт в поле Correlator Script:

```
@sink (type = 'internalEvent',
event = 'test',
@map(type = 'internalEvent'))
define stream FooStream (text string);
@source(type ='rabbitmq',
uri = 'amqp://guest:guest@localhost:5672',
exchange.name = 'logs',
@map(type='text'))
define stream BarStream (text string);
from BarStream select text insert into FooStream;
```

Этот скрипт описывает, как сообщения, отправленные в обменник "logs" локального RabbitMQ брокера (`BarStream`), перехватываются и преобразуются в набор событий AtomMind Server (`FooStream`). Предполагается, что входящие сообщения поступают в текстовом формате (`@map(type='text')`). Ожидается, что входные данные будут выглядеть так: `text:'Message'`. После получения сообщения его данные будут извлечены и помещены в поле `text` соответствующего события AtomMind Server.

Для того, чтобы этот скрипт корректно выполнялся, вам также придется настроить событие `test` контекста Коррелятора событий:





19 Продукты на основе платформы AtomMind

Помимо самой платформы AtomMind, ТВЭЛ предлагает различные основанные на ней вертикальные продукты.

Каждый продукт технически является набором модулей платформы и распространяется как:

- отдельный дистрибутив, который можно скачать с сайта ТВЭЛ
- наборы модулей, доступные в магазине приложений 1321 AtomMind

Этот раздел описывает все продукты, которые поддерживаются ТВЭЛ. В то же время здесь не описываются продукты и сервисы, разработанные нашими партнерами по всему миру.

19.1 Сетевое управление и мониторинг

AtomMind Network Manager - это законченное решение уровня предприятия для сетевого управления по IP, мониторинга и надзора за системой, основанное на платформе AtomMind. Оно может быть скомбинировано с другим расширениями AtomMind (SCADA/HMI, Автоматизация зданий, Учет рабочего времени, Контроль доступа, Управление транспортом и т.д.) для создания интегрированного центра управления.

AtomMind Network Manager обеспечивает единую сеть мониторинга объектов и инструментов, наряду с широкими возможностями их настройки, модификации и расширения.

Network Manager создан как независимая от платформы заказчика система баз данных. ТВЭЛ предоставляет пакеты для Windows, Linux/Unix и Mac OS .

Основные особенности AtomMind Network Manager

AtomMind Network Manager обеспечивает высокопроизводительные средства мониторинга и управления для крупных и комплексных корпоративных сетей:

- Усовершенствованное сетевое обнаружение 1811 обеспечивает автоматическое определение местонахождения, регистрацию и конфигурацию устройств и сервисов для управления и мониторинга.
- Топология сети 1821 для визуального, динамически обновляемого представления сетевой инфраструктуры, топологии и состояния.
- Мощная система тревог, отчетов, построения диаграмм и визуализации объектов. Интегрированный GUI-разработчик и редактор отчетов.
- Множество встроенных доступных/работоспособных 1798 и производительных 1799 инструментов анализа по всем основным областям сетевого управления и мониторинга, включая SNMP управление, управление сервером/маршрутизатором/коммутатором, мониторинг баз данных, сетевого трафика и пропускной способности, мониторинг виртуальной среды и т.п.
- Мониторинг приложений, служб и процессов.
- Поддержка стандартных протоколов и технологий наряду с передовыми возможностями обработки данных для нестандартного сетевого оборудования.
- Усовершенствованная консолидация SNMP-ловушек, Syslog сообщений и Windows Event Log уведомлений.
- Поддержка WMI: выборка класса, экземпляры класса и их метаданные; выполнение WQL запросов; получение и обработка WMI событий; применение WMI методов к объектам.
- Корпоративные возможности настройки, расширения и масштабирования системы сетевого управления; интеграция со сторонними приложениями через открытые протоколы.

19.1.1 Начало работы

Первыми шагами по использованию AtomMind Network Manager для управления и мониторинга Вашей сети и ее компонентами являются:

- Установка 1789 программного обеспечения AtomMind Network Manager.
- Обнаружение 1811 Вашей сети.
- Конфигурация 1789 сетевых элементов, предназначенных для управления/мониторинга.
- Мониторинг 1790, осуществляемый за счет доступных сервисов.
- Контроль 1790 за Вашими управляемыми элементами.

19.1.1.1 Установка AtomMind Network Manager

Дистрибутив AtomMind Network Manager включает в себя два отдельных инсталлятора: **Сервер** и **Клиент.**

AtomMind Network Manager Сервер выполняет функции сбора и обработки данных. **Клиент** - это часть **AtomMind Network Manager,** являющаяся программным средством для настольных систем рабочих станций операторов системы.

Установка и запуск

См. **Краткое руководство** для получения подробных инструкций по загрузке, установке и пуску AtomMind.

Если Вы устанавливаете AtomMind Network Manager с общего пакета AtomMind, убедитесь, что в процессе установки Вы выбрали пункт сетевое управление AtomMind Server.

Виджет быстрого запуска

При первом запуске в AtomMind Network Manager, Вы увидите виджет 943 **Быстрого старта,** который обеспечивает доступ к самым важным операциям:

Для отключения автоматического запуска данного виджета :

- Найдите функцию **Автозапуск** ( ) в Системном дереве 370
- Щелкните правой кнопкой мыши на автозапуске **Быстрого старта**
- Выберите в контекстном меню пункт **Конфигурация** ( )
- Снимите флажок **Включено** и нажмите ОК

19.1.1.2 Конфигурация

Технология управления сетью предполагает наличие двух сторон: управляющей (в нашем случае AtomMind Network Manager) и управляемой (т.е. сетевых устройств и сервисов/приложений, обеспечивающих их работу). Обе стороны должны быть сконфигурированы должным образом.

Конфигурация AtomMind Network Manager

Конфигурация системы относится к разделу Конфигурация AtomMind Network Manager 1803 . Если Вы не знакомы с продуктом, проверьте следующие задачи:

- **Регистрация устройств для управления и мониторинга**. Вы можете сделать это как вручную 1805 так и автоматически 1807 (используя обнаружение). Для лучшего понимания различных аспектов данной задачи

рекомендуется настроить сначала несколько устройств вручную. На следующем этапе, когда Вам потребуется управлять и следить за большим количеством устройств, Вы освоите автоматические операции обнаружения.

- Задача ручной регистрации устройств тесно связана с задачей по их **конфигурации,** т.е с корректировкой параметров 1805 для них. Вы должны хорошо понимать эту задачу, поскольку она является решающей для успешного сетевого управления и мониторинга.
- **Установка средств мониторинга** (т.е. тревог, отчетов или диаграмм) для зарегистрированных устройств. См. раздел мониторинг 1790, чтобы ознакомиться с доступными средствами и типами мониторинга.

Конфигурация внешней среды

Задача по **конфигурации управляемой стороны** не может быть полностью описана, так как существует огромное количество ПО и оборудования, которым можно управлять и следить. Если Вы не знакомы с проблемами конфигурации Ваших устройств, сервисов и приложений, то, на данном этапе, оставьте их настройки по умолчанию. С получением большего опыта и навыков, Вы вернетесь к этому вопросу и настроите управляемые элементы для лучшего взаимодействия с AtomMind Network Manager.

См.раздел Конфигурация внешней среды 1809 для получения большей информации о распространенных проблемах.

19.1.1.3 Мониторинг

Цель сетевого мониторинга состоит в том, чтобы понять что происходит с Вашей сетью и ее компонентами. Прежде всего, это включает в себя сбор, хранение и обработку данных, полученных от управляемых сетевых элементов. Данная задача частично выполнена путем конфигурации системы мониторинга 1789.

Сбор данных является лишь первым шагом в задаче мониторинга. Собранные данные должны быть обработаны, т.е проанализированы для того, чтобы:

- Обнаружить сетевые проблемы
- Показать их первопричины
- Понять, как они могут быть установлены
- Выбрать корректирующие решения и выполнить их.

Одной из важнейших задач данного анализа является визуализация данных. Это не означает лишь создание различных диаграмм, а в целом, любой вид представления данных в форме, удобной для:

- Понимания текущей ситуации и тенденции
- Выявления внутренней связи параметров
- Нахождения события, представляющего интерес.

19.1.1.4 Управление

AtomMind Network Manager предоставляет набор операций, которые Вы можете использовать для **управления** Вашими устройствами:

- Различные SNMP-совместимые устройства могут быть сконфигурированы или настроены для выполнения определенных действий с использованием SNMP 1954.
- Компьютерами, поддерживающими Windows ОС, можно управлять через WMI 1954.
- Если хост поддерживает SSH, для управления им может быть использовано удаленное выполнение скриптов 1955.
- Существует много действий сетевого управления 1955, предоставляемых AtomMind Network Manager. Эти действия могут быть вызваны в интерактивном режиме или автоматически(как корректирующие действия при тревоге 793 или запланированное задание 823).

19.1.2 Основные концепции AtomMind Network Manager

Всвязи с многообразием взаимосвязанных задач, сетевой мониторинг является сложной проблемой. Хотя AtomMind Network Manager может выглядеть лишь как набор полезных инструментов для этих задач, на самом деле это не так. Система объединяет и реализует в себе основные принципы и подходы по управлению сетью.

AtomMind Network Manager был разработан для того, чтобы охватить основные требования к управлению, сохраняя при этом широконастраиваемый и расширяемый характер, унаследованный от AtomMind. В этом случае,

AtomMind Network Manager рассматривается как **платформа** или **среда** для создания специализированных решений управления сетью.

В данной главе содержится информация о теоритических аспектах операций в AtomMind Network Manager. Она может быть легко пропущена в пользу решения практических задач, описанных в последующих главах.

19.1.2.1 Концепции сетевого управления

В настоящее время **сетевое управление** как термин получило множество значений, а следовательно нуждается в пояснении базовых понятий, которые будут далее использованы в руководстве.

Основной задачей AtomMind Network Manager является *функция* **сетевого управления**, то есть *"обеспечение работы сети и предоставляемых ею сервисов*".

В данном разделе содержится информация о важных терминах, понятиях, принципах и других аспектах сетевого управления.

19.1.2.1.1 Субъекты управления

AtomMind Network Manager имеет дело с сетями, конечными системами и приложениями, тем самым покрывая следующие три области сетевого управления:

- *Узконаправленное сетевое управление*: управление сетями связи и сетевыми ресурсами, обеспечивающими непрерывную связь. Например, к этому можно отнести мониторинг маршрутизаторов и коммутаторов и обеспечение должной координации их настроек.
- *Управление системой*: управление системами, которые подключены к сети. В качестве примера можно упомянуть мониторинг доступного объема хранилища на корпоративном сервере и уведомление пользователей о слишком высоком уровне использования памяти или о заканчивающемся свободном пространстве на жестких дисках.
- *Управление приложением*: управление развернутыми в системе сервисами и приложениями, которые соединены между собой по сети. Примером будет являться мониторинг и контроль Email серверов, различных TCP/UDP сервисов, баз данных и т.д.

19.1.2.1.2 Задачи управления

Функциональные возможности AtomMind Network Manager можно классифицировать по нескольким категориям:

Мониторинг

Осуществлять текущий контроль значит быть осведомленным о состоянии системы. Чтобы быть в состоянии управлять системой, прежде всего необходимо понять все, что имеет для нее значение: ее текущее состояние, поведение, причины и тенденции. **Сетевой мониторинг** несомненно является основной частью сетевого управления.

AtomMind Network Manager помогает быть осведомленным о том, что происходит с системой. Например, он позволяет определить такие проблемы, как отключение электричества, при этом приводя компоненты системы к сбою/замедлению как можно быстрее, в идеальном случае прежде, чем это повлияет на пользователей. Network manager обеспечивает это с помощью:

- *сбора информации* от устройств, сервисов, приложений и связанных с ними ресурсов (ЦП, хранилище, пропускная способность сети и т.п.)
- *хранения*, *обработки* и автоматического *анализа* собранной информации, производя при этом различные *количественные характеристики* такие как доступность, работоспособность, производительность системы и сети;
- *выявления* и *обработки* различных *событий,* сгенерированных управляемыми сетевыми элементами или инициированных непосредственно системой AtomMind в результате обработки и анализа информации.
- *представление* собранных данных и выявленных событий в удобной для пользователя визуальной/печатной форме.

Эти основные функциональные возможности дополнены богатым набором средств для настройки и расширения алгоритмов мониторинга и используемых инструментов. Кроме того, для обеспечения полноценного решения по управлению, AtomMind Network Manager может быть интегрирован практически с любой справочной службой или с системой по работе с жалобами клиентов (см. раздел Интеграция со справочной службой/системой по работе с жалобами клиентов 1956).

Контроль

Когда фактическая или потенциальная сетевая проблема обнаружена и диагностирована, она должна быть фиксирована или предотвращена соответственно. Такие корректирующие действия могут выполняться как автоматически, так и вручную, с применением соответствующих *операций управления* к управляемым устройствам, сервисам и/или приложениям. Это может включать в себя, например, изменение параметров конфигурации сетевого элемента или активицазию команды на удаленном устройстве, заставляющей его выполнить определенные операции. В целом, AtomMind Network Manager предоставляет расширяемый и настраиваемый набор корректирующих мер контроля.

Администрирование инфраструктуры управления

AtomMind Network Manager предоставляет различные возможности конфигурации, позволяющие пользователям создать определенную инфраструктуру мониторинга соответственно их потребностям. Это может быть сделано путем обнаружения, добавления или установки устройств, являющихся объектами мониторинга, путем настройки инструментов мониторинга или создания новых, путем подстройки AtomMind Network Manager для лучшей производительности и т.д.

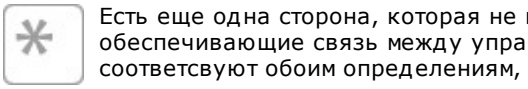
Управление активами

ИТ управление активами - другая типичная задача сетевого управления. Плагин 207 управления активами предоставляет возможности организации управления активами для оборудования и элементов ПО в сетевой среде. Для более подробного ознакомления обратитесь к разделу Управление_активами 1956.

19.1.2.1.3 Участники управления

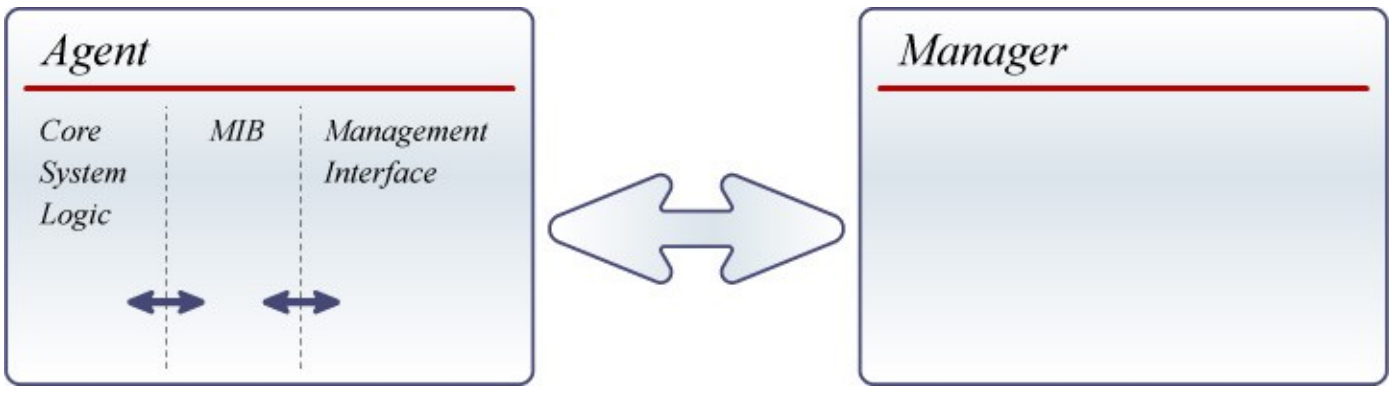
Есть две основные стороны, вовлеченных в "игру" управление:

- Управляемая сторона -- управляемая сеть состоит из множества устройств, сервисов и приложений (*сетевые элементы/детали/узлы*). *Устройство* - это сетевой элемент с присвоенным IP адресом. *Сетевой или IP хост* - другие термины, которые могут быть использованы в качестве синонимов. Устройство обеспечивает функциональные возможности с точки зрения *сервисов/приложений,* которые работают на нем. Управляемое устройство может являться устройством любого типа, включая, но не ограниченно, маршрутизаторы, серверы доступа, коммутаторы, мосты, концентраторы, IP телефоны, IP видеокамеры, хосты компьютера, принтеры. Примером сервисов и приложений является Web-сервисы, e-mail сервисы, системы управления базами данных и др. Один из наиболее важных сервисов, который будет предоставлен сетевым устройством - ICMP ping. Еще один важный тип сервиса, который может быть представлен устройством для управления - поддержка SNMP 1828. AtomMind Network Manager поддерживает много различных типов сервисов и устройств, которые более подробно описаны в разделе Мониторинг_приложения/сервиса 1888.
- Управляющая сторона -- люди, выполняющие управление и инструменты (*системы управления*) призваны для решения задач по управлению. AtomMind Network Manager - это пример подобной системы управления.



Есть еще одна сторона, которая не может легко быть классифицирована здесь: общие *носители,* обеспечивающие связь между управляющей и управляемой сторонами. Как ни странно, но носители соответсвуют обоим определениям, данным выше: как часть сети носители подвергнуты управлению, и следовательно относятся к управляемой стороне; но с другой стороны, как средство выполнения задач управления, носители можно отнести к управляющей стороне. Мы не углубляемся дальше и ограничиваемся тем фактом, что другая классификация здесь была бы более сложна и неуместна. Дальнейшие рассуждения по этому поводу мы оставляем для наших читателей.

Расширяя терминологию управления, мы вводим термины **Менеджер** и **Агент**.



Менеджер

Менеджер может быть любым видом систмы управления, которая выполняет свои обязанности путем отслеживания сетевых элементов и контроля за их деятельностью. Разнообразие функциональных возможностей, которые менеджер может обеспечить и архитектура системы безграничны. Роль менеджера может сыграть простая сетевая утилита, определенное приложение или огромная и сложная система сетевого управления.

Агент

Агент управления представляет собой отдельный сетевой элемент, который является сетевым устройством, сервисом или приложением. Как сетевой элемент, агент имеет некие существенные функциональные возможности, упомянутые здесь как его основная логика. Кроме того, агент предоставляет интерфейс управления, который позволяет управлять им и дает информацию о сетевом элементе, которая может быть использована менеджерами. Таким образом, агент может быть представлен в виде компонента, состоящего из трех частей: основная логика, интерфейс управления, информационная база управления (MIB).

ОСНОВНАЯ ЛОГИКА

Мы используем термин **Основная логика,** чтобы обратиться к существенным функциональным возможностям сетевого элемента. Он может включать в себя операции, которые проводит данный элемент, алгоритмы, составные части и отношения между ними и др.

ИНТЕРФЕЙС УПРАВЛЕНИЯ

Интерфейс управления обуславливает взаимодействие между агентом и менеджером, обеспечивая выполнение запросов и команд менеджером, отправляя обратно запрошенные данные и, при необходимости, изменяя конфигурацию элемента.

Интерфейс управления может быть реализован как специализированный интерфейс, выполняющий исключительно задачи управления. Подобными примерами являются *ICMP* и *SNMP* сервисы. Но другие интерфейсы также могут быть использованы для мониторинга и контроля определенных аспектов сетевого элемента. Например, доступность и работоспособность многих сервисов и приложений может быть проконтролирована с помощью доступа и использования их основных функциональных возможностей. Скажем, мы можем проверить работоспособность e-mail системы при подключении и соединении с сервисом через определенный протокол(POP3/IMAP/SMTP). Кроме того, мы можем убедиться, что e-mail система работает должным образом при отправке сообщения и проверки его доставки получателю.

ИНФОРМАЦИОННАЯ БАЗА УПРАВЛЕНИЯ

Информационная база управления (MIB) представляет собой часть информации управления о сетевом элементе, предлагая абстракцию управляемых элементов, используемых в целях управления.

Вся информация управления относительно сетевого элемента может быть расценена как (виртуальное) хранилище данных, которое содержит описание всех физических и логических аспектов объекта. Эти данные могут охватывать информацию о мониторинге, например, о конфигурации оборудования (доступных портов или сетевых карт), о его текущем состоянии и характеристиках; так же, как и параметры конфигурации, позволяющие менеджеру контролировать деятельность элемента.

Информационные базы управления не являются реальными базами данных; они лишь отражают информацию об основных сетевых элементах. Стоит отметить, что часто информационные базы данных представляют собой снимки данных, в определенный момент полученные от основы, которые периодически обновляются. Периоды обновления и задержки в линии связи должны быть приняты во внимание операциями управления.

Существует множество способов для представления и обмена информацией управления между агентами и менеджерами. Одним из самых распространенных является *SNMP,* который встроен в AtomMind Network Manager. Просмотрите раздел *SNMP Управление* 1828 для получения более подробной информации о SNMP, для чего и как это может использоваться в AtomMind Network Manager.

19.1.2.1.4 Взаимодействия управления

Сетевое управление, так же как и любой другой вид управления, основано на связи и взаимодействии. Менеджер 1792 взаимодействует с агентами 1795 для мониторинга и контроля их деятельности. Существует два основных подхода к организации взаимодействия между ними. Они соответствуют двум возможным ответам на вопрос "Кто инициирует взаимодействие?": взаимодействие может быть инициировано как менеджером, так и агентом. Первый подход называется опрос, и управление в свою очередь **основано на опросе (пассивное),** в то время как последний подход **основан на событии (активное)**. AtomMind Network Manager поддерживает оба подхода.

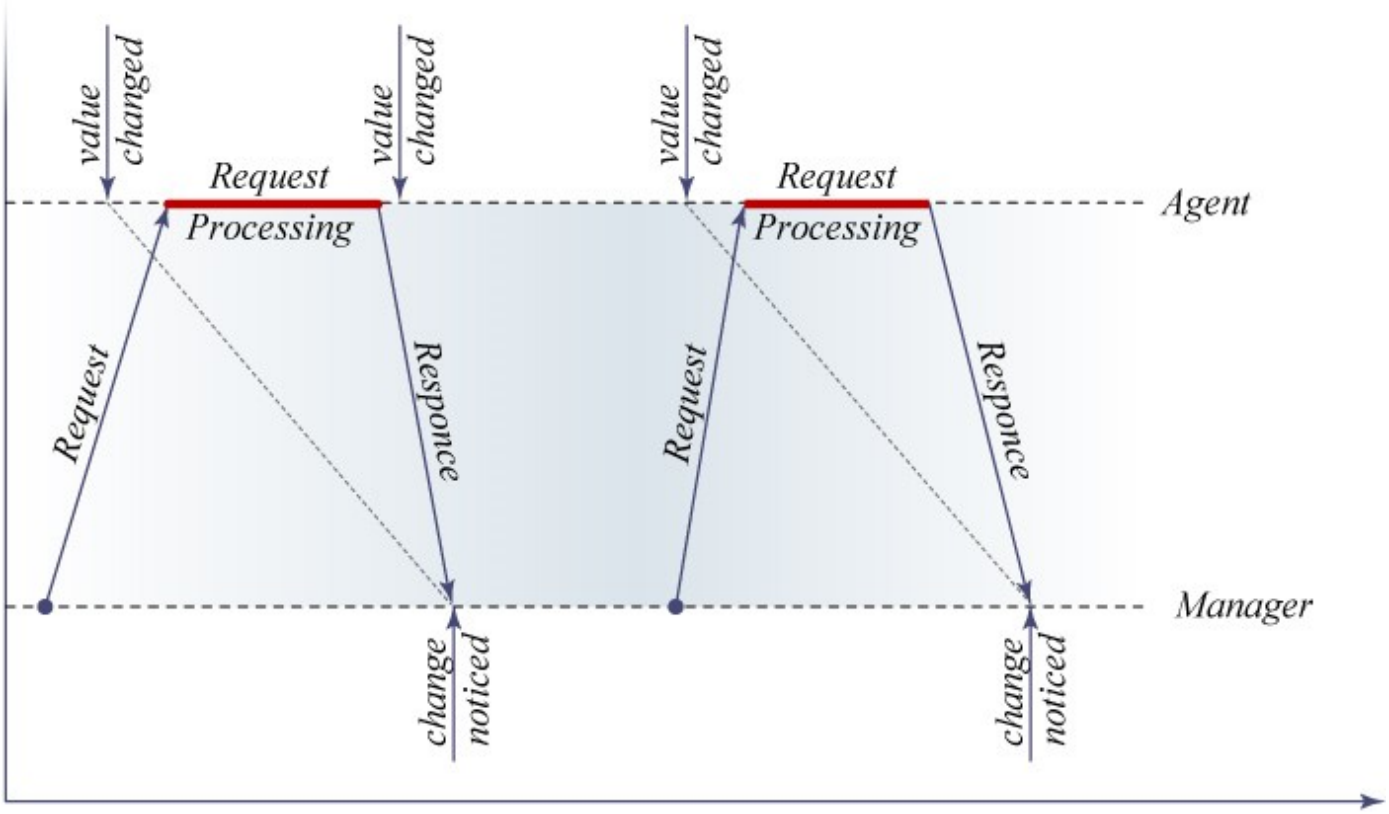
УПРАВЛЕНИЕ, ОСНОВАННОЕ НА ОПРОСЕ

В данном случае, опрос относится к диалогу менеджер-агент, который инициирован менеджером. Само взаимодействие основано на цикле запрос-ответ:

1. Менеджер делает запрос (чтобы получить часть информации управления или изменить параметры конфигурации, или заставить выполнить определенную операцию).

2. Агент получает запрос, обрабатывает его и отправляет обратно ответ, включая результат по запросу или индикатор/описание успеха/ошибки .

Когда начать диалог, решает менеджер: в определенные моменты с заранее установленными интервалами, или при определенных событиях (например, когда пользователь запрашивает обновление информации управления).



AtomMind Network Manager не инициирует опрос каждый раз, когда информация управления сетевыми элементами запрашивается операторами. Вместо этого, он кэширует[502] информацию и использует кэшированные данные по запросам, при необходимости обновляя их. Следовательно, информацию управления запрашивают (и опрос инициируется) при следующих обстоятельствах:

- когда приложение для управления изначально берет на себя управление устройством, впервые сохраняя информацию в кэш AtomMind Network Manager;
- когда информация в кэше считается устаревшей: например, если она слишком старая (относительно настроек управления устройства), или если было получено соответствующее уведомление, или если пользователь непосредственно запрашивает обновление кэша.

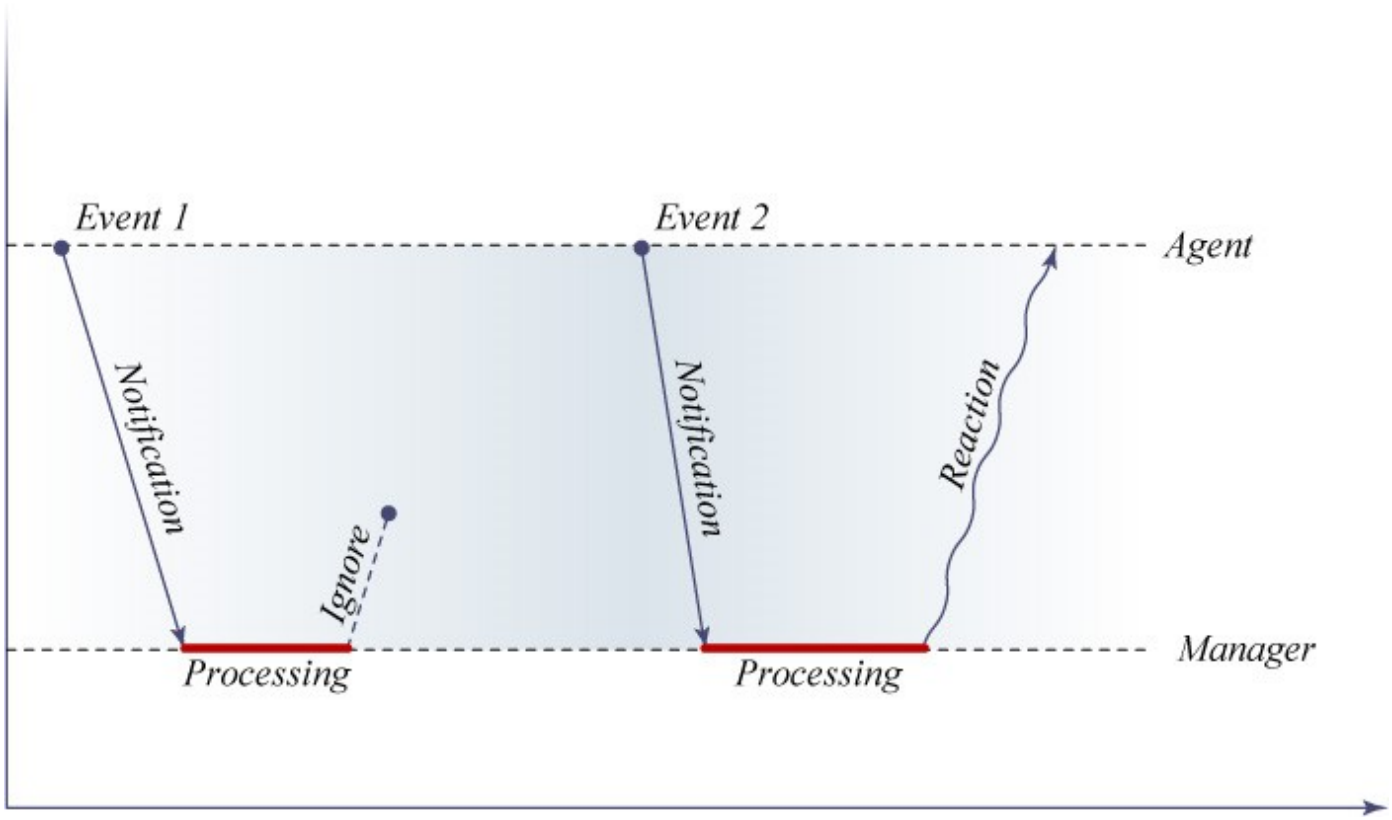
Существует определенная задержка между моментом, когда событие произошло на стороне агента (например изменение некоторого значения)и моментом, когда менеджер обнаружил его. Задержка зависит от частоты запросов опроса. С другой стороны, чрезмерно частые запросы нарушают режим работы системы сетевого управления. AtomMind Network Manager предоставляет средства корректировки частоты опросов для определенных частей информации управления согласно потребностям пользователя.

Грануляция запроса является другим, тебующим решения, вопросом в управлении основанном на опросе. AtomMind Network Manager поддерживает как объемные запросы, так и инкрементные операции, и автоматически использует соответствующий метод в различных ситуациях.

Кроме того, AtomMind Network Manager поддерживает сбор исторической информации, делая ее доступной для обработки и анализа.

УПРАВЛЕНИЕ, ОСНОВАННОЕ НА СОБЫТИИ

Связь, основанную на событии, во многом можно считать противоположной методу, основанному на опросе. Вопреки ответу, отправленному после запроса, агент самостоятельно определяет когда ему необходимо отправить сообщение о событии, без ожидания запроса. Агент инициирует связь и отправляет менеджеру сообщение, сообщая о наступлении события. Например, это может быть аварийным сигналом, который указывает, что бумага была зажата в принтере, или предупреждением о перегреве процессора на устройстве, или индикацией относительно подозрительного действие, если кто-то безуспешно пытался получить доступ к устройству несколько раз подряд и т.д. Агент должен быть настроен должным образом, чтобы отправить уведомление определенному менеджеру. Это можно сделать либо используя приложение для управления (так называемая подписка на события), либо используя внешние инструменты (для конкретного устройства) .

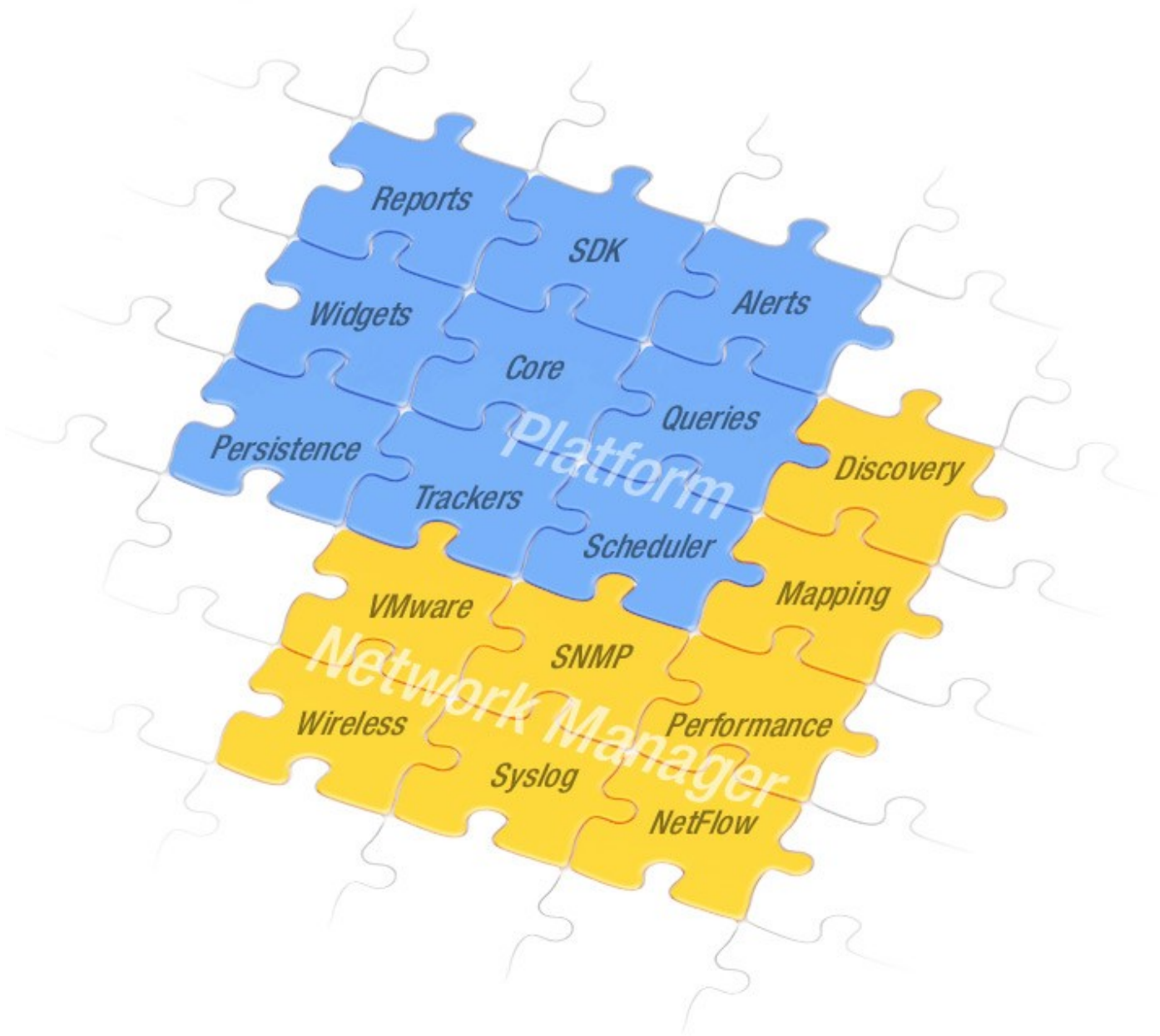


Менеджер самостоятельно решает, когда и как реагировать на полученные события. Он может среагировать немедленно с применением контролирующих действий, или с задержкой, или отправить некие уточнения запроса, или просто проигнорировать событие.

В AtomMind Network Manager управление основанное на событиии реализуется как, например, SNMP-ловушки, обработка Syslog-сообщений и NetFlow сбор данных.

19.1.2.2 Архитектура AtomMind Network Manager

Технически AtomMind Network Manager реализован как набор плагинов[207] и драйверов[518], расширяющих AtomMind.



Некоторые функции, которые предоставляет Network Manager, просто взяты из AtomMind. В то же время, AtomMind Network Manager включает в себя ряд плагинов и драйверов, которые расширяют основные функциональные возможности AtomMind и обеспечивают работу компонентов для управления и мониторинга сетевых устройств, сервисов и приложений. Данные компоненты описаны ниже.

Охватывая почти все стандартные требования для сетевого управления, AtomMind Network Manager остается высоконастраиваемой и расширяемой системой, унаследовав эти достоинства от AtomMind и ее архитектуры.

Network Host Device Driver

Network Host Device driver[593] используется в качестве основания для управления и мониторинга серверов, приложений, и сервисов, расположенных IP сети любого типа (проводное или беспроводное соединение, локальная или глобальная сеть и т.д.). Также, этот драйвер поддерживает Simple Network Management Protocol[1828] (SNMP).

Database Device Driver

Database Device Driver[647] создан для мониторинга серверов баз данных и выполнения произвольных SQL-запросов.

Network Management Plugin

Плагин сетевого управления предоставляет инструменты для конфигурации и мониторинга сетевого управления, такие как: тревоги 1956, запросы и отчеты 1956, графики 1051, виджеты 943, которые поддерживают мониторинг доступности 1853, производительности системы 1856 и производительности сети 1863 для IP хостов, SNMP-совместимых устройств, баз данных, принтеров, VMware ESX серверов, беспроводных устройств, различных web сервисов и приложений.

Плагин обнаружения

Плагин обнаружения обеспечивает поддержку для сетевого обнаружения 1811.

Плагин Syslog

Syslog плагин обеспечивает мониторинг и консолидацию *Syslog сообщений* 1883.

Плагин NetFlow

NetFlow плагин поддерживает мониторинг, разбор и анализ сетевого трафика используя протокол *NetFlow*.

Local File Driver

Local File driver обеспечивает мониторинг для файлов, расположенных на сервере AtomMind. См. раздел мониторинг локального файла 1881 для детализации.

Local Folder Driver

Local Folder driver обеспечивает мониторинг для папок, расположенных на сервере AtomMind. См. раздел мониторинг локальной папки 1882 для детализации.

19.1.2.3 Показатели

По сути мониторинг означает сбор информации от управляемых сетевых элементов и расчет эксплуатационных характеристик (**индикаторов**) для них. Удобно классифицировать логически соединенные индивидуальные индикаторы вместе в **показатели**. Таким образом, каждый показатель характеризует определенный аспект работы контролируемого элемента.

Например, показатель *Доступности сетевого хоста* может быть определен как: одно включенное устройство со статусом *online* (логическое значение, указывающее на достижимость хоста), *время отклика* (время с момента передачи и до приема) и индикаторы *уровня потери пакета*, доступные при использовании ping. Дополнительно, в показатели могут быть добавлены подробные данные *трассировки*.

Существует множество типов сетевых показателей, которые могут быть использованы для мониторинга сетей. AtomMind Network Manager предоставляет несколько наиболее часто используемых показателей. Более того, платформа AtomMind предлагает богатые возможности для расчета других показателей, если в этом есть необходимость.

Есть два основных метода расчета показателей для мониторинга поведения сетевого элемента:

- Мониторинг текущего значения: характеристики элемента получены в определенный момент времени.
- Статистический мониторинг: одна или несколько определенных отдельных характеристик накапливаются и используются для расчета агрегированного индикатора.

AtomMind поддерживает оба метода. Мониторинг может быть сконфигурирован для периодического опроса контролируемых объектов и расчета индивидуальных индикаторов, чтобы получить текущее значение и сохранить его. После, сохраненные данные могут быть агрегированы, чтобы получить статистическое представление об индикаторе.

С помощью пинга сетевого хоста, мы можем получить информацию о состоянии его готовности в определенный момент времени. Настройка устройства для периодического пинга с поддержкой истории синтезирует достаточное количество информации для расчета *статистической доступности,* т.е. период времени, когда устройство было в действующем состоянии. Это может быть как агрегированное значение для некого фиксированного временного интервала, так и доступность в течении всего периода мониторинга.

Показатели и индикаторы используются инструментами AtomMind Network Manager. Прежде всего, индикаторы доступности используются для представления состояния устройства и сервиса. Эти и другие индикаторы также используются в различных тревогах, отчетах, диаграммах, виджетах.

19.1.2.3.1 Показатели доступности и работоспособности

Доступность и работоспособность сетевого элемента помогают ответить на следующие вопросы:

- Если элемент полностью работает, т.е. он является *доступным*? Этот аспект отображен в показателе **Доступность.**
- Если элемент полностью функционален, т.е. он *работает как ожидалось*? Данный аспект охвачен показателем **Работоспособность.**

Показатель доступности характеризует основное состояние контролируемого элемента.

Показатель работоспособности описывает возможность контролируемого элемента выполнить его существенные функции.

В AtomMind Network Manager доступность проверяется при помощи простых стандартных процедур, таких как пинг сетевого устройства или соединение с сервисом через указанный порт. Задача состоит в том, чтобы гарантировать, что управляемый элемент в целом "жив".

Мониторинг работоспособности заключается в контрастной, более комплексной и "всесторонней" проверке, гарантирующей, что управляемый элемент не "только работает", а "работает должным образом". Так как значение термина "работает должным образом" отличается для различных контролируемых элементов, работоспособность, как правило, является более сложной и часто настраиваемой процедурой, как выполнение удаленного скрипта, получение и отправка email сообщений и т.п. В этом отношении, мониторинг работоспособности может быть охарактеризован как *умный мониторинг.* Процедуры проверки работоспособности часто дают не просто информацию о состоянии контролируемого элемента, но и более подробные данные от устройства.

Хотя различие между этими показателями может показаться незначительным, относительным или даже субъективным, все еще важно отличать данные показатели. Показатель работоспособности может быть рассмотрен как усовершенствованный показатель доступности. Например, доступность определена только для отдельных сервисов, а не для всего сетевого устройства в целом.

Процедура проверки доступности\работоспособности зависит от типа контролируемого элемента:

- Доступность **основного устройства IP сети** основана на ping 596 и, дополнительно, на трассировке 607 сервисов. Другими словами, доступность в этом случае определяется как *достижимость* IP хоста: устройство считается доступным, если сетевые пакеты могут быть доставлены устройству и от него.
- Если **устройство IP сети** предоставляет другие сервисы, его доступность рассчитывается как *конъюкция* (логическая *"AND"* операция) статусов доступности\работоспособности всех включенных сервисов. Поэтому, доступность основного устройства, упомянутая ранее, является лишь этапом более общей процедуры, описанной здесь.
- Доступность стандартных **сервисов IP хоста** фактически заменена проверкой работоспособности. По умолчанию, большинство сервисов настроены на простую проверку доступности, но, при необходимости, пользователь может конфигурировать их для всестороннего тестирования работоспособности. См.раздел и подразделы Network Host 593 device driver, где описаны определенные сервисы для получения более подробной информации.

  Например, SSH сервис 609 по умолчанию имеет пустой скрипт и поэтому выполняет только операции соединения, аутентификации и разъединения. Это можно считать основной процедурой проверки доступности.

  Пользователь может настроить сервис, задав скрипт. Выполнение скрипта предоставляет богатые возможности для усовершенствованного анализа контролируемого устройства, поэтому его можно отнести к проверке работоспособности.

  Обратите внимание, что проверка работоспособности включает в себя все этапы, что и выполнение основной процедуры проверки доступности. Следовательно, успешная проверка работоспособности автоматически означает, что сервис доступен.

- Доступность **ресурса** (ЦП, хранилище, процессы, сеть etc.) может быть проверена через SNMP 1828, WMI 1850 или JMX 1920.

  Например, рабочее состояние сетевого интерфейса может быть получено с помощью области `ifOperStatus` в `ifTable`. Кроме того, соответствующие SNMP-ловушки могут быть использованы для обнаружения изменения состояния.

- Проверка доступности/работоспособности некоторых **SNMP-совместимых устройств** может быть выполнена с помощью SNMP 1828.

См. разделы мониторинг принтера 1917, беспроводного устройства 1919, и/или VMWare для просмотра примеров.

- Мониторинг, включая мониторинг доступности и работоспособности **других типов устройств** может запрашивать для выполнения пользовательский драйвер устройства.

SQL database device driver 647 был введен для реляционных баз данных. Он обеспечивает мониторинг доступности и работоспособности для JDBC-совместимых систем управления базами данных.

AtomMind Network Manager обеспечивает расчет как **текущей доступности,** так и **статистической доступности.**

См. также раздел мониторинг доступности 1853.

19.1.2.3.2 Показатели производительности

Производительность - это еще один важный вопрос, требующий решения, для администраторов сети. Показатели производительности включают в себя: *индикаторы для сети в целом* (например трафик, задержки, ошибки), так же как и характеристики определенных сетевых элементов, обеспечивающих использование и доступность ЦП, хранилища (памяти и дисков), связи, ПО, виртуальных ресурсов, различных приложений\сервисов для индикации деятельности и т.д. См. разделы Сетевой мониторинг 1863 и Мониторинг производительности 1856 для получения более подробной информации о том, как показатели данного вида реализованы в AtomMind Network Manager.

19.1.2.4 Обработка данных

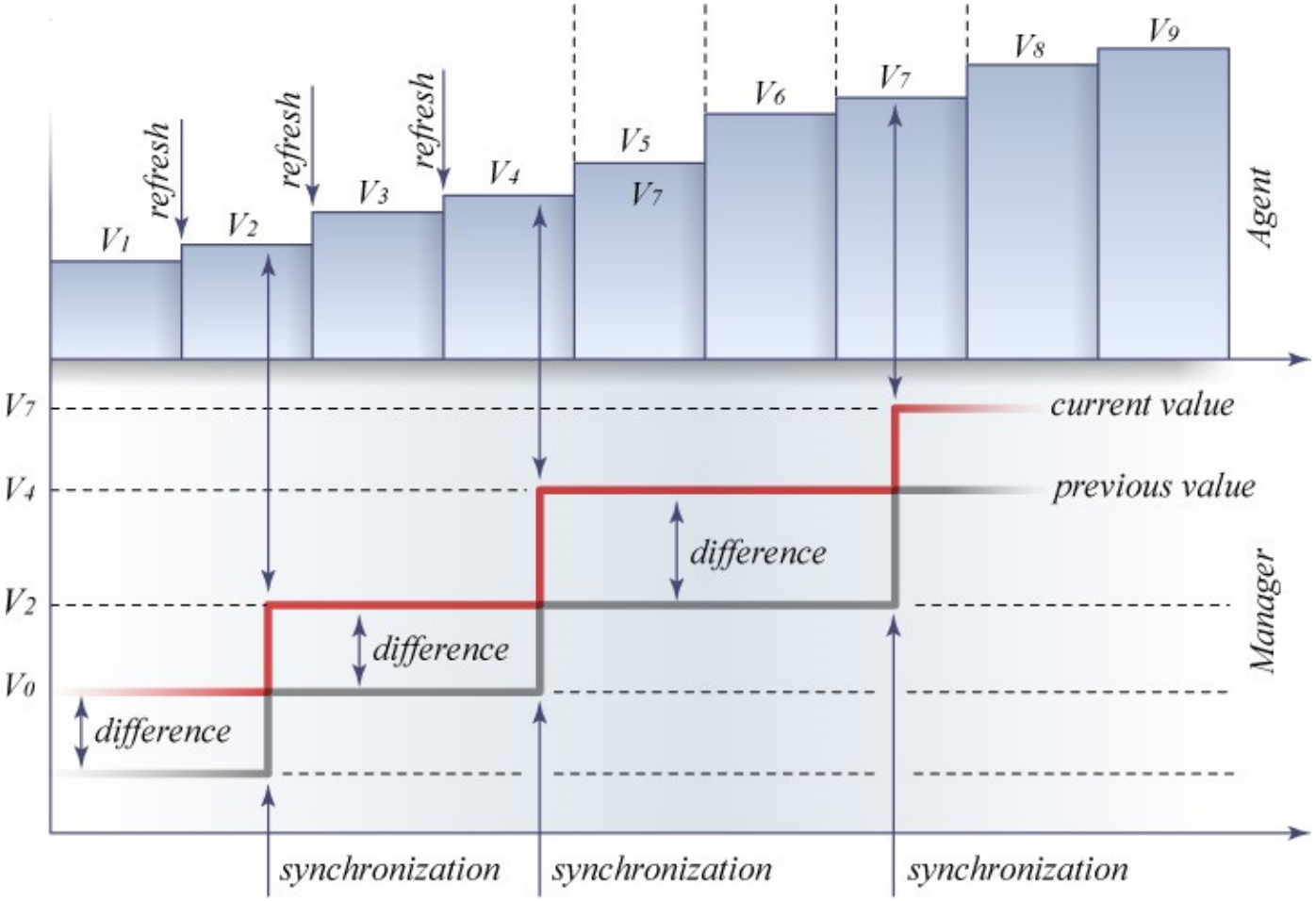
Функционально мониторинг является процессом сбора и анализа данных. Сырые данные, собранные от управляемых устройств, должны быть обработаны с целью извлечения количественных характеристик, которые мы можем проанализировать. Данный раздел содержит краткий обзор типов обработки данных, используемых в AtomMind Network Manager и связанных с ними вопросов.

Сырые данные

В некоторых случаях, количественные характеристики, которые вы хотите контролировать, могут быть непосредственно извлечены из устройства и использованы в задачах мониторинга "как есть" - без дополнительной обработки. В качестве примера можно упомянуть `hrProcessorLoad` SNMP переменную, которая указывает на "среднее значении за последнюю минуту в процентах от времени, когда процессор не находился в простое" [RFC1514 - Host Resources MIB]. Это значение характеризует текущее использование ЦП и может быть использовано "как есть", например, в диаграммах или тревогах.

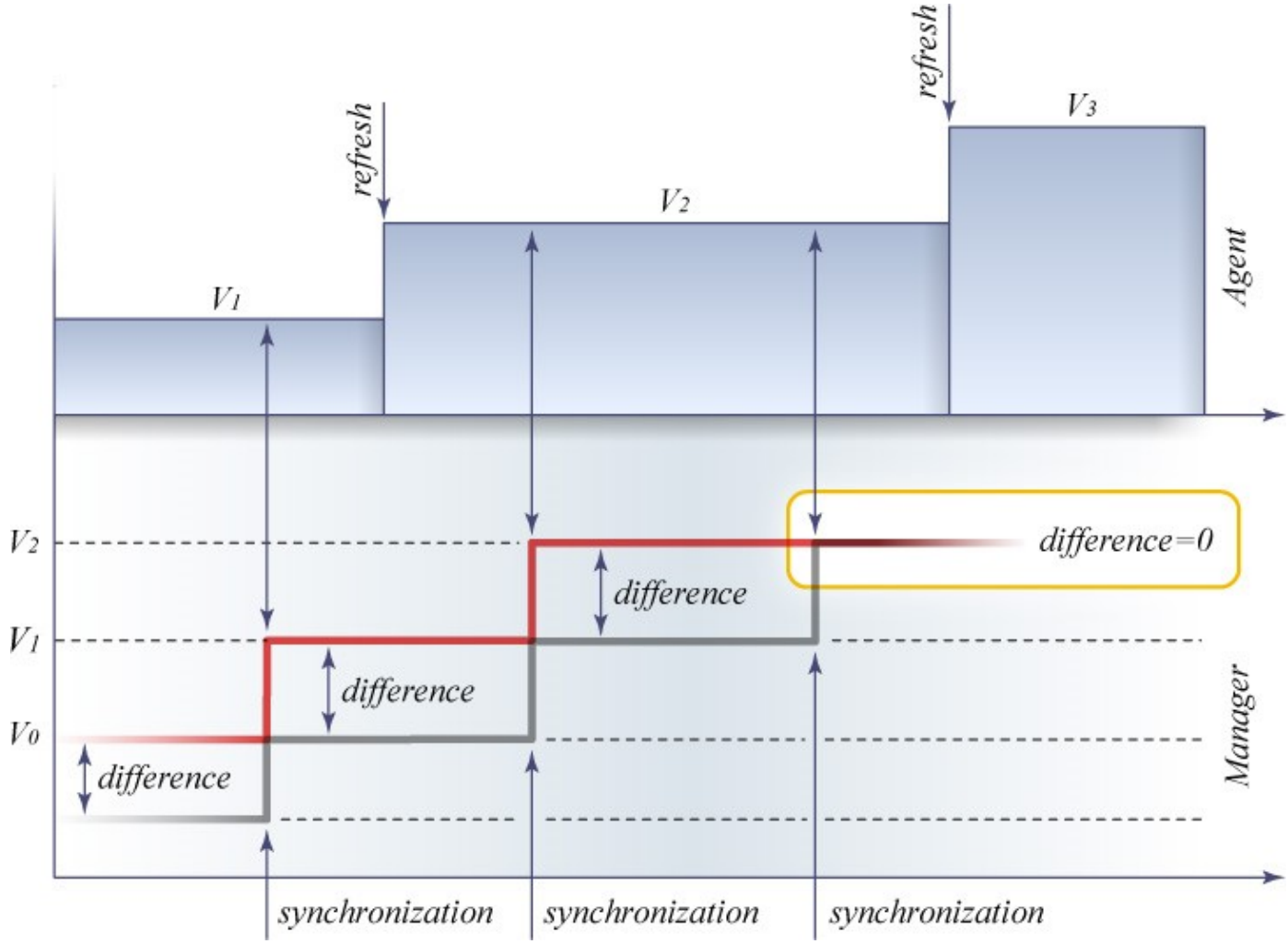
Дифференцирование

Но не все значения, которые мы хотим отследить, могут быть выражены в виде сырых данных 1799. Чаще всего, собранные данные должны быть определенным образом обработаны, чтобы стать полезными для мониторинга. *Дифференцирование* - одна из наиболее распространенных операций по обработке данных. Имеется ввиду расчет разницы между текущей и предыдущей выборками.



Разница может быть полезна в нескольких определенных случаях, но одним из важнейших и часто используемым применением этого показателя является *дифференцирование по времени*[180], описанное ниже.

Здесь должен быть рассмотрен один важный момент. Значения, которые мы используем для мониторинга, с определенным интервалом периодически обновляются со стороны устройства. AtomMind Network Manager считывает эти значения в течении стандартной синхронизации[514] в различные моменты времени (относительно собственного периода синхронизации[504]). Поэтому, обеспечение слишком маленького периода синхронизации (меньше, чем интервал обновления со стороны устройства) может дать соответствующий результат: последовательные выборки будут равны (т.к устройство не имело достаточно времени для обновления значений) и разница будет равна нулю.



Период синхронизации SNMP переменных, используемый в различных показателях, не должен быть меньше, чем интервал обновления со стороны устройства. Обратитесь к специальной документации устройства, чтобы узнать насколько часто устройство обновляет свои индикаторы и как это настроить. Кроме того, вы можете определить это опытным путем, используя инструменты AtomMind. Зная интервал обновления со стороны устройства, вы можете соответствующим образом скорректировать период синхронизации 504 (со стороны AtomMind сервера).

Деривация

Деривация является хорошим и наиболее распространенным примером дифференцирования 1799 "в действии".

Некоторые из значений, мониторинг которых проводится системой, фактически являются производными от значений, к которым AtomMind имеет доступ. Например, универсальные SNMP устройства обеспечивают только общее количество байтов, полученных на интерфейс. Таким образом, чтобы получить входной трафик интерфейса, который является (в среднем) числом байт, полученных в единицу времени, мы должны оценить изменения всех полученных байтов за все время, т.е. применить *дифференцирование по времени.* Такие показатели мы будем называть **производная по времени**.

Простейший способ расчета производной по времени состоит в том, чтобы получить текущие и предыдущие значения и вычислить соотношение между их разницей и промежутком времени. Поэтому, их реализация основана на дифференциальных показателях и сопровождается тем же вопросом 1800: производные значения могут быть равны нулю, когда период синхронизации слишком мал.

19.1.2.5 Протоколы и технологии

Данный раздел содержит краткий список протоколов и технологий, поддерживаемых и используемых в AtomMind Network Manager:

Основные протоколы	Internet Protocol (**IP**)
	Transmission Control Protocol (**TCP**)

	User Datagram Protocol (**UDP**)
Протоколы, стандарты и технологии сетевого управления	Simple Network Management Protocol (**SNMP**)
	Windows Management Instrumentation (**WMI**)
	Internet Control Message Protocol (**ICMP**)
	Syslog
	NetFlow
	Java Management Extensions (**JMX**)
	Intelligent Platform Management Interface (**IPMI**)
E-mail протоколы	Post Office Protocol (**POP**)
	Internet Message Access Protocol (**IMAP**)
	Simple Mail Transfer Protocol (**SMTP**)
Другие протоколы приложения	File Transfer Protocol (**FTP**)
	Hypertext Transfer Protocol (**HTTP**), Hypertext Transfer Protocol Secure (**HTTPS**)
	Secure Shell (**SSH**), Secure Copy Protocol (**SCP**)
	Lightweight Directory Access Protocol (**LDAP**)
	Domain Name System (**DNS**) protocol
	Dynamic Host Configuration Protocol (**DHCP**)
	Remote Authentication Dial In User Service (**RADIUS**)
	Simple Object Access Protocol (**SOAP**)
Технологии баз данных	Java Database Connectivity (**JDBC**), Open Database Connectivity (**ODBC**)

19.1.2.6 Инструменты сетевого управления

Network Manager использует оба стандартных инструмента обработки данных AtomMind и специальные компоненты, ориентированные на сетевое управление, включая:

- Плагины и драйверы 1796
- Тревоги 1956
- Диаграммы 1051
- Отчеты и запросы 1956
- Виджеты 943
- Инструментальные панели.

Эти инструменты и компоненты покрывают наиболее часто встречающиеся задачи сетевого управления. Многие специфические проблемы и задачи могут встретиться при настройке/изменении этих элементов или при создании пользовательских тревог, диаграмм, отчетов, запросов.

Все инструменты могут быть настроены или использованы в качестве шаблонов для создания новых инструментов.

19.1.2.7 Совместимость с IPv6

AtomMind Network Manager создан и утвержден для работы в IPv6 сетях. Поддержка IPv6 комплексная, почти все компоненты системы, позволяющие вводить обычный адрес IPv4, также принимают спецификацию имени хоста (имени DNS) или адреса IPv6.

Сегменты IPv6 адреса должны разделяться двоеточием, т.е. `2001:0db8:11a3:09d7:1f34:8a2e:07a0:765d`.

Если IPv6 адрес определен в настройках устройства хоста сети 593, AtomMind Network Manager будет использовать протокол ICMPv6 для его пингования 596.

19.1.3 Настройка AtomMind Network Manager

Данная глава посвящена конфигурации Вашей системы сетевого управления. Глава включает в себя настройку обеих сторон, принимающих участие в управлении и мониторинге: **управляемых элементов** (агентов), которые должны предоставлять информацию и контролировать операции, и **управляющих компонентов**, которые должны быть настроены.

Дополнительную информацию о настройке сетевой среды для мониторинга см в разделе Настройка среды 1809.

Компоненты AtomMind Network Manager описаны в двух разделах о Параметрах глобальной системы 1803 и Конфигурации управляемых устройств/сервисов 1805 и инструментальных средств 1808.

19.1.3.1 Настройка драйверов и плагинов

В этом разделе говорится об **опциях глобальной настройки** сетевых драйверов и плагинов, связанных с управлением. Эти опции используются для настройки поведения AtomMind Network Manager на глобальном уровне, в отличие от конфигураций отдельно управляемых устройств/сервисов 1805 и инструментальных средств 1808 .

Дополнительную информацию о конфигурации см в следующих разделах:

- Драйвер хоста сети 1803
- Плагин обнаружения 1804
- Плагин Syslog 1804
- Плагин NetFlow 1805

19.1.3.1.1 Настройка SNMP

В этом разделе говорится о глобальных настройках драйвера устройства SNMP. Большинство этих параметров многократно использует сервис SNMP сетевого устройства.

Директория MIB-файлов

Файлы MIB 644 описывают данные SNMP, которые может использовать AtomMind Network Manager. AtomMind Network Manager использует более чем две сотни файлов MIB, которые важны для мониторинга по SNMP, и, таким образом, поставляются в комплекте с дистрибутивом AtomMind Server. Вы можете модифицировать этот набор согласно своим потребностям, добавляя MIB-файлы для используемых Вами устройств/сервисов, удаляя неиспользуемые файлы с целью сбережения памяти и улучшения производительности сервера, обновляя старые версии до более новых и пр.

Мониторинг ловушек SNMP

Опции мониторинга ловушек SNMP 643 определяют, должен ли и каким образом AtomMind Network Manager прослушивать SNMP-извещения 1846. Убедитесь, что конфигурация Вашего AtomMind Network Manager согласуется с конфигурацией Ваших SNMP-устройств.

Автоматическое обнаружение источников SNMP-извещений

Автоматическое обнаружение ловушек SNMP 644 активизирует AtomMind Network Manager на автоматический запуск обнаружения 1811 объектов в сети, которые отправляют серверу SNMP-извещения. См. также раздел Автоматическое обнаружение 1820.

Дополнительная информация

Информацию о параметрах для драйвера SNMP см в разделе Глобальные параметы SNMP 644.

19.1.3.1.2 Настройка обнаружения

Обнаружение 1811 - это автоматическое или полуавтоматическое действие по обнаружению новых сетевых устройств, выявлению запущенных на них сервисов и приложений, добавлению и настройке обнаруженных объектов для мониторинга и управления. Обнаружение реализовано плагином обнаружения 1797.

Параметры обнаружения 1820 определяют опции для интерактивного 1819 обнаружения и обнаружения без заголовка 1819, обнаружение одного устройств 1819 и автоматического обнаружения 1820. Дополнительную информацию см. в разделе Установка и выбор сервиса 1812.

Настройка **Плагина Обнаружения** доступна в узле **Обнаружение Устройства** под элементом **Драйверы/Плагины** в **Системном дереве**. Следует перезапустить AtomMind Server, чтобы изменения вступили в силу.

19.1.3.1.3 Настройка Syslog

Поддержку Syslog 1883 реализует плагин Syslog 1797. Он настраивается через узел **Syslog** элемента **Драйверы/Плагины** в **Системном дереве**.

Настройка Syslog включает **Настройку Сервера Syslog** и параметры **Автоматического обнаружения источников сообщений**, которые описаны ниже. Обратите внимание, что следует перезапустить AtomMind Server, чтобы изменения вступили в силу.

Настройка мониторинга Syslog

Эта таблица включает параметры Консолидации и мониторинга сообщений Syslog 1883.

Свойство	Описание
Включено	Включает/отключает мониторинг сообщений syslog.
UDP	Включает/отключает использование UDP для получения сообщений Syslog.
TCP	Включает/отключает использование TCP для получения сообщений Syslog.
Порт Syslog	Определяет порт для прослушивания сообщений Syslog.
Таблица пересчета критичности	Уровень критичности для отражения уровня критичности AtomMindl 73. См. ниже.

ПЕРЕСЧЕТ УРОВНЯ КРИТИЧНОСТИ

Таблица Пересчета Критичности используется для подсчета уровня генерируемого события 73 %ag%>, основанного на исходном уровне критичности сообщений Syslog. Дополнительную информацию о генерировании событий Syslog см. в разделе Консолидация и мониторинг событий Syslog 1883.

Таблица преобразовывает каждое из значений уровня критичности Syslog (как определено в RFC 5424) в соответствующий уровень критичности AtomMind (см. раздел Уровни события 73). Если значение критичности Syslog отсутствует в таблице, полученный в итоге уровень события AtomMind оказывается равным 0.

По умолчанию таблица задана следующим образом:

Уровень критичности Syslog			**Критичность события AtomMind**	
Код	**Имя**	**Описание**	**Код**	**Уровень**
7	Debug	Сообщение уровня отладки	1	Уведомление
6	Informational	Информационное сообщение	2	Инфо
5	Notice	Нормальное, но важное состояние	2	Инфо
4	Warning	Состояния предупреждения	3	Предупреждение
3	Error	Состояния ошибки	4	Ошибка
2	Critical	Критические состояния	4	Ошибка

1	Alert	Следует немедленно принять меры	4	Ошибка
0	Emergency	Система непригодна	5	Неисправимо

Эту таблицу можно изменить с учетом необходимых для преобразования процедур.

Автоматическое обнаружение источников сообщений Syslog

Этот флажок включает или отключает автоматическое обнаружение 1885 хостов, отправляющих сообщения Syslog серверу AtomMind.

19.1.3.1.4 Настройка NetFlow

Поддержка NetFlow, которую предоставляет Плагин NetFlow 1797 позволяет AtomMind Network Manager собирать данные для сбора данных о сетевом трафике.

Настройка NetFlow определяет два аспекта: как данные собираются, и как собранные данные могут быть представлены пользователям.

Параметры сбора NetFlow определяют параметры связи, используемые AtomMind Network Manager для получения данных о потоках из источников.

Таблица Представлений NetFlow в настройках описывает виджеты, используемые для представления собранных данных NetFlow для анализа пользователями. Дополнительную информацию можно найти в разделе Табличные виджеты NetFlow.

19.1.3.2 Настройка/администрирование устройств и сервисов

Этот раздел содержит информацию и некоторые советы об установке и поддержке инфраструктуры управления/мониторинга путем настройки начальной конфигурации (как ручной 1805 так и автоматической 1807) и дальнейшем администрировании 1807.

19.1.3.2.1 Ручное добавление и настройка устройств и сервисов

Ручную настройку можно охарактеризовать как наиболее точный способ добавить и установить устройство. В то же время эта процедура может оказаться утомительной задачей, если необходимо настроить много устройств. В этом случае сетевое обнаружение может стать лучшей альтернативой.

Чтобы добавить новое устройство в систему, выполните следующее:

- вызовите действие Добавить Новое Устройство 1492 из контекста Device
- определите уникальное **Имя Устройства**, **Описание Устройства** и **Драйвер Устройства**
- щелкните OK, чтобы зарегистрировать устройство.

Дополнительную информацию см. в главе Устройства 497.

Только что добавленное устройство следует правильно настроить. Это подразумевает использование таких обычных параметров как **Адрес Хоста**, **Основные свойства Устройства** и **Опции Синхронизации Параметров Устройства** вместе со специфичными опциями, которые зависят от типа устройства (используемого драйвера), сервисов и приложений, которое оно запускает, а также роль, которую он выполняет.

Общие параметры настройки

АДРЕС ХОСТА

IP-адрес или имя хоста необходимы для большинства типов устройств и просто определяет адрес хоста для сетевого устройства.

ТИПОВЫЕ СВОЙСТВА УСТРОЙСТВА

Типовые свойства устройства определяют атрибуты стандартного устройства, такие как имя, описание, временную зону и пр. Дополнительную информацию об устройствах можно найти в разделе Типовые свойства устройства 510. Как правило, большинство из этих опций не следует изменять, однако следует убедиться, что они правильные. Обратите внимание на следующие параметры:

- **Тип устройства**. Этот параметр определяет, какую роль устройство играет в сетевой среде. Эта роль, например, может быть использована для более удобного обнаружения устройств путем организации их по типовым группам.

- Параметры синхронизации, а именно **Период Синхронизации**, **Прерывание Синхронизации и Повторное подключение при Обнаружении Ошибки**, **Включение Расширенного Статуса**, **Выражение Зависимости Устройства** определяют, каким образом информация об устройстве будет обновляться. Вместе с Опциями Синхронизации Параметров Устройства 502 эти параметры важны для правильного функционирования операции мониторинга.
- Операция **Приостановить устройство** позволяет остановить предварительный опрос устройства.

ОПЦИИ СИНХРОНИЗАЦИИ ПАРАМЕТРОВ УСТРОЙСТВА

Опции Синхронизации Параметров Устройства 502 позволяет модифицировать опции синхронизации (опроса) для каждого параметра устройства. Это наделяет пользователя многоуровневым средством управления процессом мониторинга.

Параметры, специфичные для драйвера

Настройка специфичных типов устройств описана в следующих подразделах:

- Хост сети 1806
- Локальный файл 1807
- Локальная папка 1807
- База Данных SQL 1807
- JMX 1807.

Параметры мониторинга

Параметры мониторинга распределены по различным группам параметров:

- Параметры синхронизации 1806, как то: **Период Синхронизации**, **Прерывание Синхронизации и Повторное Подключение при Обнаружении Ошибки**, **Включение расширенного статуса**, **Выражение Зависимости Устройства** в Основных Свойствах Устройства.
- Опции Синхронизации Параметров Устройства 1806.

19.1.3.2.1.1 Network Host Device Configuration

Драйвер сетевого хоста обеспечивает поддержку для ряда сервисов, которые могут быть на нем запущены. См. главу Сетевой хост 593 и содержащиеся в ней разделы, если Вас интересует подробное описание драйвера, сервисов и их параметров конфигурации. В этом разделе мы обсудим некоторые важные сервисы и специфику их настройки, на которые следует обратить внимание.

Ping

Сервис ping 596 важен для мониторинга доступности сетевого хоста. Параметры настройки по умолчанию подойдут в большинстве случаев. Однако для некоторых специфичных ситуаций необходимо установить timeout и, возможно, некоторые другие параметры.

SNMP

SNMP-мониторинг - это самый универсальный и широко используемый способ собрать подробную информацию об отдельных устройствах, сервисах и приложениях в Вашей сети.

Для отдельного, совместимого с SNMP устройства следует задать параметры опроса. Они описаны в разделе Параметры SNMP-опроса 640.

ОСНОВНАЯ НАСТРОЙКА

Сетевые устройства часто настраивают администраторы так, чтобы они отвечали требованиям безопасности и другим текущим требованиям. Т.о. следует убедиться, что **транспортный протокол**, **порт**, **версия SNMP**, **строки доступа для чтения/записи**, **параметры безопасности** и **размер PDU** на стороне AtomMind соответствуют настройке агента.

Кроме того, следует проверить, что параметры связи (**повторение команды** и **timeout**) соответствуют состоянию сети.

ПАРАМЕТРЫ ОПРОСА

Параметры опроса определяют:

- Какие данные AtomMind Network Manager будет читать с устройства
- Период опроса
- Правила хранения истории значений

Данные для чтения определяет параметр Данные для обработки 641. Существуют две опции, позволяющие найти баланс между полнотой данных и потерей производительности Вашей системы и сети. Если Вы используете **Все данные, обнаруженные опцией полного обхода SNMP-дерева** устройства**,** Вы получите все данные этого устройства, который оно предоставляет по SNMP. Это может оказаться полезным, особенно, если Вы хотите изучить новый тип устройства, обнаружить все данные, которые оно экспортирует по SNMP, и добавить необходимые MIB-файлы в систему. Но в большинстве случаев, это лишь "загрязнит" систему избыточной информацией, займет дополнительную память и другие ресурсы, что приведет к снижению эффективности работы . Т.о., **все доступные значения MIB-директории** - наилучший выбор в большинстве из случаев.

Еще один важный аспект настройки опроса - это **опции синхронизации**. Они определяют, как часто AtomMind Network Manager опрашивает устройство и отдельные SNMP-переменные устройства. Вам вновь придется найти ту "золотую середину" между производительностью и точностью информации. Используйте опции синхронизации для всего устройства (часть Общие свойства 510) и для отдельных SNMP-переменых 502 для настройки опроса согласно Вашим требованиям.

Другие сервисы

Дополнительную информацию об операциях и настройке отдельных сервисов см. в разделе Сетевой хост 593.

19.1.3.2.1.2 Настройка локального файла

Драйвер локального файла 576 предоставляет мониторинг для файлов 1881, расположенных на сервере AtomMind. Настройка проста: следует лишь определить путь к файлу, который следует отследить, а также операции, которые должны быть выполнены с использованием свойств драйвера 576.

19.1.3.2.1.3 Настройка локальной папки

Драйвер локальной папки 577 предоставляет мониторинг для папок 1882, которые располагаются на сервере AtomMind. Чтобы настроить этот тип устройств, следует задать путь к отслеживаемой папке, включить или отключить операцию чтения и определить способ мониторинга (если активизировано), используя свойства драйвера 577.

19.1.3.2.1.4 Настройка базы данных SQL

Драйвер устройства Базы Данных 647 [****] 647 SQL 647 [****] 647 предоставляет мониторинг реляционных баз данных. Чтобы настроить этот тип устройств, следует правильно **настроить драйвера** для системы БД, которую следует отслеживать, параметры связи, включая **URL БД**, **имя пользователя** и **пароль**. Если требуется выполнить глубокий мониторинг, можно определить один или несколько **запросов**, которые должны быть выполнены в БД. См. также Свойства настройки устройства базы данных SQL 648.

19.1.3.2.1.5 Настройка JMX

Драйвер устройства JMX предназначен для мониторинга Java-приложений. Дополнительную информацию о настройке мониторинга JMX см. в разделе драйвер устройства JMX 571.

19.1.3.2.2 Обнаружение устройств и сервисов

Сетевое обнаружение 1811 - это действие, которое позволяет автоматически или полуавтоматически обнаружить сетевые устройства, создать учетные записи устройств и настроить мониторинг сервиса. Это позволяет установить инфраструктуру мониторинга для обширных сетей, когда операции, выполняемые вручную 1805, становятся слишком трудоемкими.

Дополнительная информация и руководство содержится в главе Сетевое обнаружение 1811.

19.1.3.2.3 Администрирование устройств и сервисов

AtomMind Network Manager предоставляет средства для администрирования инфраструктуры управления/мониторинга. Это может привлечь управляемые устройства к:

- изменению их общих свойств 1805: переименованию, изменению описания, типа, исправления адреса хоста, приостановке и пр.

- изменению параметров мониторинга устройства 1806
- изменению параметров устройства 1806
- группированию 751 и перегруппированию устройств.

19.1.3.3 Средства администрирования AtomMind Network Manager

AtomMind предлагает различные виды инструментальных средств, помогающих осуществлять мониторинг и управление сетью: тревоги, виджеты, таблицы, запросы и отчеты, информационные панели и пр.

Некоторые из данных инструментальных средств уже готовы к применению, другие следует сначала настроить. Примеры предварительно настроенных иснструментальных средств:

- Тревоги **Устройство в режиме offline**, **Высокая степень потери пакетов**;
- Графики **Время ответа на пинг**, **Потеря пакетов**;
- Инструментальные панели **Обзор хоста сети**, **Трафик сети**;
- Различные отчеты и запросы.

Некоторые другие инструментальные средства должны быть параметризованы конкретными значениями - лишь тогда их можно использовать. Многие из них можно реализовать и настроить для определенного устройства при помощи действия Установить профиль мониторинга 1808. Другой пример - табличные виджеты NetFlow, которые строятся из представлений NetFlow.

AtomMind Network Manager -- это расширяемая система. Она предоставляет широкие возможности для изменения существующих инструментальных средств и создания новых с тем, чтобы система отвечала Вашим собственным целям и потребностям. Используйте предлагаемые инструментальные средства как основу или примеры для создания собственного инструментария, позволяющего эффективнее справляться с Вашими задачами.

19.1.3.4 Действие Настройка профиля мониторинга

Действие 87 **Настройка профиля мониторинга** предоставляет пользовательский интерфейс для добавления и настройки инструментальных средств мониторинга сети, включая тревоги 1956, графики 1051 и датчики 2181.

Все инструментальные средства, доступные для устройства, классифицируются на несколько групп согласно типу мониторинга, который они осуществляют:

- Доступность устройства (см. Доступность 1853)
- Процессы (см. Мониторинг процессов 1860)
- CPU (Мониторинг производительности CPU 1856)
- Сетевые интерфейсы (см. Мониторинг ширины канала и трафика сети 1863)
- Запоминающие устройства (см. Мониторинг хранилищ 1857)
- VMware (см. Мониторинг Сервера VMware)
- Apache (см. Мониторинг веб-сервера Apache 1891)
- Принтер (см. Мониторинг принтера 1917)
- Производительность (для отслеживания различных сервисов)
- E-mail (для отслеживания E-mail 1901 серверов)
- HTTP (для отслеживания Веб-серверов 1889)
- FTP (для отслеживания FTP 1901 серверов)
- DNS (для отслеживания DNS 1903 серверов)
- LDAP (для отслеживания AD 1900 серверов)
- SSH (для отслеживания SSH-серверов 1903)

- Отслеживание сетевых путей (для отслеживания сервиса Traceroute 607)
- Типичные сервисы TCP и UDP (для отслеживания сервисов TCP 1904 и UDP 1904)

Доступность групп мониторинга для того или иного устройства зависит от типа устройства и данных, которые оно предоставляет.

Например:

- Инструментальные средства *Мониторинг принтера* поддерживаются лишь в том случае, если у устройства есть переменная SNMP `prtGeneralTable`;
- Присутствие группы *VMware* зависит от переменной `vmTable`;
- Инструментальные средства из группы *Доступность устройства* можно использовать для любого сетевого устройства, имеющего IP.

Чтобы сгенерировать инструментальные средства мониторинга для устройства, следует:

1. Выбрать элемент **Настройка профиля мониторинга** с контекстном меню устройства.
2. Выбрать группы инструментальных средств для создания.
3. Появится отдельное окно настройки для каждой отдельной опции. Добавьте и настройте одно или несколько инструментальных средств для этой группы.

19.1.3.5 Действие Настройка мониторинга

Действие 87 **Настройка мониторинга** предоставляет пользовательский интерфейс для добавления и конфигурирования средств мониторинга сети и соединения их с устройством. Устройство может быть выбрано из существующих, либо создано новое.

Действие устанавливается в Контексте сетевого управления 1957.

Вызов действия приводит к запуску помощника, который проводит пользователя через весь процесс, включающий следующие шаги:

1. Выбор средств мониторинга (определение "предмета мониторинга"). Помощник открывает список средств мониторинга, что позволяет пользователю выбрать несколько из них для использования при данном усвловии. Список содержит средства из категории "Сетевой мониторинг".
2. Создание необходимых ресурсов. Если для одного или нескольких выбранных средств ресурсы еще не были созданы, их предложат создать на этом этапе. Пользователь может отказаться от создания одного или нескольких ресурсов на текущий момент.
3. Определение устройства для предоставления данных для мониторинга (определение "объекта мониторинга"). Возможно выбрать устройство из существующих, либо создать новое. Помощник позволяет пользователю выбрать желаемую опцию. Если пользователь хочет выбрать устройство из существующих, отобразится список устройств. В противном случае, помощник открывает диалоговое окно для добавления устройства, в котором пользователь может указать драйвер и все необходимые параметры для нового устройства. В этом случае будет добавлена и синхронизирована учетная запись для устройства.

4. Наконец, создаются средства мониторинга и "прикрепляются" к определенному устройству.

19.1.3.6 Настройка среды

Этот раздел посвящен вопросам настройки управляемых сетевых объектов для связи с AtomMind Network Manager. В этом руководстве представлены следующие темы:

- Настройка устройств SNMP 1810
- Настройка устройств WMI 1810
- Настройка Windows для обеспечения мониторинга журналирования событий Windows 1810 с использованием SNMP-ловушек
- Настройка удаленного доступа к WMI 665
- Вопросы настройки TCP/IP 1811 для ОС Windows
- Включение ICMP для мониторинга доступности, используя Пинг и Трассировку.

19.1.3.6.1 Настройка устройства SNMP

Для обеспечения обмена данных устройства, совместимые с протоколом SNMP, должны быть настроены должным образом. В разделе Настройка агентов SNMP 1832 содержится инструкция по настройке Windows, Linux, система Solaris, устройств Cisco, Microsoft SQL, серверов Lotus Domino и Oracle.

19.1.3.6.2 Настройка устройств WMI

Чтобы использовать WMI для управления удаленным компьютером с ОС Windows, следует настроить на нем сервис WMI. См. Настройка удаленного доступа к WMI 665.

19.1.3.6.3 Получение событий журнала Windows

Журнал Событий Windows предназначен для предоставления детальной информации о функционировании ОС Windows и приложениях, запускаемых под ней.

Сообщения журнала событий Windows следует конвертировать в SNMP-ловушки для получения AtomMind Network Manager. Информацию о получении и обработке ловушек см. в разделе ловушки SNMP 1846.

Этот раздел содержит инструкции по настройке ПК с Windows для отправки SNMP-ловушек, когда добавляются определенные сообщения в Журнале Событий Windows.

Следует правильно установить и настроить SNMP-агент, чтобы отправлять ловушки в хоста AtomMind, на котором запущен сервер. См. Установка SNMP на системах Windows 1833 (и в частности, Этапы настройки SNMP-ловушек 1835).

Определение и экспортирование отображений событий

Утилита Майкрософт `evntwin` используется для выборки событий и подготовки необходимых данных. Она предоставляет графический интерфейс для выбора, настройки и экспортирования сообщений журнала событий для перевода их в SNMP-ловушки:

1. Запустите утилиту `evntwin` из меню, выбрав элемент **Запустить** или из командного окна.
2. Появится окно **Переводчик События в Ловушку**.
3. Выберите опцию **Пользовательская настройка** под группой **Тип конфигурации**.
4. Кликните мышкой **Редактировать**. Должен появится список **Источники событий**.
5. Выберите события, которые необходимо перевести в ловушки (кнопку **Поиск** можно использовать для поиска в списке) и кликните **Добавить**.
6. Должно появится окно **Свойства**. Условия создания ловушек можно модифицировать здесь.
7. Повторите этапы 5-6 для каждого события, которое Вас интересует (можно также выбрать несколько событий для добавления, используя множественный выбор Ctrl или Shift)
8. Кликните кнопку **Применить**.
9. Выделите все элементы в **Событиях, которые следует перевести с список ловушек** и нажмите кнопку **Экспортировать.** Выберите месторасположение и имя файла, чтобы сохранить определения преобразований событий в ловушки.

Экспортированный файл с преобразованиями должен быть текстовым файлом с одной или более строками следующего формата:

```
#pragma add <LogName> "<SourceName>" <EventID> <EventCount> <TimeInterval>
```

Настройка SNMP-ловушек

Утилита Майкрософт `evntcmd` используется для настройки перевода событий в ловушки, основанные на информации из конфигурационного файла. Чтобы это выполнить, следует запустить `evntcmd`, присвоив ему имя экспортируемого файла изменений.

Если на ПК с Windows используется Управление Доступом Пользователя (UAC), следует запустить интерпретатор команд (`cmd`) в режиме **Администратора** для того, чтобы получить доступ к утилите `evntcmd`.

19.1.3.6.4 TCP/IP Limitations of Desktop Versions of Windows

Все версии системы Microsoft Windows для рабочих станций, начинающиеся с Windows XP с пакетом обновлений 2 (SP2), ограничивают количество полуоткрытых подключений (SYN) до 10 в секунду.

Это может негативно повлиять на функциональность мониторинга сети при помощи Network Manager AtomMind.

Процесс мониторинга сети может значительно замедлиться. Если достигнут вышеупомянутый лимит, в Журнале Событий Windows можно найти следующую запись:

```
Product: Windows Operating System
ID: 4226
Source: TCPIP
Symbolic Name: EVENT_TCPIP_TCP_CONNECT_LIMIT_REACHED
Message: TCP/IP has reached the security limit imposed on the number of concurrent
(incomplete) TCP connect attempts
```

Т.о., не рекомендуется устанавливать Network Manager AtomMind на версии для настольной системы Microsoft Windows (Windows XP, Windows Vista, Windows 7) с целью мониторинга больших сетей. **Серверные версии** Windows (Windows 2000, 2003 Server или 2008 Server) не имеют данных ограничений, поэтому следует использовать именно их.

19.1.4 Сетевое обнаружение

Обнаружение - это действие по обнаружению чего-то нового. В сфере управления сетью обнаружение обозначает процесс нахождения нового сетевого устройства, определения запущенных на нем сервисов и приложений, а также добавление их для мониторинга и управления.

В частности, *процедура обнаружения* описана в разделе Процесс обнаружения 1811.

С практической точки зрения, обнаружение дополняет, расширяет и автоматизирует утомительную задачу ручной настройки и поочередного добавления устройств 1805. Оно позволяет сформировать базу инфраструктуры мониторинга, и затем поддерживать ее в актуальном состоянии. Более узко обнаружение устройства можно использовать для выполнения нескольких задач управления сетью:

- Интерактивное обнаружение нескольких устройств 1819 для начальной настройки мониторинга крупных сетевых сегментов.
- Обнаружение отдельного устройства 1819 оказывается полезным, когда необходимо добавить и настроить одно устройство для мониторинга.
- Скрытое обнаружение одного и нескольких устройств 1819 для полностью автоматизированной операции обнаружения (например, назначение выполнения этой операции периодически или в ответ на определенные события).
- Повторное обнаружение сервисов 1820 запускается на тех устройствах, которые были добавлены ранее.
- Автоматическое обнаружение и добавление 1820 устройств, которые выявляют себя при отправке серверу AtomMind уведомлений определенного типа (например, сообщение Syslog или SNMP-ловушки).

В AtomMind Network Manager обнаружение доступно через действия 87 **Обнаружение сетевых устройств** и **Обнаружение отдельных устройств** в контексте устройств 1494, а также может быть вызвано через определенные инструментальные средства пользовательского интерфейса (такие как контекстное меню, задачи, избранное и пр.), как описано в разделе Использование обнаружения 1818. Некоторые задачи процесса обнаружения 1811 могут также выполняться с использованием функции **Обнаружение сетевых устройств** контекста Обнаружение 1962.

19.1.4.1 Процесс обнаружения

В целом процедура обнаружения включает две основные *стадии*, каждая из которых состоит из нескольких *этапов*:

I. **Поиск** 1812**: Поиск устройств, находящиеся в режиме online, и сервисов, которые они выполняют согласно параметрам поиска** 1812. Стадия делится на четыре этапа, представляющих описание параметров поиска 1812 (подготовительная часть) и сканирование 1817:

1. **Выбор сервисов и установка параметров** 1812. Определите *список сервисов* для поиска во время обнаружения. Список сервисов - это часть параметров поиска 1812.

2. **Описание адресов хоста**[1815]. Выберите *адреса для сканирования* во время обнаружения (см параметры поиска[1812]).
3. **Настройка обнаружения**[1815]. Настройте, *как будет выполняться поиск* (точность, скорость, временное ограничение и пр.).
4. **Сканирование**[1817]. На этом этапе выполняется *сетевое обнаружение*. Система выполняет итерацию в списке заданных IP-адресов и проверяет доступность выбранных сервисов.

II. **Обработка результатов**[1817]: **Добавьте некоторые или все из обнаруженных устройств и настройте соответствующие сервисы.** Стадия включает в себя следующие этапы:

5. **Просмотр результатов обнаружения**[1817]. Выберите устройства для добавления/обновления и сервисы, которые требуется просмотреть.
6. **Применение результатов**[1818]. На этом этапе создаются/обновляются учетные записи устройства. Настройка отслеживаемых сервисов обновляется в параметрах учетной записи устройства.
7. **Отображение** (не обязательный этап). Если обнаружение выполняется в интерактивном режиме, система предлагает пользователю создать карту сети[1821] для обнаруженных устройств..
8. **Отчет о статусе обнаружения**[1818]. Окончательные результаты обнаружения демонстрируются пользователю. Это включает список отсканированных/созданных/обновленных устройств, ошибки обнаружения и статусы задач сканирования отдельного хоста[1817].

19.1.4.1.1 Обнаружение в стадии поиска

Поиск - первая стадия Процесса обнаружения[1811]. Его цель - обнаружить доступные сетевые хосты и определить, какие сервисы/приложения они выполняют в текущий момент. Это достигается при помощи сканирования[1817] IP-адресов для отдельных сервисов, т.е. осуществляется попытка установить связь с каждым из сервисов, назначенных для обнаружения на определенном IP-адресе. Способ выполнения сканирования определен в **параметрах поиска** обнаружения.

Параметры поиска определяют, *что следует искать, где искать* и *как искать*:

- **Что сканировать?** Ответ на этот вопрос приходит во время этапа **Выбора сервиса**[1812], список сервисов, которые должны быть обнаружены, задаются вместе с опциями сканирования для каждого из них.
- **Где сканировать?** Этап **Описание адресов хоста**[1815] позволяет составить список IP-адресов для сканирования.
- **Как сканировать?** Этап **Настройка обнаружения**[1815] позволяет задать параллельность процесса сканирования, ограничения времени и точности обнаружения.

Как устанавливаются параметры, зависит от режима обнаружения:

- В полностью интерактивном[1819] режиме подсказчик проводит пользователя через процедуру установки параметров обнаружения.
- В автоматическом[1820] режиме и режиме интерактивного обнаружения одного устройства[1819] используются параметры обнаружения по умолчанию[1820].
- Параметры для бездумного[1819] обнаружения выставляются в свойствах задачи[827].

Таким образом, мы можем сказать, что стадия **Поиска** разделена на *две части*: описание параметров[1812] и процесс сканирования[1817], т.е. непосредственно сам поиск с учетом заданных параметров.

19.1.4.1.1.1 Установка и выбор сервиса

Отвечая на вопрос **"Что находить?"**[1812], этап **Установка и выбор сервиса** задает *список сервисов,* которые вместе с *параметрами* будут использоваться для определения доступности сервиса.

Пинг-сервис - это единственный *обязательный* сервис, который проверяется во время обнаружения, поскольку он используется, чтобы определить "живой" ли (находится ли в режиме on-line) хост или адрес во время процедуры сканирования[1817]. Другие сервисы AtomMind Network Manager *опциональны.*

В **Параметры сервиса** включены следующие поля:

- Флажок **Обнаружение** обозначает, что сервис включен в процесс обнаружения (отмечен галочкой) или не включен (не отмечен галочкой).
- **Время ожидания** определяет период времени, назначенный на одну попытку определения сервиса во время сканирования[1817].
- **Попытки** определяет количество попыток для определения доступности сервиса.

- **Опции сервиса** задают параметры сервиса, используемого процедурой сканирования 1817.

Следующая таблица содержит список сервисов, которые AtomMind Network Manager может обнаружить, а также назначенные опции сервиса и краткое описание процедуры обнаружения.

Имя сервиса	Опции обнаружения	Действие обнаружения
Пинг	Packet Data Size (см. также Свойства пинга 597)	Проверяет, правильно ли хост отвечает на *запросы Ping (ICMP echo)*.
SNMP	Protocol Port SNMP Protocol Version Read Community Protocol Maximum PDU Size Initial OID Filter OID Use multi-variable requests to read tables Security Level Username Authentication Protocol Authentication Password Privacy (Encryption) Protocol Privacy (Encryption) Password (см. также Свойства связи по SNMP 640)	Проверяет, запущен ли *SNMP-агент* на хосте. Это выполняется посредством отправки запроса SNMP GETNEXT с изначальными OID в качестве параметров. Когда устновлен фильтр по OID, сервис считается "живым" только если устройство отвечает с правильным значением SNMP, и его OID входит в поддерево Фильтра OID; в противном случае, правильного ответа от устройства достаточно, чтобы считать его SNMP сервис "живым".
TCP	Port Timeout (см. также Свойства сервиса TCP 599)	Проверяет, слушает ли устройство назначенные TCP-порты, пытаясь установить связь по TCP с заданным Timeout. Только порты, к которым успешно подключен AtomMind Network Manager, будут настроены для дальнейшего мониторинга. Параметр Времени ожидания Обнаружения не принимается во внимание этим сервисом. Данные во время обнаружения хосту не отправляются.
UDP	Port Timeout Data To Send (см. также Свойства сервиса UDP 600)	Проверяет, отвечает ли хост, когда данные пользователя отправлены на удаленный порт в виде UDP-датаграммы. Порт считается "доступным", если AtomMind Network Manager получил ответ на его датаграмму. Лишь порты, от которых AtomMind Network Manager получил ответы, будут настроены для дальнейшего мониторинга. Параметры Времени ожидания обнаружения игнорируются этим сервисом.
HTTP	Port URL Request Type (см. также Свойства HTTP)	Проверяет, доступен ли веб-сервер на данном Port, отправляя запрос GET или POST с заданным URL на удаленный хост и читая ответ.
FTP	Port (см. также Свойства FTP 607)	Проверяет, принимает ли устройство *FTP-соединение* на определенном порту.

DNS	Port Protocol (см. также Свойства DNS 600)	Проверяет, отвечает ли устройство на заранее определенный *запрос поиска имени в DNS,* отправленный на Port через заданный Protocol (*UDP* или *TCP*). Если устройство правильно отвечает на запрос, мы полагаем, что сервис DNS работает.
IMAP	Port (см. также Свойства IMAP 602)	Проверяет, принимает ли сервис соединение на определенном Port и отвечает ли теми же самыми данными. Если соединение установлено успешно и получен не пустой ответ, сервис работает.
POP3	Port (см. также Свойства POP3 601)	Проверяет, принимает ли сервис соединение на определенном Port. Если соединение успешно установлено, и получен не пустой ответ, сервис работает.
SMTP	Port (см. также Свойства SMTP 603)	Проверяет, правильно ли устройство отвечает на *команду HELO Протокола SMTP* (согласно RFC 821). Если устройство отвечает *OK* (код ответа *250*), SMTP-сервис считается доступным.
SSH	Port (см. также Свойства SSH 610)	Проверяет, принимает ли устройство *SSH-соединение* (через *протокол SSH-2*) на определенном Port. Если соединение может быть установлено, сервис работает.
DHCP	MAC Address (см. также Свойства DHCP 611)	Проверяет, отвечает ли устройство на запросы DHCP. AtomMind Network Manager отправляет запрос *DHCP-обнаружения* и ожидает ответа. Если ответ получен в течение заданного времени ожидания, сервис считается доступным.
LDAP	Port Username Password (см. также Свойства LDAP 612)	Проверяет, принимает ли сервис LDAP-соединения с заданными параметрами. LDAP-сервис работает, если соединение установлено, и авторизация LDAP успешна.
Radius	Authentication Port Username Password Radius Secret (см. также Свойства радиуса 613)	Проверяет, принимает ли устройство *Запросы авторизации по радиусу*. Если устройство возвращает ответ *Access Accept* (см RFC 2138), сервис считается доступным.
WMI	Domain Username Password Namespace (см. также Драйвер устройства WMI 661)	Проверяет, принимает ли устройство соединения WMI.

Варианты параметров

Все сервисы, исключая *TCP-сервисы*, *UDP-сервисы* и *Пинг,* поддерживают множество **вариантов параметров**, используемых во время сканирования. Варианты предоставляются как множественные записи в таблице опции обнаружения сервиса. AtomMind Network Manager пытается обнаружить сервис, используя каждый заданный вариант.

Сервер останавливает итерацию вариантов параметров сервиса при обнаружении первого подходящего "рабочего" варианта.

Централизованное хранение опций обнаружения

Когда запущен новый процесс обнаружения, все опции обнаружения сервиса загружаются из *Глобальной настройки плагина обнаружения сетевого устройства* 207.

19.1.4.1.1.2 Определение адресов хостов

Этап **Определения Адресов Хостов** отвечает на вопрос **"Что сканировать?"** 1812. Ответ представляет собой *список IP-адресов и/или имен хоста* для сканирования 1817.

Обнаружение часто выполняется на крупных сегментах сети, с сотнями или даже тысячами адресов. Создание списка IP вручную, путем последовательного добавления отдельных адресов, оказывается в данных условиях очень утомительной задачей. По этой причине AtomMind Network Manager позволяет определить диапазон адресов и, тем самым, облегчает работу пользователей. Тем не менее проблема еще пока не решена, поскольку сетевые администраторы часто имеют дело с большим числом подсетей, сегментов и диапазоном адресов, ввод которых требует больших временных затрат. К счастью, в большинстве случаев нет необходимости определять диапазон вручную, поскольку информация уже хранится в таких сетевых устройствах, как роутеры и свичи. AtomMind Network Manager получает спецификации сети через SNMP и использует их для облегчения процедуры указания адресов.

Таким образом, список сканируемых адресов может быть сформирован в три этапа:

1) Определение **Адресов сходных маршрутизаторов**.
   Исходные маршрутизаторы - это маршрутизатор или коммутатор с поддержкой SNMP, которые можно использовать для получения информации об IP-подсетях, доступных в Вашей сети. Каждый из определенных *исходных маршрутизаторов* запрашивается на предмет сетевых интерфейсов, которые они поддерживают, а затем адресов интерфейсов, с которыми связан каждый сетевой интерфейс. Если адрес интерфейса представляет *сегмент сети* (или *подсеть*), AtomMind Network Manager получает все подсети, о которых знает маршрутизатор. Они включаются в список диапазонов IP и подсетей на следующем этапе.
   Используемые исходные маршрутизаторы определяют значения `IP Address or Host Name` вместе с `SNMP Connection Properties` , используемыми для получения требуемых данных из маршрутизатора/коммутатора.

2) Определение **диапазона IP**.
   Существует два способа определения *диапазона IP*:

   a) по IP-адресу и маске сети

   b) по паре (начало и конец) IP-адресов.

Все IP-адреса, принадлежащие определенным диапазонам, будут добавлены к списку индивидуальных IP-адресов с целью редактирования на следующем этапе.

3) Определение **индивидуальных IP-адресов и имен хоста**.
   Пользователь может редактировать список адресов, преобразовывая, удаляя или добавляя отдельные адреса.
   Полагая, что редактировать большие списки адресов бессмысленно, (например, автоматически сгенерированных из диапазонов IP), система предлагает пропустить этот этап, если в списке более тысячи адресов.

19.1.4.1.1.3 Настройка обнаружения

Этап **Настройки обнаружения** определяет, как долго будет длиться операция обнаружения, как много будет использовано вычислительных ресурсов, и насколько точным будет результат - все это отвечает на вопрос **"Как обнаруживать?"** 1812. Следующие параметры вводятся для обеспечения контроля над процессом обнаружения.

Параллельность (число потоков)

Сканирование 1817 включает в себя множество индивидуальных задач по определению отдельных сервисов, запущенных на том или ином устройстве. Поскольку каждое из этих заданий обычно вызывает относительно длительные задержки (от сотни миллисекунд до нескольких секунд), поочередное их исполнение может занять длительное время. Вместо этого они могут быть выполнены одновременно, значительно сократив общее время обнаружения.

Единовременное сканирование выполняется сразу в нескольких потоках. Итоговое количество параллельных потоков задано параметром **Параллельность (число потоков)**.

Большое количество потоков может помешать добиться цели. Ваш процессор будет стараться обработать много потоков (например, 2000 потоков на двухъядерной системе) и выполнение операции обнаружения замедлится.

Чтобы выбрать число потоков, следует учесть количество планируемых для сканирования хостов, а также доступные ресурсы и системные ограничения. Эти параметры индивидуальны для каждого случая, и невозможно рекомендовать конкретный алгоритм для определения потока "в целом". Однако можно рекомендовать следующие практические методы:

- Число потоков следует выбрать с учетом системных ограничений и рекомендаций (например, несколько сотен для ОС Windows).
- Учитывая упомянутые ограничения, число потоков может быть близким числу сканируемых хостов и не превышать их.

- Если другие процессы запущены на Вашей системе, Вам, возможно, придется сократить количество потоков. Изучите, что еще может потреблять ресурсы системы помимо AtomMind.

РЕКОМЕНДУЕМОЕ ЧИСЛО ПОТОКОВ

Потоки	Выполнение обнаружения
10	Медленная скорость обнаружения с минимальными издержками производительности для малых сетей.
100	Оптимальное значение для большинства сетей, с издержками выше среднего.
1000	Крупные издержки и большая нагрузка на CPU (до 100%). Подходит для обширных сетей.

Максимальное итоговое время обнаружения

Поскольку в отдельных случаях на операцию обнаружения может уйти длительный период времени, его можно ограничить, если выставить параметр **Максимальное итогового время обнаружения**. Как только время обнаружения превысит этот порог, процесс сканирования будет завершен. В этом случае результат будет содержать все устройства и сервисы, обнаруженные за это время.

Максимальное время обнаружения одного хоста

Ограничение по времени можно применить не только для всего процесса обнаружения, но и для задач сканирования одного хоста. Если сканирование отдельного хоста не завершено за **Максимальный период обнаружения одного хоста**, оно останавливается и обозначается как *незавершенное*. Обнаруженные сервисы будут хранится до этапа завершения обработки.

Обнаружить устройства, не отвечающие на пинг

Полному сканированию сервисов предшествует определение статуса on-line для каждого из определенных хостов. Отфильтровывая "мертвые" (находящиеся в режиме offline) адреса, можно в значительной мере сократить продолжительность процесса обнаружения. Однако это может вызвать неточность, поскольку некоторые существующие хосты могли отклонить запрос пинга.

Параметр **Обнаружить устройства, не отвечающие на пинг**, позволяет выбирать между скоростью и точностью. Если параметр отключен, сканирование может проигнорировать хосты, не отвечающие на запрос пинга, с целью сохранения скорости. В ином случае, все выбранные хосты будут полностью просканированы, обеспечивая наиболее точный результат.

Обновить существующие устройства

Во время процесса обнаружения будут сканированы все устройства обозначенных IP диапазонов. Параметр **Обновить существующие устройства** позволяет обновлять сервисы на существующих устройствах.

Описания устройств

Обычно желательно использовать описательные *имена хостов* вместо малозначащих *IP-адресов* там, где это возможно, т.е. автоматически назначать описания обнаруженным устройствам. AtomMind Network Manager предлагает для этого три метода:

1. Назначить непосредственно соответствующий **IP-адрес** (или имя хоста) как описание для обнаруженного устройства. Это наиболее быстрый, но менее описательный метод. В некоторых случаях, сетевые администраторы предпочитают описывать учетные записи устройств, используя для этого их IP-адреса, поскольку они могут сформировать хорошо скомпонованную систему и структуру идентификации хостов.

2. Используйте обратное разрешение имен **DNS,** чтобы найти полностью уточненное доменное имя для данного IP-адреса. На обратный просмотр может потребоваться длительное время, поэтому этот метод может увеличить общее время обнаружения.

3. Воспользуйтесь преимуществом данных SNMP, использовав переменную **SNMP sysName** в качестве описания учетной записи устройства**.** Она сохраняет "назначенное администратором имя для управляемого узла" (см. RFC1213), и часто (согласно правилу) это полностью уточненное доменное имя узла. Запросы SNMP обычно быстрее, чем обратный запрос в DNS, поэтому этот метод может улучшить выполнение обнаружения, по сравнению с предыдущим методом. Вместе с тем AtomMind Network Manager будет использовать обратный запрос к DNS для устройств, которые не поддерживают SNMP.

19.1.4.1.1.4 Сканирование

Сканирование - это ключевой этап процесса обнаружения, который выполняет поиск сервисов, доступных на сетевых хостах.

Получив определенный адрес, AtomMind Network Manager проверяет доступность сервисов, выбранных для обнаружения.Это называется задачей **сканирования одного хоста** и состоит из двух этапов:

1. **Проверка Сервиса пинг**[1813]. *Эхо-запрос ICMP* используется для тестирования *доступности устройства*. Если устройство правильно отвечает на запросы пинга, мы полагаем, что устройство доступно, и переходим на следующий этап сканирования. В ином случае, в зависимости от значения параметра Обнаружить устройства, не отвечающие на пинг[1816], мы либо переходим на следующий этап (если параметр `Discover Ping-Failed Devices` установлен как `true`), либо полагаем, что устройство недоступно, пропуская следующий этап (если `Discover Ping-Failed Devices` установлен как `false`).
2. **Проверка других сервисов**. Система пытается определить доступность каждого сервиса, выбранного для обнаружения.

Обратите внимание, что процесс общего сканирования ограничен Максимальным итоговым временем обнаружения[1816], в то время как *задачи сканирования одного хоста* ограничены Максимальным временем обнаружения одного хоста[1816].

19.1.4.1.2 Обработка результатов обнаружения

Это второй этап обнаружения. По окончании поиска нам нужно *обработать и применить полученные результаты*. Это включает в себя:

- Выбор интересующих устройств и сервисов.
- Просмотр и (возможно) изменение параметров поиска (см. **Просмотр результатов обнаружения**[1817])
- Применение выбранных результатов поиска (см. **Применение результатов поиска**[1818])

Интерактивный помощник обнаружения также предлагает пользователю **создать карту сети**[1821], и затем показывает **Отчёт статуса обнаружения**[1818], скомпонованный из результатов обнаружения.

19.1.4.1.2.1 Просмотр результатов обнаружения

После завершения стадии поиска[1812], мы получаем список обнаруженных устройств и сервисов. Некоторые из результатов могут оказаться полезными и должны быть использованы для обновления текущей настройки мониторинга, в то время как другие следует отбраковать. Это можно выполнить в *интерактивном режиме* путем редактирования Результатов Обнаружения:

Поле	**Описание**
Создать/обновить	Включите этот флаг, чтобы создать новую учетную запись устройства или обновить существующую.
Имя устройства	Имя учетной записи устройства. AtomMind Network Manager предлагает значение по умолчанию, но администратор в праве переписать его.
Описание устройства	Описание устройства автоматически генерируется согласно параметру Описание Устройства[1815], но может быть отредактировано.
Тип устройства	Автоматически определенный тип устройства. Системный администратор может изменить его, учитывая тип/цель используемого устройства.
Обнаруженные сервисы	Выбираемый список сервисов для мониторинга и их параметры:

Поле	**Описание**
Использовать сервис	Включите этот флаг, если следует отслеживать сервис.
Имя сервиса	Имя сервиса и/или описание.
Параметры сервиса	Параметры мониторинга, заданные для сервиса.

IP-адрес или имя хоста	Показывает адрес или имя хоста устройства.

В тихом режиме учетные записи для **всех** обнаруженных устройств будут добавлены/обновлены. **Все** обнаруженные сервисы будут активизированы для мониторинга с использованием настроек для сервиса в параметрах задачи 823.

19.1.4.1.2.2 Применение результатов обнаружения

На этом этапе сервер создает/обновляет учетные записи устройства и включает/отключает отслеживаемые сервисы.

Когда обновляются существующие учетные записи устройств, сервисы, не обнаруженные на них, не активизируются. Можно сказать, что существующие сервисы считаются "временно недоступными".

Единственным исключением здесь является сервис SNMP, который отключается, если процесс обнаружения не смог подключится к Агенту SNMP, запущенному на хосте. Сервис SNMP оказывается отключенным, потому что:

- мониторинг SNMP потребляет значительный объем системных ресурсов (время CPU, память, диск)
- сервис SNMP включается по умолчанию для добавляемых вручную устройств.

19.1.4.1.2.3 Отчет об обнаружении

Отчет об обнаружении суммирует итоговые результаты обнаружения и отображает список вовлечённых устройств и информацию о статусе:

Имя поля	**Информация о статусе**
IP-адрес и имя хоста	Хост, выбранный для обнаружения.
Закончено	Указывает на статус соответствующей задачи сканирования одного хоста 1817: • `Yes`, если сканирование хоста было успешно завершено, • `No`, если сканирование хоста было прервано по определенной причине и вовсе не было запущено.
Результат	Суммирует результаты обнаружения для хоста, объединяя результаты поиска 1812 и стадии обработки результата 1817: • причину прерывания для незавершенных задач сканирования одного хоста 1817; • основную информацию для только что созданных учетных записей устройства, включая их описание и тип; • примечания для обновленных существующих учетных записей устройства; • включенные/отключенные сервисы. Пустое значение в поле означает (в зависимости от значения поля *Завершено*): • если значение *Завершено* - `No`, пустое значение *Результат* показывает, что сканирование соответствующего устройства не было запущено (ввиду прерывания процесса обнаружения, например, из-за превышения ограничения `Maximum Total Discovery Time`); • если значение *Завершено* - `Yes`, пустое значение *Результат* показывает, что для устройства ничего не изменилось ("живой" сервис не был обнаружен или результат обнаружения был проигнорирован, поскольку не был выбран для применения).

19.1.4.2 Использование обнаружения

Случаи использования обнаружения можно разделить на две группы:

- в некоторых случаях необходимо сканировать несколько хостов, определяя рабочие и обнаруживая предлагаемые ими сервисы;
- в других случаях есть лишь один хост, который нужно исследовать и определить, какие сервисы на нем запущены.

Т.о., AtomMind Network Manager предлагает два действия обнаружения: *обнаружение множества устройств и обнаружение одного устройства*.

Так же, как и любое действие AtomMind, обнаружение можно запустить в любом из двух режимов: *интерактивном* или *тихом*. Последний позволяет автоматически повторять обнаружение с заданными параметрами (например, планируя его), в то время как первый обеспечивает лучший контроль за процессом, поддержку которого осуществляет помощник, позволяющий контролировать процесс обнаружения при помощи заданных параметров для каждого этапа обнаружения, или, как вариант, позволяющий прекратить обнаружение.

В целом, обнаружение реализовано как действия `Обнаружение сетевых устройств` и `Обнаружение одного устройства`, находящиеся в контексте `Устройства` (`users.admin.devices`). Действия можно активизировать при помощи определенных инструментальных средств пользовательского интерфейса (таких как контекстное меню, задачи, избранные объекты и пр.). Следующая таблица содержит типы обнаружения, которые делятся на четыре группы:

	Действие обнаружения множества устройств	**Действие обнаружения одного устройства**
Интерактивный режим	Интерактивное обнаружение множества устройств	Интерактивное обнаружение одного устройства Повторное обнаружение
Тихий режим	Тихое обнаружение множества устройств	Тихое обнаружение одного устройства Автоматическое обнаружение

К информации о том, как использовать все типы обнаружения, можно перейти по ссылкам.

19.1.4.2.1 Интерактивное обнаружение множества устройств

Интерактивное обнаружение множества устройств позволяет сканировать крупные сетевые сегменты, полностью контролировать и настраивать процесс используя понятный GUI-интерфейс. Интерактивный помощник помогает пользователю пройти через все этапы обнаружения. Помощник позволяет отслеживать процесс, просматривать и изменять опции для всех аспектов обнаружения, а также прерывать процесс на любом из этапов.

Чтобы активизировать обнаружение в интерактивном режиме множества устройств, можно выбрать действие **Обнаружение Сети** () из контекстного меню или связанные действия из панели контекст **Устройств** () в системном дереве.

Используйте элемент **Сетевое обнаружение** из Избранного, чтобы быстро перейти к действию Полное интерактивное обнаружение.

19.1.4.2.2 Интерактивное обнаружение одного устройства

Интерактивное обнаружение одного устройства обеспечивает быстрый способ обнаружения сервисов отдельного хоста. Система запрашивает имя хоста или IP-адрес, на которых следует обнаружить сервисы. Больше не требуется ничего вводить, поскольку используются параметры обнаружения по умолчанию и, соответственно, выполняются следующие действия:

- Будет создана учетная запись устройства с адресом/именем, если она не существует;
- Все обнаруженные сервисы будут настроены для мониторинга в новой/существующей учетной записи.

Интерактивное обнаружение одного устройства можно активизировать, выбрав действие **Обнаружить отдельное устройство** из Контекстного меню или Связанные действия панели контекста **Устройства** () в Системном дереве.

19.1.4.2.3 Тихое обнаружение одного и множества устройств

Тихий режим не предполагает взаимодействия с пользователем, а *опции обнаружения* определяются заранее, когда действие настраивается на не интерактивное исполнение: как часть Действий автоматической коррекции при извещениях об ошибках или свойства запланированных задач. Действия `Network Device Discovery` и `Discover Single Device` находятся в контексте `Devices` (`users.admin.devices`).

Чтобы создать задание обнаружения:

- создайте новую задачу планировщика, выбрав действие **Создать Запланированное Задание** () из контекста **Запланированные Задания** ().
- Заполните **свойства запланированного задания**:
  - Выберите контекст **Устройства** () (под учетной записью соответствующего пользователя) в поле **Маска Контекста**.
  - Убедитесь, что выбранное **Действие** - **Сетевое Обнаружение**.
  - Определите нужные **Параметры Обнаружения**.
- Кликните OK, чтобы создать новое задание.
- Добавьте один или несколько **Триггеров**, чтобы назначить график обнаружения.

19.1.4.2.4 Повторное обнаружение сервисов устройства

Сервис повторного обнаружения - это действие обнаружения, которое применяется к существующей учетной записи устройства. **Повторное обнаружение** отдельного устройства доступно как действие 87 в контекстном меню 372 или панели Связанные действия 372 контекста устройства.

После того, как помощник Повторного обнаружения запущен, он запрашивает, какие сервисы необходимо обнаружить, (см. раздел Установка и выбор сервиса 1812), выполняет сканирование 1817, представляет результаты для просмотра 1817, а затем применяет 1818 их.

19.1.4.2.5 Автоматическое обнаружение

Автоматическое обнаружение вызывает действие Обнаружение одного устройства 1819 в тихом 1819 режиме автоматически в ответ на появление сообщения *SNMP-ловушек* и/или *SysLog*. Это помогает автоматизировать задачу обнаружения новых устройств, при этом настройка мониторинга всегда остается актуальной.

Чтобы активизировать Автоматическое обнаружение:

- Для SNMP-ловушек: активизируйте `Автоматическое обнаружение SNMP-ловушек` в глобальных настройках плагина 207 `Сетевого Устройства`;
- Для Syslog-сообщений: активизируйте `Автоматическое обнаружение источников сообщений Syslog` в глобальных настройках плагина `Syslog`.

19.1.4.3 Параметры обнаружения по умолчанию

Параметры обнаружения по умолчанию предоставляют параметры по умолчанию для нахождения сервисов и их последующего мониторинга. Эти параметры используются для обнаружения одного устройства 1819, автоматического обнаружения 1820, и как значения по умолчанию для полного интерактивного 1819 обнаружения и тихого 1819 обнаружения.

Чтобы редактировать параметры обнаружения по умолчанию, зайдите в глобальные настройки 207 плагина 207 Обнаружение устройств сети.

19.1.5 Визуализация сети

AtomMind Network Manager визуализирует различные аспекты состояния сети и приложений, используя

- Инструментальные панели 912 общего обзора и по отдельным устройствам
- Статические, географические и топологические карты 1821

Интерфейс операторов первой линии

Операторы центра управления сетью и системные администраторы обычно начинают пользоваться AtomMind Network Manager, открывая одну инструментальную панель, которая визуализирует общее состояние сети. Продукт включает несколько заранее настроенных инструментальных панелей для беглого просмотра состояния сети:

ОБЗОР СЕТИ

Инструментальная панель обзора сети отображает текущее состояние вашей сети и ее проблемы.

ТОП 10

Эта инструментальная панель позволяет просматривать список Топ 10 устройств с наивысшей нагрузкой на процессор, использованием памяти/диска, трафиком и утилизацией интерфейса, временем отклика, коэффициентом потери пакетов и другим.

Она также отображает процессы/сервисы, которые потребляют большое количество ресурсов.

ОБЗОР УСТРОЙСТВ СЕТИ

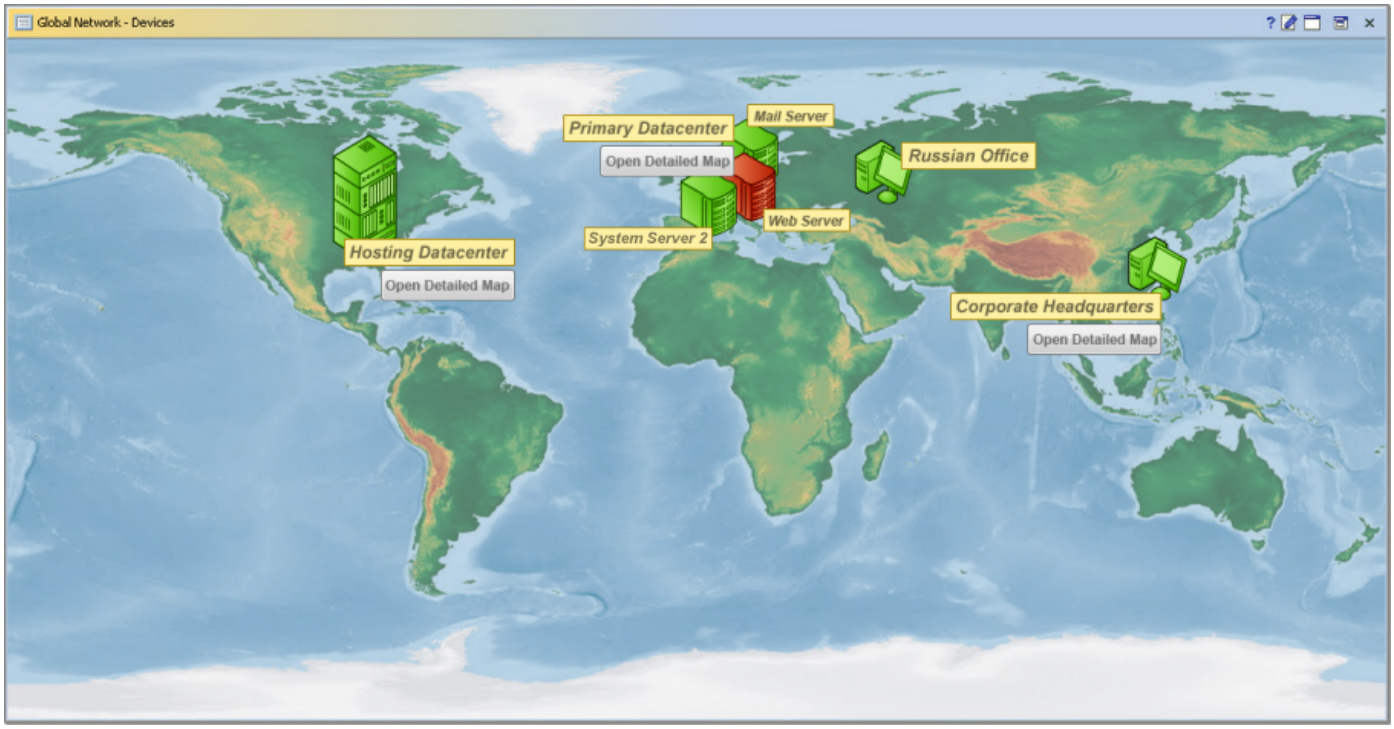
Инструментальная панель устройств отображает ключевые индикаторы состояния и производительности сетевых устройств:

- Общая информация об устройстве (тип, операционная система, размер памяти, время подключения и т.д.)
- Диаграмма доступности устройств
- Графики времени отклика и коэффициента потери пакетов
- График загрузки процессора
- График трафика и пропускной способности
- Статистика интерфейса сети устройства
- Список состояний сервисов
- Таблица активных тревог устройства
- График использования памяти и дискового пространства
- Таблица объемов хранения

19.1.5.1 Карты сети

Карты и диаграммы сети используются для наглядного представления топологии и состояния отслеживаемой IT-инфраструктуры. **Карты сети** - это разновидность Карт устройств 1315, они могут описывать любое отслеживаемое оборудование, представленное как Контекст Устройства в AtomMind, включая маршрутизаторы, коммутаторы, серверы, рабочие станции.

Заметим, что карты сети статичны. См. просмотр топологии сети 1827, чтобы узнать, как построить динамические диаграммы топологии сети.



Карты можно настроить так, чтобы они отражали сеть наиболее подходящим для Вас образом:

- Можно установить цвет фона, текстуры или рисунка.

- Сетевые устройства могут быть представлены заранее заданными типовыми изображениями.
- Статус устройства может быть представлен текстовыми ярлыками и/или обозначен цветом.
- Карты можно встроить на каждом уровне детализации (можно детализировать в глубину или увеличивать изображение).

Дистрибутив AtomMind Network Manager включает сотни динамических векторных изображений, которые могут использоваться на картах сети. Вот пример, иллюстрирующий лишь некоторые из них:

19.1.5.2 Визуализация сети

Этот раздел описывает обнаружение и мониторинг топологии сети, т.е. соединения между отдельными портами и интерфейсами различных сетевых устройств.

Существует несколько основных аспектов мониторинга топологии:

- Обнаружение 1821 топологии
- Редактирование 1826 топологии
- Просмотр 1827 топологии

19.1.5.2.1 Обнаружение топологии сети

AtomMind Network Manager умеет обнаруживать топологию сети на уровне 2 (уровень соединений) и уровне 3 (уровень IP сети) сетевой модели OSI. Обнаружение топологии строится на анализе информации, собранной с использованием сетевых протоколов, в основном протокола SNMP.

Предварительные условия

Чтобы позволить AtomMind Network Manager успешно обнаруживать топологические соединения в вашей сети, убедитесь, что выполняются следующие условия:

- Вы завершили обнаружение сетевых устройств 1811 при помощи сканирования подсетей ваших IP сетей. В ином случае можно вручную создать учетные записи для всех или некоторых устройств сети.

На топологической карте будут отображаться только устройства, имеющие соответствующие учетные записи устройств 497 сервера.

- Учетные записи всех устройств, которые поддерживают SNMP, особенно коммутаторы и маршрутизаторы, имеют включенную опцию опроса SNMP и имена сообществ SNMP или правильно настроенное подтверждение аутентификации SNMP v3.
- Синхронизация 514 всех устройств успешно завершена (в большинстве случаев полностью синхронизированные устройства представлены иконкой в системном дереве 370).

Автоматическое обнаружение топологии

AtomMind Server заранее настроен для выполнения периодического обнаружения топологии сети каждые 5 минут. Обнаружение стартует при помощи задачи по расписанию 823 **Обнаружение топологии сети**.

Для отключения автоматического обнаружения топологии или изменения ее периода отключите задачу по расписанию или измените ее настройку простое расписание 828.

Обнаружение топологии вручную

Чтобы вручную запустить обнаружение топологии, откройте контекстное меню узла Сервера ( ) в системном дереве и выберите **Обнаружение топологии сети** из подменю **Управление сетью**.

Процесс обнаружения топологии

В большинстве случаев обнаружение топологии сети заканчивается за считанные секунды, поскольку вся запрошенная информация уже содержится в моментальных снимках 502 сетевых устройств сервера. Как только заканчивается обнаружение, возможно увидеть количество вновь обнаруженных и обновленных соединений в статусной строке AtomMind Client.

19.1.5.2.2 Алгоритмы обнаружения топологии сети

CDP (Cisco Discovery Protocol)

ВАЖНЫЕ ПЕРЕМЕННЫЕ

cdpCacheTable, ifTable, ipAddrTable

АЛГОРИТМ

Весь алгоритм основан на анализе переменной cdpCacheTable. Как только сетевое устройство получает эту переменную, следующий набор инструкций выполняется для каждого отдельного ряда:

1. Получить значение cdpCacheIfIndex и сохранить его как исходный интерфейс для потенциальной связи;

2. Получить значение cdpCacheDevicePort и сохранить его как описание сетевого интерфейса для потенциальной связи;

3. Проверить значение поля cdpCacheAddressType. Если равно 1 (IP-адрес), получить фактический адрес из поля cdpCacheAddress и попытаться найти соответствующее сетевое устройство в системе;

4. Если устройство найдено, проверить наличие cdpCacheTable;

4.1. Если в наличии, попытаться найти IP-адрес начального сетевого устройства в любом cdpCacheAddress поле таблицы. Как только найдено, сохранить значение соответствующего поля cdpCacheIfIndex как целевой интерфейс для потенциальной связи;

4.2. В противном случае, проверить содержание переменной ifTable. Если описание сетевого интерфейса (шаг 2) присутствует в каком-либо ifDescr поле таблицы, использовать значение ifIndex того же ряда в качестве целевого интерфейса;

5. Если ни исходный, ни целевой интерфейсы еще не упоминались ни в одной другой связи топологии, сохранить новую связь с этими интерфейсами.

Алгоритм на основе таблиц маршрутизации

ВАЖНЫЕ ПЕРЕМЕННЫЕ

ipCidrRouteTable (or ipRouteTable), ipAddrTable

АЛГОРИТМ

Вся необходимая для алгоритма информация включена в переменную ipCidrRouteTable (или ipRouteTable при отсутствии).

1. Получить тип маршрута (ipCidrRouteType или ipRouteType). Для любого значения, отличного от 4 (непрямой маршрут), перейти к выполнению следующего шага;

2. Получить исходный интерфейс (ipCidrRouteIfIndex или ipRouteIfIndex);

3. Перейти к значению следующего шага трассировки (ipRouteIfIndex или ipRouteNextHop);

4. Для каждого устройства сети в системе проверить его переменную ipAddrTable. Если следующий шаг трассировки (шаг 3) присутствует в каком-либо поле ipAdEntAddr, сохранить соответствующее значение поля ipAdEntIfIndex как целевой интерфейс;

5. Если ни исходный, ни целевой интерфейсы еще не упоминались ни в одной другой связи топологии, сохранить новую связь с этими интерфейсами.

LLDP (Link Layer Discovery Protocol)

ВАЖНЫЕ ПЕРЕМЕННЫЕ

lldpRemTable, lldpLocPortTable, ifXTable (or ifTable), lldpLocChassisId

АЛГОРИТМ

Для обработки при помощи данного алгоритма, сетевое устройство должно иметь как lldpRemTable, так и lldpLocPortTable переменные. Для каждого такого устройства выполняется следующий набор инструкций:

1. Получить тип шасси (поле lldpRemChassisIdSubtype в lldpRemTable);

2. Если это тип 4 (MAC-адрес), получить фактический адрес из поля lldpRemChassisId и попытаться найти соответствующее сетевое устройство в системе. Если устройство найдено, сохранить его как целевое устройство;

3. Получить номер локального порта (поле lldpRemLocalPortNum в lldpRemTable);

4. С использованием номера порта найти соответствующую запись в lldpLocPortTable (поле lldpLocPortNum) и проверить ID-подтип его порта (поле lldpLocPortIdSubtype);

5. В случае подтипа 3 (MAC-адрес) или 7 (локальный адрес), получить фактический адрес из поля lldpLocPortId и попытаться найти его в переменной ifXTable (или ifTable) (поле ifPhysAddress) и использовать значение соответствующего поля ifIndex как исходный интерфейс;

6. Если в таблице нет подходящей записи, просто использовать номер локального порта (шаг 3) как исходный интерфейс;

7. Получить ID подтип удаленного порта (поле lldpRemPortIdSubtype);

8. Используя подтип порта, вычислить целевой интерфейс, как описано в шаге 5;

9. Если это невозможно, получить описание удаленного порта (поле lldpRemPortDesc в lldpRemTable) и попытаться найти соответствующую запись в переменной lldpLocPortTable целевого устройства (поле lldpLocPortDesc). Использовать значение из поля lldpLocPortNum как целевой интерфейс;

10. Если все равно безрезультатно, попытаться найти ID локального устройства (переменная lldpLocChassisId) в lldpRemTable целевого устройства (поле lldpRemPortId). Использовать значение из поля lldpRemLocalPortNum как целевой интерфейс;

11. Если ни исходный, ни целевой интерфейсы еще не упоминались ни в одной другой связи топологии, сохранить новую связь с этими интерфейсами.

STP (Spanning Tree protocol)

Поддерживаются два STP режима: Classic STP и Multiple STP (MSTP)

Classic STP

ВАЖНЫЕ ПЕРЕМЕННЫЕ

dot1dStpPortTable, dot1dBaseBridgeAddress, dot1dStpPriority, dot1dBasePortTable, ifXTable (or ifTable)

АЛГОРИТМ

Данный алгоритм доступен только для сетевых устройств, имеющих переменные dot1dStpPortTable и dot1dBaseBridgeAddress.

1. Получить основной адрес моста (dot1dBaseBridgeAddress);

2. Для каждой отдельной записи в dot1dStpPortTable получить ID соседнего моста (поле dot1dStpPortDesignatedBridge), ID соседнего порта (поле dot1dStpPortDesignatedPort) и соседний MAC-адрес (адрес извлекается из ID соседнего моста, начиная с 7-го символа);

3. Если основной адрес моста и соседний MAC-адрес не совпадают, перейти к следующему шагу;

4. Попытаться найти соседний коммутатор в системе. Он должен содержать соседний MAC-адрес в своей ifXTable (поле ifPhysAddress);

5. Получить исходный порт (поле dot1dStpPort);

6. Попытаться найти запись в переменной dot1dBasePortTable с указанием исходного порта (поле dot1dBasePort). Если такая запись имеется, использовать значение ее поля dot1dBasePortIfIndex как исходный интерфейс;

7. Проверить, содержит ли соседнее устройство переменную dot1dStpPortTable;

7.1. Если содержит, попытаться найти запись в переменной dot1dStpPortTable такую, чтобы значение ее поля dot1dStpPortDesignatedBridge совпадало с ID соседнего моста (шаг 2), а значение поля dot1dStpPortDesignatedPort совпадало с ID соседнего порта (шаг 2);

7.1.1. Получить значение поля dot1dStpPort записи и конвертировать его, как описано в шаге 6. Использовать вычисленное значение как целевой интерфейс;

7.2. В противном случае, разделить ID соседнего порта (шаг 2) при помощи разделителя в виде двоеточия (:) и использовать второй фрагмент как целевой интерфейс;

8. Если ни исходный, ни целевой интерфейсы еще не упоминались ни в одной другой связи топологии, сохранить новую связь с этими интерфейсами.

MSTP

ВАЖНЫЕ ПЕРЕМЕННЫЕ

swMSTPMstPortTable, swMSTPPortTable, dot1dBaseBridgeAddress, ifXTable (or ifTable)

АЛГОРИТМ

Данный алгоритм доступен только для сетевых устройств, имеющих переменные swMSTPMstPortTable и dot1dBaseBridgeAddress.

1. Получить основной адрес моста (dot1dBaseBridgeAddress);

2. Для каждой отдельной записи в swMSTPMstPortTable, не имеющей статус "неактивный" (значение поля swMSTPMstPortStatus не равно 2), получить MAC-адрес заданного моста (адрес извлекается из заданного моста (поле swMSTPMstPortDesignatedBridge field), начиная с 7-го символа);

3. Если основной адрес моста и заданный MAC-адрес не совпадают, перейти к следующему шагу;

4. Попытаться найти заданный коммутатор в системе. Он должен содержать заданный MAC-адрес в его ifXTable (поле ifPhysAddress);

5. Получить исходный порт (поле swMSTPMstPort field). Использовать его как исходный интерфейс;

6. Проверить, содержит ли заданное устройство переменную swMSTPMstPortTable;

7. Если содержит, попытаться найти запись в переменной swMSTPMstPortTable такую, чтобы значение ее поля swMSTPMstPortDesignatedBridge совпадало с основным адресом моста (шаг 1);

8. Использовать значение ее поля swMSTPMstPort как целевой интерфейс;

9. Если ни исходный, ни целевой интерфейсы еще не упоминались ни в одной другой связи топологии, сохранить новую связь с этими интерфейсами.

AFT алгоритм (Address Forwarding Tables)

ВАЖНЫЕ ПЕРЕМЕННЫЕ

Отсутствуют

АЛГОРИТМ

1. Попытаться найти прямые связи;

1.1. Для каждого отдельного коммутатора в системе выполнить итерацию по его AFT;

1.2. Получить исходный интерфейс записи;

1.3. Для каждого сетевого устройства, связанного с этим интерфейсом, проверить, содержит ли его AFT обратную связь;

1.4. При обнаружении обратных связей, использовать ее интерфейс как целевой;

1.5. Если ни исходный, ни целевой интерфейсы еще не упоминались ни в одной другой связи топологии, сохранить новую связь с этими интерфейсами.

2. Попробовать найти связи через L1 коммутаторы;

Если кратко, этот шаг включает попытку найти пару сетевых устройств (или несколько пар) так, чтобы:

2.1. Набор устройств, связанных с определенным сетевым интерфейсом исходного устройства, имел некоторые общие элементы с набором устройств, связанных с определенным сетевым интерфейсом целевого устройства;

2.2. В то же время набор устройств НЕ связанных с одним и тем же сетевым интерфейсом исходного устройства, не должен иметь общих элементов с набором устройств, НЕ связанных с одним и тем же сетевым интерфейсом целевого устройства;

2.3. Если выполняются оба условия, проверить, упоминались ли исходный и целевой интерфейсы в какой-либо другой связи топологии. Если нет, сохранить новую связь с этими интерфейсами.

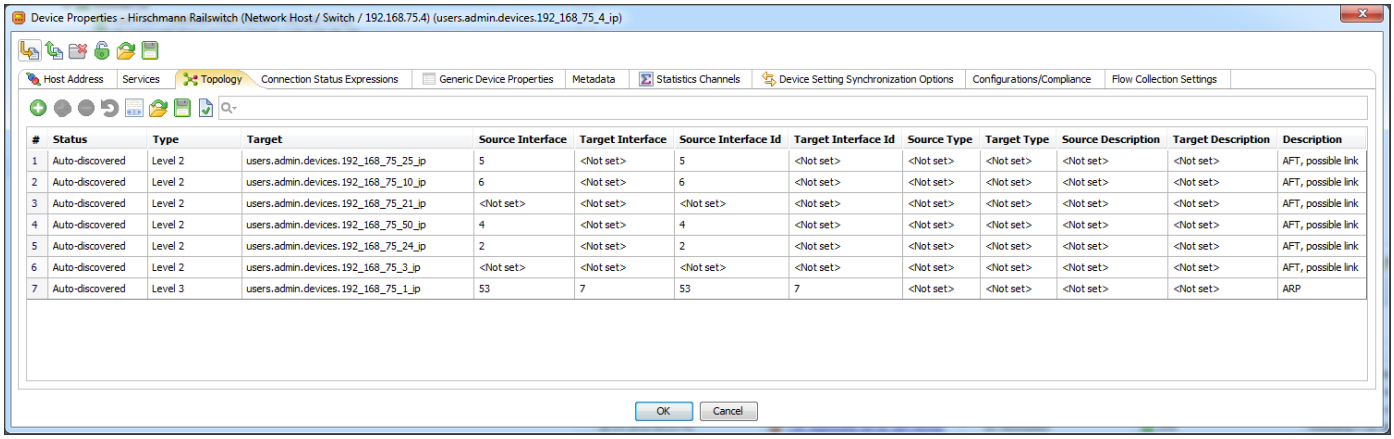
3. Попытаться предположить связи. Предположение связей - довольно сложный алгоритм, и в данном разделе не объясняется.

19.1.5.2.3 Редактирование топологии сети

Информация о топологии сети содержится в учетных записях[497] устройств сети. Каждая учетная запись устройства содержит информацию о входящих и/или исходящих соединениях.

Соединение всегда устанавливает связь между двумя устройствами, но только одна из их учетных записей будет содержать информацию об этом соединении.

Часть базы данных топологии, относящаяся к определенному устройству, может редактироваться путем запуска действия Редактировать свойства устройства[1494] из контекста устройства сети и переключаться на вкладку **Топология** ():



#	Status	Type	Target	Source Interface	Target Interface	Source Interface Id	Target Interface Id	Source Type	Target Type	Source Description	Target Description	Description
1	Auto-discovered	Level 2	users.admin.devices.192_168_75_25_ip	5	<Not set>	5	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	AFT, possible link
2	Auto-discovered	Level 2	users.admin.devices.192_168_75_10_ip	6	<Not set>	6	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	AFT, possible link
3	Auto-discovered	Level 2	users.admin.devices.192_168_75_21_ip	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	AFT, possible link
4	Auto-discovered	Level 2	users.admin.devices.192_168_75_50_ip	4	<Not set>	4	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	AFT, possible link
5	Auto-discovered	Level 2	users.admin.devices.192_168_75_24_ip	2	<Not set>	2	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	AFT, possible link
6	Auto-discovered	Level 2	users.admin.devices.192_168_75_3_ip	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	<Not set>	AFT, possible link
7	Auto-discovered	Level 3	users.admin.devices.192_168_75_1_ip	53	7	53	7	<Not set>	<Not set>	<Not set>	<Not set>	ARP

Настройки соединения топологии

База данных топологии содержит следующую информацию о каждом соединении:

- **Статус**. Соединения со статусом **Автоматически обнаруженные** создаются во время обнаружения топологии. Такие соединения будут автоматически управляться (редактироваться/удаляться) во время последующих циклов обнаружения. Соединения со статусом **Определенные пользователем** устанавливаются при добавлении вручную или переопределении. Такие соединения видны на топологической карте, но никогда не редактируются и не удаляются при процессе обнаружения. И, наконец, соединения со статусом **Удалено** не отображаются на картах, они также не редактируются и не удаляются при процессе обнаружения. Этот статус помогает спрятать соединения и защитить их от повторного автоматического обнаружения и повторного создания.
- **Тип**. Тип соединения: **Уровень 2**, **Уровень 3** OSI или общее соединение **Сеть**.
- **Цель**. Равновправный узел этого соединения, являющийся путем[42] контекста устройства, к которому ведет соединение.
- **Интерфейс источника**. Количество соединений сети в устройстве-источнике.
- **Интерфейс цели**. Количество соединений сети в целевом устройстве.
- **Описание**. Описание пользовательского соединения.

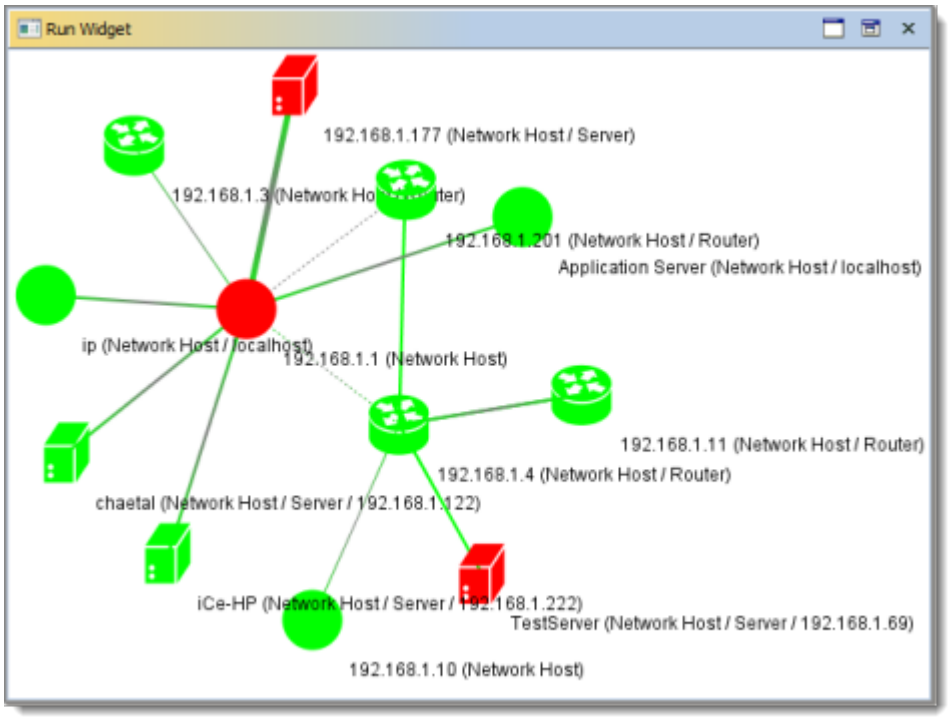
19.1.5.2.4 Просмотр топологии сети

Топология сети лучше всего визуализируется при помощи диаграмм. В AtomMind карты топологии запускаются виджетами 943, в частности компонентом виджета диаграмма 1025. Диаграммы топологии очень удобны для мониторинга топологии и статуса сети, поскольку они поддерживают:

- Приближение, панорамирование, вращение и смещение
- Выбор и перемещение отдельных узлов
- Просмотр статусов и пропускной способности отдельных соединений и шлюзовых интерфейсов
- Просмотр статусов сетевых устройств
- Работу с инструментальными панелями отдельных устройств

Карта топологии сети

Инсталляция AtomMind Network Manager включает виджет **Карта топологии сети**, который заранее настроен для обеспечения оптимального просмотра всей топологии:



Однако этот виджет полностью настраивается, как и любое другое средство управления данными AtomMind. Возможно изменить:

- Ряд параметров сетевых устройств, отображаемых на карте топологии
- Тип компоновки диаграммы топологии и ее параметры
- Тип карты (уровень 2, уровень 3 или совмещенный)
- Период обновления карты
- Любые другие настройки

19.1.5.2.5 Трассировка маршрута сети

AtomMind Network Manager может выполнять автоматическую трассировку маршрута сети (используя пакеты ICMP или UDP подобно тому, как делает утилита `traceroute` Unix/Linux). Для более подробного ознакомления см. Сервис трассировки маршрута 607.

Тревоги трассировки маршрута

Имя тревоги	Условия активации	Примечания
Хост не в маршруте	Хост (определенный как IP-адрес или имя хоста) не появляется в результатах	Для создания используйте группу **Трассировка пути** в действии

	маршрута.	Установить профайл мониторинга 1808.
Слишком долгое время отклика хоста	Время отклика хоста в **маршруте** приводит к превышению порога (определенного в миллисекундах)	Для создания используйте группу **Трассировка пути** в действии Установить профайл мониторинга 1808.
Изменение в количестве шагов	Количество шагов в **маршруте** изменилось.	Для создания используйте группу **Трассировка пути** в действии Установить профайл мониторинга 1808.
Слишком много шагов	Количество шагов в **маршруте** превышает заданный порог.	Для создания используйте группу **Трассировка пути** в действии Установить профайл мониторинга 1808.

19.1.6 Мониторинг и управление по SNMP

Простой Протокол Сетевого Управления (**SNMP**) - это сетевой протокол, основанный на UDP, широко используемый для мониторинга и контроля за сетевыми устройствами, сервисами и приложениями.

SNMP определен набором стандартов, разработанных Группой специалистов по разработкам в интернете (IETF), который касается не только самого протокола, но и языка спецификаций (SMI и последующего SMIv2), Информационной Базы Управления (MIB), ряда стандартных определений MIB и даже архитектуры агентов.

Обзор и основные концепции SNMP представлены в разделе Основы SNMP 1829.

В AtomMind Network Manager поддержку SNMP обеспечивает Драйвер устройства сетевого хоста 598. Драйвер обеспечивает:

- операции чтения и записи по SNMP
- отслеживание устройств, совместимых с SNMP:
  - опрос (периодическое чтение полной или выборочной информации, доступной через SNMP)
  - получение и обработка SNMP-ловушек
- управление стандартными и общими устройствами SNMP
- кэширование информации на сервере
- хранение информации в БД
- управление опциями мониторинга:
  - периоды синхронизации
  - стратегии опроса: все OID, OID, распознаваемые директорией MIB или только важные.
  - запуск автоматических действий в ответ на SNMP-ловушки
- управление по MIB, включая:
  - набор стандартных MIB
  - инструментальные средства для добавления и настройки использования пользовательских MIB

SNMP используется в сетевом обнаружении, чтобы:

- находить маршрутизатор и получать информацию о топологии сети
- обнаруживать устройства, совместимые с SNMP
- получать некоторые их свойства

В разделе Использование SNMP 1831 описаны различные практические вопросы использования SNMP в AtomMind Network Manager.

19.1.6.1 Основы SNMP

SNMP - это протокол связи между системными *администраторами* и *управляемыми системами*. Управляемая система запускает программный компонент, именуемый *агентом*, который отправляет информацию администратору через SNMP. Данные управления выводятся в виде переменных, описывающих конфигурацию системы. Эти переменные можно затем запросить (и иногда установить) через *управляющие приложения*, например, *AtomMind Network Manager*. Связь между объектами управления реализуется путем обмена *сообщений* протокола. Отдельное SNMP-сообщение часто называют *PDU-модулем данных протокола*.

Этот раздел содержит следующие темы:

- Версии SNMP 1829
- Структура информации управления 1829
- Операции SNMP 1830
- Безопасность SNMP 1830.

19.1.6.1.1 Версии SNMP

AtomMind Network Manager поддерживает SNMPv1 (1-ю версию Простого протокола сетевого управления), SNMPv2c (2-ю версию Простого протокола сетевого управления, основанного на объединении) и SNMPv3 (3-ю версию Простого протокола сетевого управления).

SNMPv1 - это первичное внедрение протокола SNMP, который все еще поддерживают многие поставщики. Текущее состояние SNMPv1 определяют RFC 1157, RFC 1155, RFC 1212, RFC 1213, RFC 1215.

SNMPv2c, определенный в RFC 1901, RFC 2578, RFC 2579, RFC 2580, RFC 3416, RFC 3417 и RFC 3418, улучшает версию 1 в областях производительности и взаимодействия агентов. SNMPv2 совместим с SNMPv1 в двух областях: форматы сообщения и операции протокола.

Основная проблема SNMPv1 и SNMPv2c - это слабая модель безопасности 1830.

SNMPv3 (также известный как STD0062) улучшает безопасность, добавляя свойства конфиденциальности, интеграции и авторизации. Эту версию SNMP определяет RFC 3411 -- RFC 3418. IETF наделил SNMPv3 полноценными интернет-стандартами (самый высокий уровень для RFC) и признает его как текущую стандартную версию SNMP.

Как показывает практика, все выше упомянутые версии SNMP широко используются в настоящий момент. RFC 3584 описывает общие аспекты совместимости между этими версиями SNMP.

19.1.6.1.2 Структура данных управления

То, как административная информация представлена, и какие данные она содержит, определяет 1-я версия Структуры административной информации (SMIv1 , RFC 1155) и 2-я версия Структуры административной информации (SMIv2 , RFC 2578). Управляемые системы представляют свою конфигурацию в форме *управляемых объектов* или *переменных*. Значения управляемых объектов можно запросить и иногда настроить через административные приложения.

Управляемые объекты описаны несколькими атрибутами. Наиболее важные - идентификаторы объектов (имени), типа, и шифрования.

ИДЕНТИФИКАТОРЫ ОБЪЕКТОВ

Идентификатор объектов (OID) уникально определяет управляемый объект.

Управляемые объекты организованы в виде древовидной иерархии. Идентификатор объектов показывает место отдельного объекта в данной иерархии, как ряд целых чисел, находящихся на узлах дерева и разделенных точками (.). Это известно как числовая форма Идентификатора объектов. Есть также и понятная человеку форма, которая представляет каждый идентификатор объектов как ряд имен.

Пример: текстовое описание управляемого объекта представляет объект с числовым ID `1.3.6.1.2.1.1.1`, который соответствует понятному имени `iso.org.dod.internet.mgmt.mib-2.system.sysDescr`.

ТИП

Тип управляемого объекта в контексте SNMP определен подклассом Абстрактной синтаксической нотацией версии один (ASN.1). ASN.1 определяет независимую от ПК форму представления и передачи данных между администраторами и агентами.

Существует несколько основных типов. Их можно использовать для определения более сложных объектов, содержащих другие объекты и пр., таким образом, составляющие части деревообразной иерархии, упомянутой ранее.

Описания управляемого объекта обычно группируют согласно типу административной задачи, устройства или поставщика. Эти описания хранятся в файлах MIB. Они используются сетевой административной системой для мониторинга и управления устройствами, сервисами, задачами и пр. Например, AtomMind Network Manager оснащен набором наиболее важных встроенных MIB. Специфичные для поставщика или задачи MIB можно добавить для расширения функциональности AtomMind Network Manager. Дополнительную информацию об управлении файлами MIB в <%AGNM% см. в разделах Управление файлами SNMP 1841.

В SNMP набор соответствующих переменных можно сгруппировать вместе, чтобы сформировать более крупные структуры, представленные в виде таблиц. Таким образом, объекты могут содержать значения двух видов: скалярные величины или таблицы. У скалярных величин одно значение. Например, `iso.org.dod.internet.mgmt.mib-2.interfaces.ifNumber` (`1.3.6.1.2.1.2.1`) имеет скалярную величину, которая представляет собой итоговое число интерфейсов (портов), доступных на сетевом устройстве. Таблицы представляют несколько записей с идентичной структурой. Например, переменная `iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable` (`1.3.6.1.2.1.2.2`) содержит список входных интерфейсов, каждый из которых содержит информацию об отдельном сетевом интерфейсе, например, поток отражения, физический адрес, пропускную способность и пр.

КОДИРОВАНИЕ

Управляемые объекты кодируются в восьмибитовый поток для передачи по сети. Алгоритмы кодирования и декодирования, используемые в SNMP, описаны в разделе *Основные правила кодирования* (BER).

19.1.6.1.3 Операции SNMP

Стандарты SNMP определяют набор операций, которые может вызвать администратор или агент для того, чтобы выполнить административную задачу или мониторинг. Все функции, которые предоставляют агенты, смоделированы в SNMP как операции чтения и записи, применяемые к переменным агента.

Операции чтения и записи

Существует несколько операций SNMP, определенных стандартами. Мы сгруппируем эти операции в две главные категории, согласно основным административным операциям: чтения (мониторинга) и записи (управления).

ОПЕРАЦИИ ЧТЕНИЯ

Отслеживание задач можно выполнить при помощи операций чтения. Сюда включены операции опроса, *инициируемые* администратором, и *предупреждения без запроса* (события), отправленные агентами. Операции опроса выполняются как различные *операции получения* (например, `GetRequest`, `GetNextRequest`, `GetBulkRequest`), в то время как события представлены операциями `Trap` и `InformRequest`.

ОПЕРАЦИИ ЗАПИСИ

Операции управления выполняются как операции изменения, т.е. операции записи переменных SNMP, предоставленных управляемым агентом. Так, операции `SetRequest` используются для удаленной модификации переменных агента.

Подтвержденные и не подтвержденные операции

Протокол SNMP предназначен для поддержки подтвержденных и не подтвержденных операций. Например, `Trap` - это не подтвержденная операция, в то время как `InformRequest` - подтвержденная.

Подтвержденные операции были представлены для установления механизма надежной передачи и обработки данных SNMP. Например, администратор не обязательно доставляет и обрабатывает ловушки. Даже такой надежный протокол, как TCP, не дает в этом случае гарантию потому, что административное приложение может "рухнуть" во время принятия и/или обработки данных. С подтвержденной операцией, получающий движок SNMP подтверждает получение данных. Например, ответ на операцию протокола `InformRequest` указывает, что было получено уведомление, передана модель защиты и обработана администратором. Аналогичным образом, ответ на `SetRequest` указывает, что запрос о записи агент действительно обработал.

19.1.6.1.4 Безопасность SNMP

Безопасность SNMP основана на так называемых *Строках доступа*, которые играют роль пароля. Администраторы и агенты используют строки доступа для авторизации перед выполнением различных операций. Существуют три строки доступа, контролирующие различные виды деятельности:

- **Строку доступа Только для чтения** использует агент для авторизации администратора перед выполнением операций чтения.

- **Строку доступа Только для записи** использует агент для авторизации администратора перед чтением и изменением значений данных.
- **Строку доступа ловушки** использует администратор, когда необходимо решить, можно ли доверять уведомлению о событии.

Основная проблема безопасности SNMPv1 и SNMPv2 заключается в том, что строки доступа передаются как текст, что делает управляемую сеть слабозащищенной для третьей стороны. SNMPv3 представляет кодировку строк доступа и, таким образом, в значительной степени усиливает безопасность административных операций. Хотя SNMPv3 делает сеть безопаснее, она все же может подвергаться различным атакам, поэтому следует внимательно отнестись к вопросу безопасности.

Мы рекомендуем рассмотреть возможность использования SNMPv3 с целью обеспечения безопасности SNMP в Вашей сети. В частности, следует принять во внимание и использовать Модель безопасности пользователя (USM) для версии 3 Простого протокола сетевого управления (SNMPv3) и Модель управления доступом просмотра (VACM) для Простого протокола сетевого управления (SNMP).

19.1.6.2 Использование SNMP в AtomMind Network Manager

AtomMind предоставляет мониторинг пользовательских устройств SNMP, предлагая возможности для:

- Связи с устройствами SNMP при помощи **драйвера устройства Простой Сетевой Протокол**
- Получение ловушек SNMP из сетевых устройств
- Редактирование информационной базы управления 1841 (добавление, удаление и изменение пользовательских MIB файлов при необходимости)
- Обработка любых пользовательских данных, полученных из устройств SNMP и их представление согласно требованиям универсальных средств AtomMind

AtomMind запрашивает полную информацию обо всех доступных объектах SNMP (идентификаторах объектов) через процедуру *SNMP Walk* (т.е. множественные вызовы `GetNextRequest`). Необходимые значения периодически прочитываются (см. также Синхронизацию 514) и все полученные данные кэшируются (см также Кэш параметров 502).

Период синхронизации по умолчанию для устройств SNMP - 24 часа. Некоторые объекты опрашиваются за более короткий период, например, сетевой интерфейс и статистические данные процесса прочитываются каждые 15 и 30 секунд.

AtomMind Network Manager интенсивно использует SNMP для управления элементами сети. Это подразумевает выполнение нескольких задач:

- Настройка 1832 менеджеров и агентов с целью использования связей SNMP.
- Мониторинг 1845 сетевых элементов через чтение значений переменных SNMP и прослушивание административных событий, отправленных как SNMP-ловушки.
- Подсчет характеристик сетевых объектов, представляющих их в доступной форме, анализ, определение, проблемы, принятие решений с использованием собранных данных SNMP 1847.
- Контролирование 1847 настройки и состояния управляемых объектов путем изменения значений переменных SNMP, которые они предоставляют.
- Генерирование SNMP-ловушек 1848.
- Обнаружение устройств и сервисов, где SNMP используется для получения данных от маршрутизаторов и коммутаторов, а также для обнаружения агентов SNMP.

19.1.6.2.1 Настройка SNMP

Как настроить мониторинг и управление SNMP? Поскольку процесс управления состоит из двух частей: менеджер и управляемый объект (агент), обе стороны должны быть правильно настроены для связи через SNMP. Соответственно, эта часть содержит два раздела - Настройка агентов SNMP 1832, и Установка AtomMind Network Manager 1839.

19.1.6.2.1.1 Настройка SNMP-агентов

Эта глава представляет собой обзор аспектов настройки агентов SNMP и предоставляет подробные инструкции для настройки агентов SNMP на некоторых сетевых системах.

Настройка агентов

Типичный агент SNMP реализуется как компонент программного обеспечения, запускаемый на сетевом устройстве. Компонент программного обеспечения следует правильно установить и настроить. Процедура установки/настройки во многом зависит от типа сетевой системы и среды программного обеспечения агента. Информацию о том, как установить SNMP на том или ином типе сетевой системы см в соответствующем разделе.

Для настройки необходима следующая информация:

- **Имена доступа** (community) в Вашей сети
- **Назначение ловушек** для каждого доступа
- **IP-адреса и имена ПК** для администрируемых по SNMP хостов

Дополнительные инструкции по настройке агентов SNMP можно найти в следующих главах:

- Системы Windows 1833
- Системы Linux 1836
- Системы Solaris 1837
- Устройства Cisco 1838
- Сервера Microsoft SQL 1838
- Сервера Lotus Domino 1839
- Сервера Oracle 1839

Общие настройки

Все устройства SNMP имеют следующие общие параметры настройки:

Параметры	Описание
sysLocation	Физическое месторасположение отслеживаемого устройства.
sysContact	Определяет главное контактное лицо для устройства.
sysName	Должно быть установлено в полное доменное имя (FQDN) управляемого устройства. Иными словами, это имя хоста, ассоциируемое с IP-адресом управляемого устройства.
Read-only access community string	Строка доступа в режиме только для чтения используется для получения доступа к запрашиваемой административной информации от агента SNMP.
Read-write access community string	Используя строку доступа с режиме записи/чтения, можно изменять переменные MIB на сетевом элементе.
Trap community string	Строка доступа ловушки будет включена в отправляемые устройством ловушки, и менеджеры ловушек могут использовать ее, когда необходимо определить, нужно ли обрабатывать полученную ловушку.
Trap destination	Адреса, на которые отправляются ловушки.

У устройств могут быть разные варианты доступа и параметры ловушки. Например, устройства Cisco поддерживают различные строки доступа для разных частей MIB, что разрешает отдельным группам переменных иметь мелкоструктурный контроль за доступом. Многие производители разрешают устанавливать ограничения на хостах, согласно которым разрешено выполнять запросы SNMP, тем самым предоставляя другой уровень безопасности, в дополнение к строке доступа.

Существует группа опций настройки, которые могут встретиться во время управления сетевыми системами разных производителей. Следует обратиться к руководству для Вашего устройства/программного обеспечения или иной доступной документации, например, к RFC.

Вопросы безопасности

Не забудьте изменить строки доступа, настроенные по умолчанию, на значения с высоким уровнем безопасности. Не следует выбирать слова, используйте лучше буквенно-числовую комбинацию. Используйте разные строки для доступа на чтение и запись.

Серьезной проблемой может оказаться отправление строки доступа чтения и записи в виде простого текста через SNMPv1 и v2. Таким образом, строки доступа оказываются потенциально доступны каждому с доступом к анализатору пакетов, т.е. почти каждому пользователю Вашей сети с ПК и общедоступным программным обеспечением.

Вы можете установить органичения для устройств, которые могут выполнять запросы SNMP, если Ваш агент это поддерживает. Таким образом, даже если кто-то получает строку доступа, ему придется подделать IP-адрес одной из Ваших станций управления, чтобы нанести какой-либо вред. Это мера позволяет снизить риск, но не дает полную гарантию безопасности. Лучшим решением будет обезопасить пакеты SNMP, сделав их невидимыми для внешней стороны сегмента сети управления. Для этого следует настроить маршрутизаторы и firewall должным образом. К сожалению, не всегда оказывается возможным установить отдельную административную сеть, или использовать ее из разных мест. Чтобы сделать административный трафик приватным, следует использовать решение VPN или какую-то из форм туннелирования.

Многие устройства могут генерировать ловушки ошибки авторизации, когда кто-то пытается получить к ним доступ, используя неправильные строки доступа. Если Ваше устройство поддерживает это свойство, используйте его для выявления попытки нежелательного доступа к Вашему устройству.

SNMPv3 позволяет решить большинство вопросов безопасности. В частности, он гарантирует, что все строки доступа закодированы.

Чтобы правильно установить *Простой протокол сетевого управления* (*SNMP*) на системе **Windows,** следует**:**

1. Включить агент 1833 SNMP 1833 (если он еще не включен)
2. Настроить агент 1834 SNMP 1834 так, как Вам это необходимо

Перейдите в нужный подраздел, если требуется включить *SNMP* на:

- Windows Server 2008 1833
- Windows Vista или Seven 1835
- Windows XP, 2000 или 2003 1834
- Windows NT 1834
- Windows 98 1834

У Вас должны быть **привилегии администратора**, чтобы включить SNMP на системе Windows.

Установить SNMP на Windows Server 2008

Вы можете установить сервис SNMP на **Windows Server 2008,** используя один из следующих вариантов**:**

- использовать встраиваемый модуль Менеджера Сервера для добавления свойства сервиса **SNMP**
- или использовать следующую команду:
  ```
  start /w ocsetup SNMP-SC
  ```

Этапы установки SNMP на Windows Vista и Windows 7

1. В **Панели Управления** выберите **Программы**
2. Под элементами **Программы и Свойства**, выберите **включить или выключить свойства Windows**
3. В списке свойств для **Windows** выберите **свойства SNMP** и раскройте список так, чтобы Вы смогли увидеть **Провайдера WMI SNMP**
4. *Включите флажок* для **Провайдера WMI SNMP**. Флажок для **свойств SNMP** устанавливается автоматически потому, что провайдеру необходим SNMP

5. Кликните **OK**
6. Из **командной строки** или **меню Пуск** запустите **Services.msc** и убедитесь, что сервис SNMP запущен.

Этапы установки SNMP на Windows XP, 2000 или 2003

1. Кликните **Пуск** выберите **Настройки**, кликните по **Панели управления**, дважды кликните по **Добавить или удалить программы**, а затем кликните **Добавить/удалить компоненты Windows**
2. Выберите **Компоненты**, кликните по **Инструменты Управления и Мониторинга** (не изменяя состояние флажка!) и затем кликните по элементу **Детали**.
3. Установите флажок для **Простого протокола сетевого управления** и кликните **OK**
4. Кликните **Продолжить**
5. Вставьте CD-диск или укажите полный путь к необходимым файлам.

Процесс установки теперь завершен. Помимо этого автоматически устанавливается БД управления ресурсами хост-системы. Для настройки агентов SNMP для ответа на запросы SNMP, обратитесь к разделу **Настройка** агентов SNMP.

Этапы установки SNMP на Windows NT

1. Правой кнопкой мыши кликните по иконке **Сетевое окружение** на *Рабочем столе*
2. Кликните **Свойства**
3. Кликните **Сервисы**
4. Кликните **Добавить**. Появится диалоговое окно **Выберите Сетевой сервис**
5. В списке **Сетевых сервисов**, кликните по сервису **SNMP,** а затем кликните **OK**
6. Вставьте CD-диск или задайте полный путь к необходимым файлам и кликните **Продолжить**
7. После того, как все необходимые файлы скопированы, появится диалоговое окно **Свойств Microsoft SNMP**

Процесс установки завершен. Помимо этого автоматически устанавливается БД управления ресурсами хост-системы. Чтобы настроить *SNMP-агенты* так, чтобы они отвечали на запросы SNMP, изучите раздел **Настройка** агентов SNMP.

Этапы настройки SNMP на Windows 98

1. Вставьте CD-диск *Windows 98*
2. На панели **управления сетью**, кликните **Добавить**. Появится диалоговое окно **Выбрать тип сетевого компонента**
3. Дважды щелкните по элементу **Сервис**
4. В диалоговом окне **Выберите сервис сети** кликните по элементу **Имеется диск**
5. Укажите путь к директории `tools\reskit\netadmin\snmp` на Вашем CD в диалоговом окне **Установить с диска**, а затем кликните **OK**
6. Выберите **агент Microsoft SNMP** из списка *Модели* в диалоговом окне *Выбрать сервис сети*, а затем кликните **OK**

Процесс установки завершен. Помимо этого автоматически устанавливается БД управления ресурсами хост-системы. Чтобы настроить *SNMP-агенты* так, чтобы они отвечали на запросы SNMP, изучите раздел **Настройка** агентов SNMP.

Выберите нужный подраздел, если требуется включить SNMP на:

- Windows XP, 2000, Server 2003, Server 2008, Vista и Seven
- Windows **NT**

Информация о том, как активизировать агенты SNMP на системах Windows содержится в разделе Включение агентов SNMP на системах Windows.

Настройка агента SNMP на Windows XP, 2000, Server 2003, Server 2008, Vista и Seven

ЭТАПЫ НАСТРОЙКИ АГЕНТА SNMP

1. Кликните **Старт**, выберите **Настройки**, кликните по **Панели управления**.
2. Под **Средствами Администрирования**, щелкните по элементу **Сервисы**.
3. В панели **Детали**, щелкните правой кнопкой мыши по **сервису SNMP и** выберите **Свойства**.
4. Во вкладке **Безопасность**, выберите **Отправлять ловушку авторизации**, если Вы хотите, чтобы сообщение ловушки отправлялось всегда при неудачной попытке авторизации.
5. Под элементом **Принятые строки доступа**, кликните **Добавить**.
6. Под элементов **Права доступа**, выберите *права доступа* для данного хоста для обработки запросов SNMP из выбранного сообщества.
7. В элементе имя доступа, введите *имя доступа* (чувствительно к регистру), а затем кликните **Добавить**.
8. Выберите, нужно ли принимать пакеты SNMP от хоста:

Чтобы принимать запросы SNMP от любого хоста в сети, независимо от ID, щелкните по элементу **Принимать пакеты SNMP** с любого хоста.

Чтобы ограничить принятие пакетов SNMP, кликните **Принимать пакеты SNMP с этих хостов**, затем кликните **Добавить** и введите соответствующие имена хостов, IP- или IPX-адреса, а затем вновь кликните **Добавить**.

9. Кликните **Применить**, чтобы применить все изменения.

ЭТАПЫ НАСТРОЙКИ ЛОВУШЕК SNMP

1. Кликните **Старт**, выберите **Настройки**, кликните по **Панели управления**.
2. Под **Средствами Администрирования**, щелкните по элементу **Сервисы.**
3. В панели **Детали**, щелкните правой кнопкой мыши по **сервису SNMP и** выберите **Свойства**.
4. В закладке **Ловушки**, под **именем сообщества**, введите *имя сообщества* (чувствительно к регистру), которому компьютер будет отправлять сообщения ловушки, а затем кликните **Добавить** в список.
5. Под элементом **Место назначение ловушки**, кликните **Добавить**.
6. В поле **имя хоста, IP-** или **IPX-адрес**, введите *имя хоста или его IP-адрес* сервера AtomMind Server , куда отправлять ловушку, а затем щелкните **Добавить**.
7. Повторите этапы с 4 по 6 до тех пор, пока не добавите все необходимые доступы и места назначений ловушек.
8. Щелкните **ОК**, чтобы применить изменения.

Настройка агента SNMP на Windows NT

ЭТАПЫ НАСТРОЙКИ АГЕНТА SNMP

1. Кликните **Старт**, выберите **Настройки**, кликните по **Панели управления**.
2. Под **Средствами Администрирования**, щелкните по элементу **Сервисы.**
3. В панели **Детали**, щелкните правой кнопкой мыши по **сервису SNMP и** выберите **Свойства**.
4. Во вкладке **Безопасность**, выберите **Отправлять ловушку авторизации**, если Вы хотите, чтобы сообщение ловушки отправлялось всегда при неудачной попытке авторизации.
5. Под элементом **Принятые строки доступа**, кликните **Добавить**.
6. В ячейке **Имена сообществ**, введите имена доступа для авторизации запросов SNMP.
7. Чтобы внести имя в список **Принятых имен доступа**, кликните **Добавить**.
8. Повторите этапы с 5 по 7 для каждого из имен доступа.
9. Чтобы определить, принимать ли пакеты SNMP от любого хоста или только от определенных, кликните по одной из двух опций:

Принимать пакеты SNMP от любого хоста, если не нужно отвергать никакие из пакетов SNMP на основании идентификатора компьютера-источника.

Принимать пакеты SNMP только от этих хостов, если пакеты SNMP должны приниматься только с перечисленных компьютеров. Чтобы назначить определенные хосты, кликните **Добавить**, введите *имя* или *адрес хоста*, с которого Вы будете получать запросы по **IP или IPX**, а затем кликните **Добавить (**повторите этот это действие для каждого из добавляемых хостов)

10. Во вкладке **Агент**, задайте нужную информацию (такую как комментарии о пользователе, месторасположении и сервисах).

11. Кликните **OK,** чтобы применить изменения.

ЭТАПЫ НАСТРОЙКИ ЛОВУШЕК SNMP

1. Кликните **Старт**, выберите **Настройки**, кликните по **Панели управления**. Под **Средствами Администрирования**, щелкните по элементу **Сервисы.**
2. В панели **Детали**, щелкните правой кнопкой мыши по **сервису SNMP и** выберите **Свойства**.
3. Кликните по вкладке **Ловушки**.
4. Введите *строку доступа* (чувствительно к регистру) в ячейке **Имя доступа** для каждого доступа, на который компьютер должен оправлять ловушки.
5. После ввода каждого имени, кликните **Добавить**, чтобы добавить имя в список.
6. Чтобы задать хосты для каждого доступа, на который Вы отправляете ловушки, после добавления доступа (когда он все еще выделен), щелкните по **Добавить** под элементом **Местоназначение ловушки**.
7. Введите имя хоста или адрес в ячейку **Хост IP/Адрес**, а затем кликните **Добавить**, чтобы переместить в список **место назначения Ловушки** для выбранного доступа.
8. Повторите этап 7 для каждого добавляемого хоста.
9. Кликните **OK**, чтобы применить все изменения.

Чтобы правильно установить *Простой протокол сетевого управления* (*SNMP*) на Вашей системе **Linux**, следует:

1. Активировать 1833 агент SNMP 1836 (если он еще не включен)
2. Настроить 1837 агент SNMP 1834 согласно Вашим требованиям

Обратитесь к соответствующему подразделу, если требуется активировать *SNMP-агент* на системе *Linux* двумя способами*:*

- Используя пакет RPM 1838
- Используя ZIP-архив

Установить, используя RPM

1. Загрузите последнюю *rpm-версию* SNMP из http://prdownloads.sourceforge.net/net-snmp/net-snmp-5.1.1-1.rh9.i686.rpm?download
2. Зарегистрируйтесь как *root*-пользователь.
3. Перед установкой новой версии *net-snmp*, Вам потребуется удалить ранние версии *net-snmp* со своего компьютера. Чтобы включить в список версии *net-snmp* , установленные на Вашем компьютере, выполните следующую команду:
   ```
   rpm -qa | grep "net-snmp"
   ```
4. Если на Вашем компьютере уже есть установленные версии, удалите их при помощи следующей команды:
   ```
   rpm -e <полное имя SNMP-агента, выданное предыдущей командой> --nodeps
   ```
5. Если на Вашем компьютере нет ранее установленных версии, тогда выполните следующую команду, чтобы установить новую версию:
   ```
   rpm -i <загруженная версия SNMP-агента> --nodeps
   ```

Процесс установки завершен. Если необходимо настроить SNMP-агенты так, чтобы они отвечали на запросы SNMP, обратитесь к разделу Настройка SNMP-агентов 1837.

Установить, используя ZIP

1. Загрузите zip-версию SNMP из https://sourceforge.net/projects/net-snmp/files/ucd-snmp/
2. Извлеките файл, используя следующую команду:
   ```
   tar -zxvf ucd-snmp-4.2.6.tar.gz
   ```
3. Зарегистрируйтесь как *root*-пользователь.
4. Выполните команду, чтобв установить путь компилятора C:
   ```
   export PATH=<gcc path>:$PATH
   ```

5. Выполните следующие четыре команды из директории, где Вы извлекли *ucd-snmp* (`directory_name` -- это директория инсталляции SNMP-агента; предпочтительно выбрать директорию под `/root`, поскольку директории `/usr` и `/local` могут содержать файлы старой версии SNMP):
```
./configure --prefix=<directory_name> --with-mib-modules="host"
make
umask 022
make install
```

Процесс установки завершен. Чтобы настроить SNMP-агенты так, чтобы они отвечали на запросы SNMP, обратитесь к разделу Настройка SNMP-агентов 1837.

Если необходимо активировать SNMP-агент на системах Linux, обратитесь к разделу Активирование SNMP-агента на системах Linux 1838.

Чтобы настроить *SNMP-агент,* Вам необходимо изменить файл `snmpd.conf`:

- После строки:
```
# name incl/excl subtree mask(optional)
```
вставьте строку
```
view allview included .1.3.6
```
- Измените строку после:
```
# group context sec.model sec.level prefix read write notif
```
из
```
access notConfigGroup "" any noauth exact systemview none none
```
в
```
access notConfigGroup "" any noauth exact allview none none
```
- Затем перезапустите snmp-агент, используя:
```
/etc/rc.d/init.d/snmpd restart
```

Чтобы правильно установить *Простой протокол сетевого управления* (*SNMP*) на Вашей системе **Solaris,** следует**:**

1) **Активировать** 1837 SNMP-агент 1837 (если он не активирован)

2) **Настроить** 1837 SNMP-агент 1837 так, как это необходимо Вам

Чтобы установить SNMP, следуйте следующим этапам:

1. Загрузите последнюю версию SNMP из https://sourceforge.net/projects/net-snmp/files/ucd-snmp/

2. Извлеките файл, используя следующую команду:
```
tar -zxvf ucd-snmp-4.2.6.tar.gz
```

3. Зарегистрируйте как *root*-пользователь.

4. Выполните команду, чтобы установить путь к компилятору C:
```
export PATH=<gcc path>:$PATH
```

5. Выполните следующие четыре команды из директории, где Вы извлекли *ucd-snmp* (`directory_name` -- это директория для установки SNMP-агента; предпочтетельно выбирать директорию под `/root`, поскольку директории `/usr` и `/local` могут содержать файлы старой версии SNMP):
```
./configure --prefix=<directory_name> --with-mib-modules="host"
make
umask 022
make install
```

Процесс установки завершен. Чтобы настроить SNMP-агенты так, чтобы они отвечали на запросы SNMP, обратитесь к разделу Настройка SNMP-агентов 1837.

Если необходимо активировать SNMP-агент на системах Solaris, обратитесь к разделу Активирование SNMP-агента на системе Solaris 1837.

Чтобы настроить SNMP-агент, выполните следующее:

1. Остановите агент, если он уже запущен и используется
```
/etc/init.d/init.snmpdx stop
```

2. Выполните следующие изменения в файле /etc/init.d/init.snmpdx

Замените строки
```
if [ -f /etc/snmp/conf/snmpdx.rsrc -a -x /usr/lib/snmp/snmpdx ]; then
/usr/lib/snmp/snmpdx -y -c /etc/snmp/conf -d 3 -f 0
fi
```
с
```
<Installation Directory>/sbin/snmpd
```

Замените строку
```
/usr/bin/pkill -9 -x -u 0 '(snmpdx|snmpv2d|mibiisa)'
```
с
```
/usr/bin/pkill -9 -x -u 0 '(snmpd)'
```

3. Перезапустите агент, использующий
   ```
   /etc/init.d/init.snmpdx start
   ```

Чтобы настроить SNMP-агенты на устройствах Cisco, Вам нужно зарегистрироваться на устройстве и переключиться в **режим с расширенными возможностями**. Затем Вы можете запустить следующие команды из командной строки:

Команды для активирования SNMP

```
#configure terminal
#snmp-server community <community_string> [rw/ro]
```
(например: `snmp-server community public ro`)
```
#end
#copy running-config startup-config
```

Команды для активирования ловушки

```
#configure terminal
#snmp-server enable traps snmp authentication
#end
#copy running-config startup-config
```

Команды для установления AtomMind Server в качестве хоста

```
#configure terminal
#snmp-server host <AtomMind Server IP> <Trap community string> snmp
```
(например: `snmp-server host 192.168.9.58 public snmp`)
```
#end
#copy running-config startup-config
```

Убедитесь, что SNMP-агент запущен на сервере. Если нет, обратитесь к разделу Установка SNMP на системах Windows 1833.

Затем запустите сервис **SQLSERVERAGENT,** следуя описанным ниже этапам

Запуск сервиса SQLSERVERAGENT на Windows XP, 2003 или 2000

- Кликните **Пуск**, выберите **Настройки**, а затем кликните по **Панели управления**
- Щелкните дважды по **Средствам администрирования**, а затем кликните дважды по элементу **Управление компьютером**.
- В дереве консоли щелкните **Сервисы и Приложения**, а затем кликните по элементу **Сервисы**
- Кликните правой кнопкой мыши по **SQLSERVERAGENT,** а затем кликните **Запуск**

Запуск сервиса SQLSERVERAGENT на Windows NT

- Щелкните правой кнопкой мыши по иконке **Сетевое окружение** на *Desktop*

- Кликните по элементу **Свойства**
- Кликните по элементу **Сервисы**
- Щелкните правой кнопкой мыши по **SQLSERVERAGENT,** а затем кликните Запуск

SNMP -агент работает аналогично Сервису Windows и запускается автоматически, даже если не запущен Domino. Следует это учитывать при обновлении Domino: Вы должны остановить сервисы LNSNMP и Windows SNMP до начала процесса обновления.

- **Остановите** сервисы LNSNMP и SNMP при помощи следующих команд:
```
net stop lnsnmp
net stop snmp
```
- **Настройте** Агент SNMP Lotus Domino как сервис, используя следующую команду:
```
lnsnmp -Sc
```
- **Запустите** сервисы SNMP и LNSNMP, используя следующие команды:
```
net start snmp
net start lnsnmp
```

Мониторинг сервисов Oracle может быть предоставлен при помощи **Oracle Intelligent Agent**, который может быть настроен так, что сторонние системы могут получать SNMP-ловушки и собирать информацию. Для этого Oracle Intelligent Agent следует настроить так, чтобы он распознавал запросы SNMP, исходящие от главного агента.

Чтобы установить это на ПК с Windows, следует выполнить следующее:

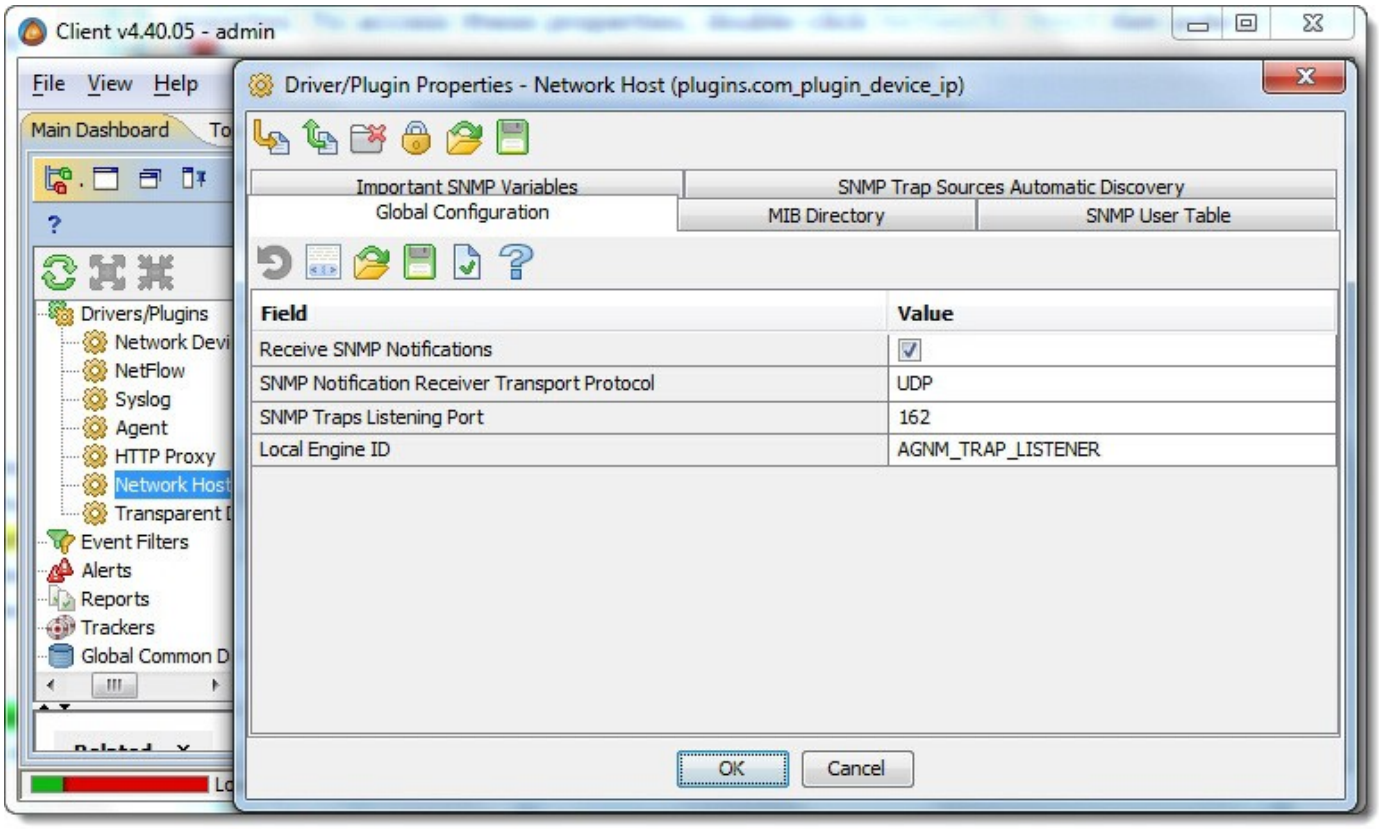
1. После установки и настройки SNMP-агентов на Вашем ПК с Windows, следует интегрировать SNMP с Intelligent agent. Для этого нужно установить **Oracle Peer SNMP Master Agent** и **SNMP Encapsulator Agent** на сервере Oracle. Обратите внимание, что эти агенты должны быть одной и той же версии, что и Intelligent Agent и установлены на одном и том же **ORACLE_HOME**. После завершения установки будут созданы сервисы **Oracle SNMP Peer Encapsulator** и **Oracle Peer SNMP Master Agent**. (Если Вы не устанавливаете программное обеспечение для Intelligent Agent на выбранном по умолчанию $ORACLE_HOME, имена всех сервисов будут начинаться следующим образом: Oracle<home name>.)
2. Для связи главного SNMP-агента со стандартным SNMP-сервисом и Intelligent Agent, файл **SNMP-**сервисов должен быть правильно настроен**.** Вы должны назначить свободный порт для инкапсулированного агента (**Microsoft SNMP Service**). Обычно Microsoft SNMP Service использует порт **1161**, заданный в файле **SERVICES,** который находится в директории **<WINDOWS>\SYSTEM32\DRIVERS\ETC** (где **<WINDOWS>** назначает путь к инсталляции windows). Убедитесь, что у Вас есть следущие строки в файле, изменяющие порт по умолчанию (161) на другой доступный порт (1161 в этом примере):
```
snmp 1161/udp snmp
snmp-trap 1162/udp snmp
```
3. Проверьте, что файлы **HOSTS** и **LMHOSTS.SAM** в **<WINDOWS>\SYSTEM32\DRIVERS\ETC** (где **<WINDOWS>** назначает путь к инсталляции windows) содержит отображение IP-адресов на имена хостов для всех компьютеров в установке SNMP. Это в значительной степени улучшит эффективность работы системы, даже если Вы используете DHCP и WINS.

19.1.6.2.1.2 Настройка SNMP в AtomMind Network Manager

Чтобы настроить AtomMind Network Manager на выполнение административных задач с использованием SNMP, следует выполнить следующее:

- Настроить драйвер хоста IP 1840.
- Предоставить и настроить все необходимые MIB-файлы. См раздел Управление MIB- файлами 1841.
- Добавить и настроить сетевые устройства для SNMP-управления 1842.

В AtomMind Network Manager SNMP поддерживает Драйвер сетевого хоста 593. Настройка SNMP доступна в виде свойств драйвера Сетевого узла. Чтобы получить доступ к этим свойствам, дважды кликните по элементу Сетевой узел под узлом Драйверы/Плагины, расположенным в Системном дереве.



Существует несколько вкладок, которые представляют различные аспекты использования SNMP в AtomMind Network Manager, как описано ниже.

Глобальная конфигурация

Вкладка Глобальная конфигурация содержит свойства SNMP-ловушек, предназначенных для:

- активизация/отключение SNMP-уведомлений;
- транспортный протокол, используемый для получения уведомлений;
- номер порта, используемый для получения уведомлений;
- ID локального обработчика, который используется для обработки уведомлений версии 3 SNMP.

Дополнительная информация содержится в разделах SNMP-мониторинга 598 драйвера сетевого хоста.

MIB-директория

Вкладка Директория MIB содержит список загруженных MIB-файлов и MIB-ошибок компиляции. Эта тема описана в разделе Управление MIB-файлами 1841.

Пользовательская SNMP-таблица

Пользовательская SNMP-таблица содержит список пользователей SNMP и их свойства. Эта таблица используется для предоставления опций безопасности, специфичных для версии 3 SNMP. См. Пользовательские SNMP-таблицы 645, описанные в главе Глобальные настройки SNMP.

Автоматическое обнаружение источников SNMP-ловушек

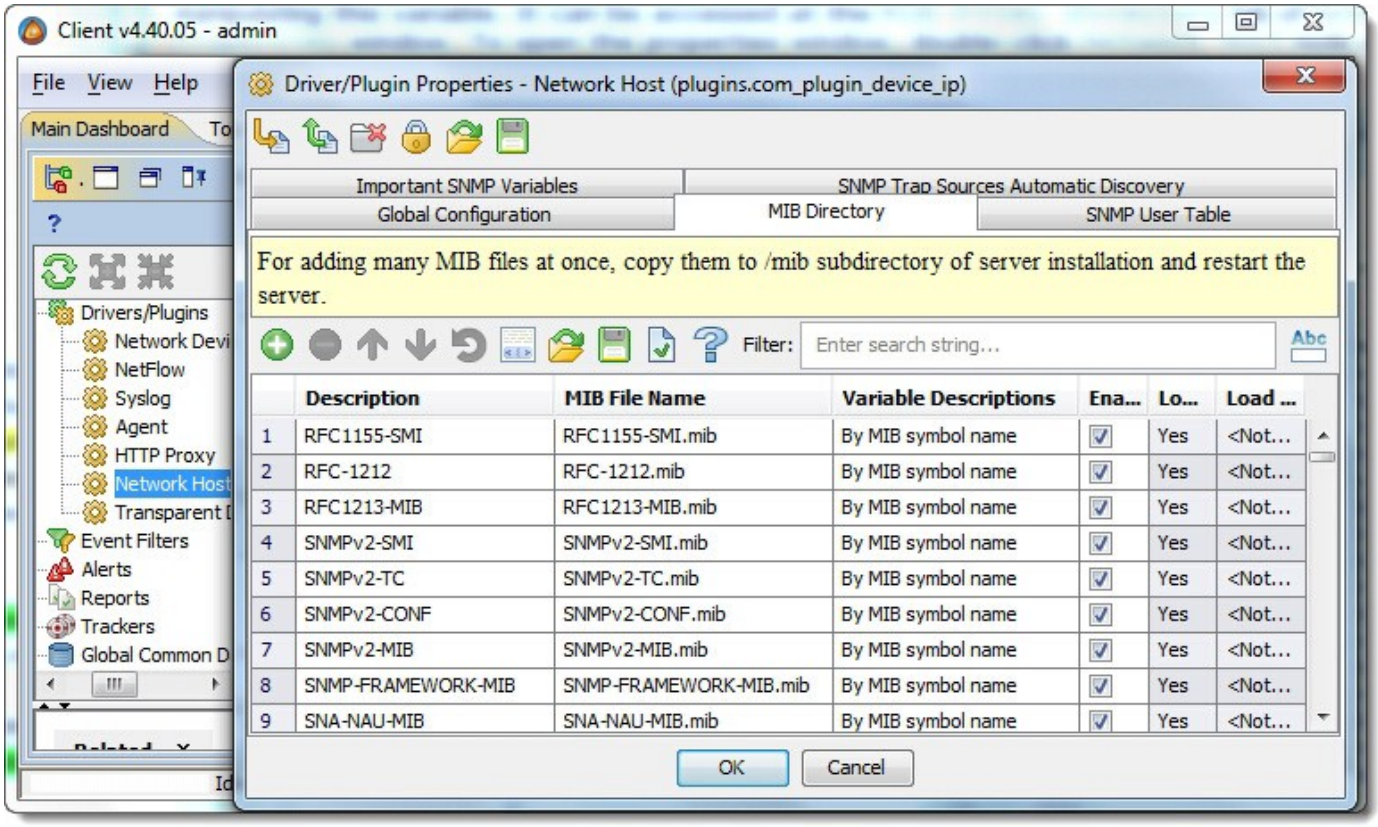
Когда AtomMind Network Manager получает SNMP-ловушку, он может автоматически обнаружить устройство, отправившее это уведомление (См. раздел Автоматическое обнаружение источников SNMP-ловушек 1847). Это свойство контролирует настройка Включить автообнаружение источника SNMP-ловушек.

MIB-файлы описывают части административной информации, которую предоставляют агенты. Загруженные MIB-файлы и скомпилированный AtomMind Network Manager формируют **настройку MIB**, которая используется для интерпретации данных, полученных от SNMP-агентов. Если полученные данные не описаны должным образом, менеджер не сможет их обработать. Такие данные будут отображаться в "сырой" форме. Таким образом, полная и правильная настройка MIB чрезвычайно важна для успешного сетевого мониторинга.

Существуют тысячи MIB-файлов, доступных для различных сетевых ресурсов. Они периодически обновляются, приобретая новую функциональность и избавляясь от погрешностей и дефектов. Полномасштабная функциональность настройки MIB, позволяющая добавлять и использовать MIB производителей, важна для решений по управлению сетями.

Директория MIB-файлов

В AtomMind настройка MIB поддерживается Драйвером сетевого хоста 593. MIB-файлы, их статусы и соответствующие опции представлены в виде записей табличной переменной в Директории MIB-файлов 644 (`mibDirectory`). Пользователи могут управлять настройками MIB, изменяя эту переменную. Получить доступ к ней можно через вкладку `Директории файлов MIB` окна `Свойства драйвера сетевых узлов`. Чтобы открыть окно со свойствами, дважды кликните по узлу `Сетевой узел` под элементом `Драйверы/Плагины` , который располагается в *Системном дереве*.



ПОРЯДОК ЗАГРУЗКИ

Определения MIB-файлов могут ссылаться на определения из других MIB -файлов. Соответственно, MIB-файл может зависеть от другого MIB в директории. Таким образом, порядок загрузки MIB-файлов очень важен. Неправильный порядок может во многом замедлить загрузку MIB и привести к ошибкам компиляции, вызванным неразрешенными ссылками.

AtomMind Network Manager загружает MIB-файлы в том же самом порядке, как они появляются в таблице `MIB Files Directory`. Если Вы хотите изменить порядок, переместите строки вверх или вниз.

ДОБАВЛЕНИЕ MIB-ФАЙЛОВ

Можно добавить MIB-файл в настройку, создав новую запись в директории файлов MIB. Заполните поля с именем файла и его описанием, и выберите для MIB метод описания переменных SNMP. Кроме того, можно скопировать один или несколько MIB-файлов в поддиректорию `/mib` инсталляции AtomMind Server. В обоих случаях определения из MIB-файла будет доступно после перезапуска сервера. Статус MIB-файлов и ошибки загрузки доступны в соответствующих полях записи MIB-директории.

ДЕ-АКТИВИРОВАНИЕ И УДАЛЕНИЕ MIB-ФАЙЛОВ

Если MIB-файлы следует исключить из настроек сервера, их можно де-активировать, если убрать галочку в поле **Включен**. MIB-файл останется в директории, и его можно будет вновь активировать.

Если же необходимо удалить MIB-файл из директории, следует:

- Выбрать элемент **Удалить MIB-файл** в контекстном меню **Сетевого узла** под элементом **Драйверы/Плагины** в *Системном дереве*. Затем необходимо выбрать MIB-файл, который Вы хотите удалить с диска и директории MIB.
- Щелкните правой кнопкой мыши по MIB-файлу во время просмотра списка файлов MIB. Выберите в контекстном меню **Удалить MIB-файл.**

РЕДАКТИРОВАНИЕ MIB-ФАЙЛОВ

MIB-файл из директории можно редактировать непосредственно в AtomMind Network Manager. MIB-редактор поддерживает функцию подсветки синтаксиса. Существуют два способа, как начать редактирование:

- Выберите элемент **Редактировать MIB-файл** в контекстном меню **Сетевого узла** под элементом **Драйверы/Плагины** в *Системном дереве*. Затем выберите MIB-файл, который необходимо отредактировать, из раскрывающегося списка.
- Щелкните правой кнопкой мыши по MIB-файлу во время просмотра списка файлов MIB. Выберите в контекстном меню **Редактировать MIB-файл**.

В этом разделе описаны доступные опции, а также упомянуты ключевые моменты, которые следует учитывать при настройке отдельных устройств для управления по SNMP.

Активирование SNMP

Убедитесь, что SNMP активирован на устройстве, которым Вы планируете управлять. SNMP активируется по умолчанию для добавляемых вручную устройств. Для обнаруженных устройств SNMP активируется или де-активируется автоматически согласно результатам сканирования сервиса.

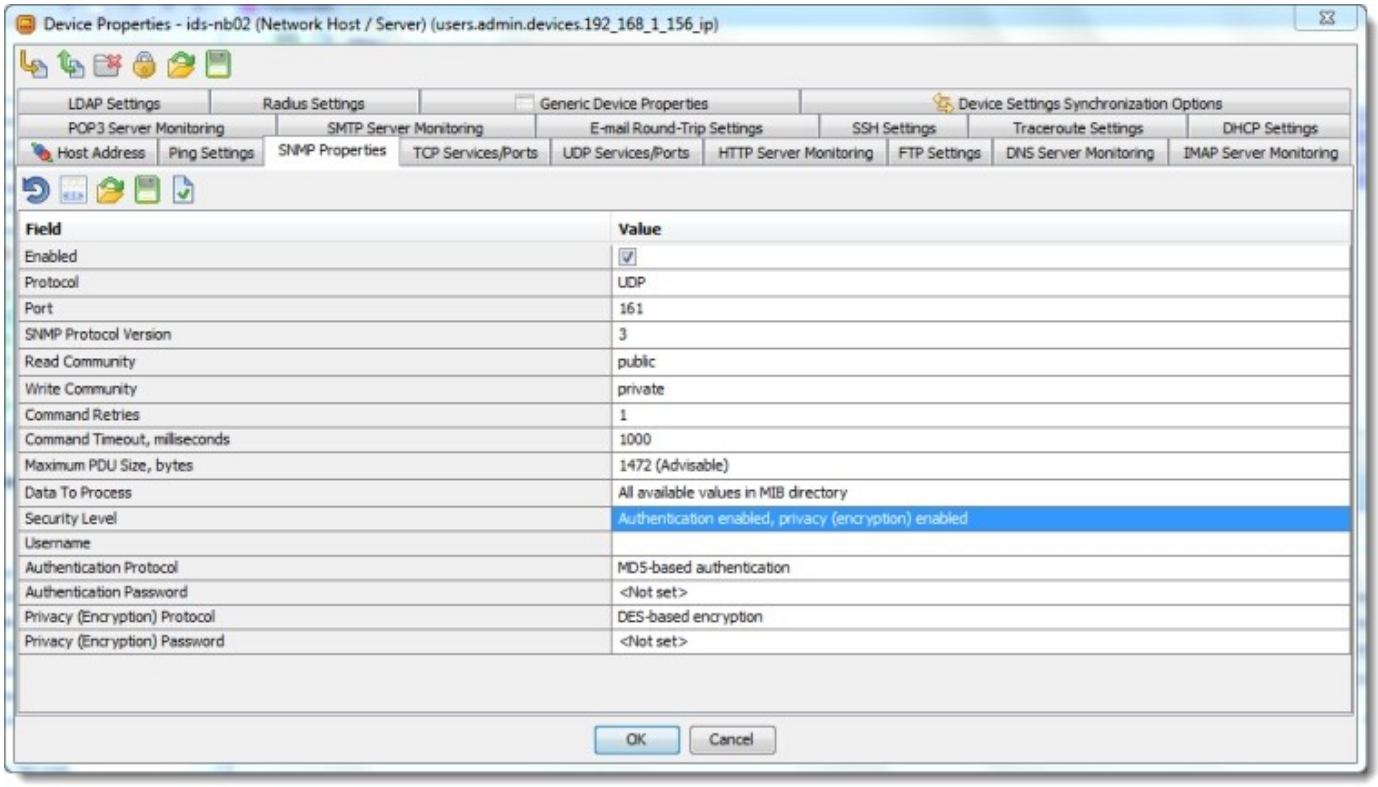
Если SNMP деактивирован для обнаруженного устройства, или активированный SNMP-сервис находится в режиме "offline", попробуйте выполнить слудующее:

1. Убедитесь, что поле `IP-адрес или Имя узла` во вкладке `Адрес узла` заполнено правильно.
2. Убедитесь, что устройство действительно совместимо с SNMP.
3. Убедитесь, что антивирусная программа и/или firewall не мешают связи по SNMP.
4. Перепроверьте выбранные Вами параметры для SNMP: транспортный протокол, строки доступа и пр. (см. дополнительную информацию ниже)

Будучи уверенным, что все параметры верны, активируйте SNMP-сервис, если он не активирован, или же используйте синхронизацию. Следует учитывать, что для полной синхронизации с SNMP-устройством может потребоваться значительный объем времени. Кроме того, Вы можете повторить попытку обнаружения SNMP-сервиса на этом устройстве.

Настройка SNMP-устройства

Параметры SNMP-устройства представлены переменной `SNMP Properties` в контекстах устройства Сетевого хоста (См. Свойства связи по SNMP 640).



Свойства SNMP используются для установления связи с SNMP-агентом. Таким образом, обе стороны должны быть настроены идентично.

Свойства SNMP должны согласовываться с настройкой агента для успешной установки с ним связи.

ВЫБОР ТРАНСПОРТНОГО ПРОТОКОЛА

SNMP может работать по двум транспортным протоколам: UDP (Протокол UDP) или TCP (Протокол TCP). UDP - это протокол *без организации соединения*, т.е. он пытается установить между агентом и менеджером передачу на основе дайтаграмм, и не надежный, т.е. получение сообщений не гарантируется, и порядок сообщений может меняться. TCP, напротив, гарантирует доставку сообщений и предоставляет устойчивое к сбоям восстановление связи после ошибки; сообщения доставляются в соответствии с той последовательностью, с которой они отправлены.

AtomMind Network Manager поддерживает оба протокола UDP и TCP, как транспортные протоколы для SNMP. Какой из них Вы выберите для своей сети?

SNMP через UDP оказывает меньшее воздействие на работу сети и не пытается ретранслировать утраченную или испорченную информацию. UDP - транспортный протокол, выбираемый по умолчанию для SNMP; в частности он подходит для проблематичных сетей. (Обратите внимание, что SNMP изначально предназначался для передачи по UDP.)

Ненадежность протокола UDP можно сбалансировать периодическими запросами, и/или способностью AtomMind Network Manager повторять отправку запросов, если ответ не получен в течение определенного периода времени (см ниже). Но это не работает для ловушек, поскольку они являются, как правило, операциями без подтверждения. Поэтому, если ловушка не была отправлена от агента менеджеру по какой-либо причине, менеджер не узнает, что она была отправлена; а агент не будет об этом проинформирован, и т.о., не сможет повторить отправку ловушки. Эту проблему следует принять во внимание при настройке управления по SNMP.

Используйте информационные запросы вместо ловушек для обеспечения надежной доставки.

SNMP по TCP в основном предназначен для поддержки более эффективных механизмов групповой пересылки данных. Возможно обмениваться множеством пар SNMP-запросов/ответов по одному (устойчивому) соединению по TCP, даже отправлять множество SNMP-сообщений агенту до получения ответов. SNMP по TCP предназначен для случаев, когда размер передаваемых данных большой. Это тот случай, когда TCP может оказаться эффективнее, чем UDP. Однако, в целом, в целях повышения эффективности при использования TCP в качестве транспортного протокола для SNMP-опроса необходима правильная настройка IP-уровня сети, учитывающая условия используемой среды.

Еще один важный момент - надежность. SNMP по TCP обеспечивает надежный обмен SNMP-сообщения между SNMP-обработчиками. В частности, TCP гарантирует (при отсутствии атак), что доставленные данные не повреждены, потеряны, а порядок их доставки не нарушен. В то же время эти свойства TCP не гарантируют, что отправленные данные будут получены и обработаны менеджером. Для этого следует использовать подтвержденные операции.

Выбор UDP в пользу TCP и наоборот не имеет большого значения для обеспечения безопасности, поскольку оба являются слабозащищенными: например, UDP подвержен атакам IP-подмены, в то время как TCP может быть уязвим при атаках типа "отказ в обслуживании". В обоих случаях, следует рассмотреть другие опции безопасности 1830.

ПОРТ

По умолчанию SNMP использует порт 161 для отправки и получения запросов. Если у устройства нестандартная конфигурация SNMP, можно выбрать другой номер порта.

ВЕРСИЯ ПРОТОКОЛА SNMP

AtomMind Network Manager поддерживает SNMPv1, SNMPv2c и SNMPv3. Выберите для устройства подходящий.

ИМЯ СООБЩЕСТВА

Не забудьте правильно установить свойства `Read Community` и `Write Community`, чтобы получить доступ к чтению и записи административной информации устройства через SNMPv1/SNMPv2c.

КОЛИЧЕСТВО ПОВТОРОВ И ТАЙМАУТ

Если ответ на запрос AtomMind Network Managera не поступает в течение заданного периода времени, он повторяет запрос. Эти настройки особенно важны, когда UDP используется в качестве транспортного протокола, поскольку он не надежен, и SNMP-сообщения могут потеряться.

Конкретные значения зависят от параметров сети. Их можно выбрать эмпирическим путем, начиная со значений по умолчанию.

МАКСИМАЛЬНЫЙ РАЗМЕР PDU, БАЙТ

Это максимальный размер SNMP-сообщения, которое может получить агент. Его использует AtomMind Network Manager, чтобы ограничить размер SNMP-запросов.

Рекомендуемый размер 1472 октетов (см, например, Преобразование транспортного протокола для SNMP). Но некоторые агенты не способны обрабатывать сообщения, размер которых превышает 484 октетов, что является "гарантированным" значением.

ОБРАБАТЫВАЕМЫЕ ДАННЫЕ

Многие SNMP-агенты отображают множество SNMP-переменных. Не все из этих переменных используются. Чтение, обработка и хранение неиспользуемых переменных -- это использование сети, процессора и ресурсов памяти впустую. Очевидно, что нет смысла читать все доступные переменные, а доступ осуществляется лишь к тем, которые необходимы для отслеживания отдельного объекта. То, что является полезным на самом деле, зависит от задач и целей пользователя. Чтобы отвечать различным пользовательским требованиям, AtomMind Network Manager предоставляет две опции:

- AtomMind Network Manager может читать, обрабатывать и хранить все значения, которые показывает SNMP-агент. Эту опция, например, можно использовать для изучения любого неизвестного устройства.
- Другой вариант -- обработка только тех значений, которые описаны загруженными MIB-файлами 644, и игнорирование нераспознанных. Это может пригодиться в том случае, когда требуется подробная информация об отслеживаемом устройстве.

УРОВЕНЬ БЕЗОПАСНОСТИ

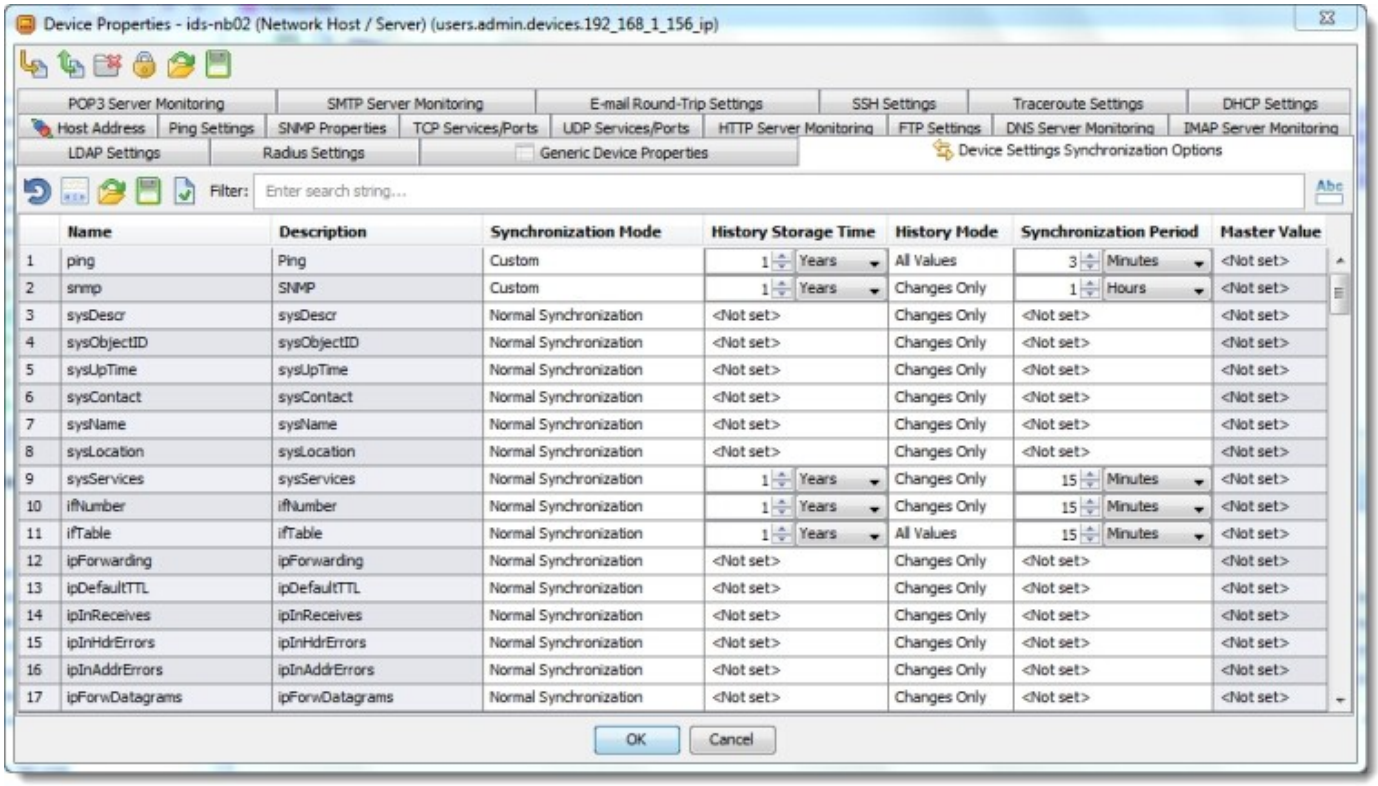
Эти опции доступны только для 3 версии SNMP. Пользователь может выбрать уровень безопасности, пользовательское имя, пароли и протоколы для авторизации и шифрования. См. раздел Описание опций безопасности SNMP 641.

Опции синхронизации

AtomMind Network Manager периодически опрашивает отслеживаемый объекты. Интервал между смежными запросами является очень важным параметром мониторинга. Он определяет баланс между точностью отслеживания данных и загрузкой процессора/памяти/сети. Кроме того, интервал опроса не должен быть короче периода обновления на стороне агента (см раздел Обработка данных 1799).

Опрос в AtomMind Network Manager внедрен в виде синхронизации 514; *интервал опроса* представляет `Synchronization Period` (см. Общие свойства устройства 510), его можно настроить через Опции настройки

синхронизации устройства 502. Вы можете определить общий период синхронизации для отдельных SNMP-переменных.



	Name	Description	Synchronization Mode	History Storage Time	History Mode	Synchronization Period	Master Value
1	ping	Ping	Custom	1 Years	All Values	3 Minutes	<Not set>
2	snmp	SNMP	Custom	1 Years	Changes Only	1 Hours	<Not set>
3	sysDescr	sysDescr	Normal Synchronization	<Not set>	Changes Only	<Not set>	<Not set>
4	sysObjectID	sysObjectID	Normal Synchronization	<Not set>	Changes Only	<Not set>	<Not set>
5	sysUpTime	sysUpTime	Normal Synchronization	<Not set>	Changes Only	<Not set>	<Not set>
6	sysContact	sysContact	Normal Synchronization	<Not set>	Changes Only	<Not set>	<Not set>
7	sysName	sysName	Normal Synchronization	<Not set>	Changes Only	<Not set>	<Not set>
8	sysLocation	sysLocation	Normal Synchronization	<Not set>	Changes Only	<Not set>	<Not set>
9	sysServices	sysServices	Normal Synchronization	1 Years	Changes Only	15 Minutes	<Not set>
10	ifNumber	ifNumber	Normal Synchronization	1 Years	Changes Only	15 Minutes	<Not set>
11	ifTable	ifTable	Normal Synchronization	1 Years	All Values	15 Minutes	<Not set>
12	ipForwarding	ipForwarding	Normal Synchronization	<Not set>	Changes Only	<Not set>	<Not set>
13	ipDefaultTTL	ipDefaultTTL	Normal Synchronization	<Not set>	Changes Only	<Not set>	<Not set>
14	ipInReceives	ipInReceives	Normal Synchronization	<Not set>	Changes Only	<Not set>	<Not set>
15	ipInHdrErrors	ipInHdrErrors	Normal Synchronization	<Not set>	Changes Only	<Not set>	<Not set>
16	ipInAddrErrors	ipInAddrErrors	Normal Synchronization	<Not set>	Changes Only	<Not set>	<Not set>
17	ipForwDatagrams	ipForwDatagrams	Normal Synchronization	<Not set>	Changes Only	<Not set>	<Not set>

Период синхронизации для SNMP-устройств по умолчанию 24 часа. Некоторым переменным присваивают особые периоды синхронизации, например, данные сетевого интерфейса и статистики процесса зачитываются каждые 15 минут и каждые 30 секунд соответственно. Для этих и других переменных можно задать свой собственный период времени.

19.1.6.2.2 Мониторинг по SNMP

AtomMind Network Manager предоставляет мониторинг для практически любого типа сетевых устройств, совместимых с SNMP. Мониторинг, включающий опрос и события, предлагают SNMP-операции *get* и *traps*. Дополняя друг друга, эти технологии мониторинга являются неотъемлемой частью стратегии надежного мониторинга.

SNMP-опрос позволяет получить все метрические данные, необходимые для управления. При периодическом опросе, информации поступает с задержкой, определенной интервалом мониторинга. Это может снизить точность мониторинга и скорость получения ответа. **События** могут помочь решить эту проблему: как только менеджер получает сообщение о событии, он может запросить свежую информацию, необходимую для обработки события. С другой стороны, событие предоставляет лишь ограниченную информацию об определенном аспекте жизни управляемого объекта. Более того, нет гарантии, что события, реализованные как SNMP-ловушки, будут доставлены менеджеру. Т.о., наилучшая стратегия - опрос значений, относящихся к определенному событию, когда менеджер получает уведомление, и периодическое выполнение полного опроса. Используя периодический опрос и события вместе, можно достичь необходимого сочетания точности, скорости, надежности и эффективного использования сетевых и вычислительных ресурсов.

В этой главе рассказывается об использовании SNMP-опроса 1845 и ловушек 1846 в контексте AtomMind Network Manager, за ней (главой) следует раздел о том, как эта информация в дальнейшем обрабатывается и анализируется 1847.

19.1.6.2.2.1 SNMP-опрос

SNMP-опрос позволяет Вам вести постоянный мониторинг значений различных идентификаторов объекта. AtomMind Network Manager использует SNMP-опрос для мониторинга доступности, работоспособности сетевых интерфейсов, CPU, памяти и дисков, процессов на рабочих станциях, VMware-серверов, беспроводных устройств, принтеров и пр. Существует множество примеров их использования в соответствующих разделах данного руководства.

SNMP опрос - это ряд `get` -запросов и ответов. AtomMind Network Manager скрывает информацию об этой связи, реализуя операции чтения в рамках стандартного процесса синхронизации 514. Полученная информация вносится к ядро для обработки и использования различными средствами мониторинга.

19.1.6.2.2.2 Получение SNMP-ловушек и сообщений

SNMP-ловушка/информационное сообщение - это предоставляемое без запроса сообщение, генерируемое агентом для уведомления менеджера о произошедшем событии.

AtomMind Network Manager может получать и обрабатывать SNMP-уведомления обоих типов: ловушки и сообщения. При получении уведомления, оно используется в качестве источника для внутреннего `trap`-события AtomMind. Это позволяет одинаково обрабатывать SNMP-уведомления, аналогично тому, как обрабатываются события AtomMind 73 других типов.

Пример: чтобы запустить действие в ответ на уведомление, можно создать тревогу для соответствующего события `trap` и установить для него корректирующее действие 793. Например, Вы сможете выполнить сценарий на удаленном ПК с Linux, когда получен определенный тип ловушки.

См. также раздел Мониторинг и объединение SNMP-уведомлений 1886 и, в частности, SNMP-уведомление о тревогах 1886.

Таким образом, AtomMind Network Manager позволяет организовать мониторинг по событиям почти для любого пользовательского устройства SNMP и при необходимости совместить 1845 их с опросом SNMP.

Реализация

Когда AtomMind Network Manager получает SNMP-уведомление, он активирует событие 73 `trap` в контексте администрирование 1450. Таблица данных события заполняется следующими свойствами (см. RFC 1215 RFC 1157):

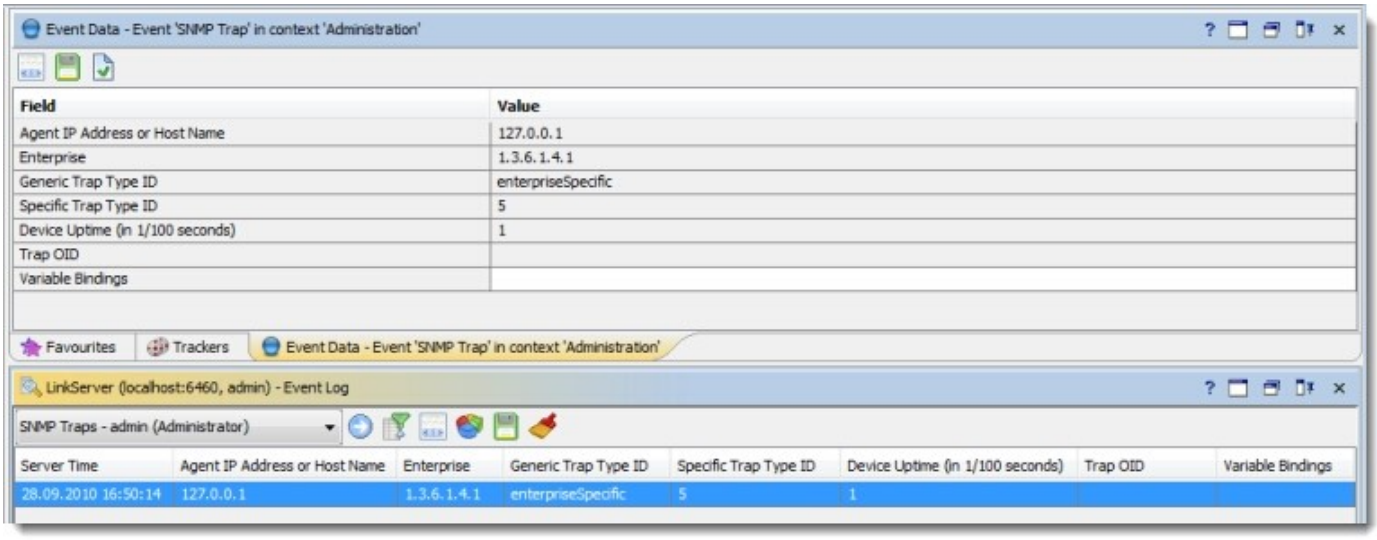
Поле события SNMP-уведомления	Описание
IP-адрес или имя узла агента	Адрес системы-генератора SNMP-уведомлений.
Версия SNMP	Версия полученного уведомления.
Тип уведомления	Тип уведомления (Ловушка или Сообщение)
ID объекта ловушки	Определяет тип управляемого объекта, который генерирует ловушку.
Общий тип ID ловушки	Определяет общий тип ловушки, одной из `coldStart` (0), `warmStart` (1), `linkDown` (2), `linkUp` (3), `authenticationFailure` (4), `egpNeighborLoss` (5), `enterpriseSpecific` (6)
Специфичный тип ID ловушки	Зависящий от производителя ID типа ловушки. Доступен, если Общий ID ловушки -- `ENTERPRISE_SPECIFIC` (6)
Доступное время устройства	Количество времени (1/100 секунды) между инициализацией устройства и созданием уведомления.
ID объекта ловушки	Идентификации ловушки для SNMP версии 2(c) и 3 уведомлений.
Привязки переменной	Привязки переменной, связанной с уведомлением.
ID обработчика	Достоверный ID обработчика уведомления.

Пример: отправка SNMP-ловушек из рабочей станции с Linux/Unix

Чтобы убедиться, что AtomMind Network Manager правильно получает SNMP-ловушки, можно отправить их при помощи команды `snmptrap`. Например, если Ваш AtomMind Network Manager настроен принимать SNMP-ловушки со строкой доступа `public` , используя транспортный протокол `UDP` на выбранном по умолчанию порту `162`, Вы можете использовать следующую команду:

```
sudo snmptrap -v1 -c public udp:<AtomMind Server IP address>:162 enterprises localhost 6 5 1
```

Она создаст версию 1 SNMP ловушки `enterpriseSpecific` с 5 специальным типов ловушек, с `enterprises` (`1.3.6.1.4.1`) - зависящий от производителя ID объекта, и 1/100 значением доступного времени устройства. Вы можете убедиться, что AtomMind Network Manager генерирует соответствующее событие, используя фильтр Журналирования событий SNMP-ловушек. Дважды щелкните мышью по записи события, чтобы увидеть его полные данные:



Можно настроить AtomMind Network Manager на автоматическое обнаружение отправителей SNMP-ловушек и создание для них учетных записей устройств[497]. См. Глобальные настройки для Автоматического обнаружения источников SNMP-ловушек[644] драйвер устройства сетевого хоста.

19.1.6.2.2.3 Использование данных SNMP

Данные SNMP, полученные из управляемых устройств, вводятся в ядро системы. Информация доступна в форме настроек устройства Сетевого хоста[593] и используется инструментальными средствами AtomMind Network Manager.

Просмотр данных SNMP

Всю информацию, собранную из SNMP-устройств, можно просмотреть как параметры[642] данного устройства. Чтобы просмотреть информацию, выберите устройство в Системном дереве, а затем выберите действие **Настроить устройство** в его контекстном меню. Кроме того можно просто дважды щелкнуть мышью по устройству, и откроется окно настройки устройства. Может быть множество переменных (параметров), десятки вкладок. Вы можете активировать/отключить разметку с вкладками для более удобства просмотра.

Обработка данных SNMP

Данные SNMP, полученные из определенного устройства, используют различные инструментальные средства AtomMind Network Manager, такие как тревоги, виджеты, графики, запросы и пр. Они анализируют данные, чтобы определить тип устройства и процесса. Данные SNMP можно использовать для получения статуса, числовых значений и характеристик определенного отслеживаемого устройства.

Например, устройство классифицируется как принтер, если его SNMP- переменная `prtGeneralTable`. Эта переменная вместе с другими данными для устройства используется средствами отслеживания принтера для получения о нем данных: описания, модели, серийного номера, свойств, текущего статуса. SNMP использует тревоги принтера, чтобы определить возможные проблемы принтера, такие как заедание бумаги, отсутствие бумаги, перенаполнение лотка, отсутствие чернил и пр.

Вы можете создать собственные инструментальные средства для осуществления мониторинга за отдельными типами SNMP-устройств в Вашей сети. В большинстве случаев это можно выполнить непосредственно в среде AtomMind Network Manager без программирования. В более сложных условиях можно использовать SDKAtomMind[1339]. Он позволяет создавать собственные плагины, которые будут обрабатывать данные SNMP наиболее удобным способом для используемых устройств.

19.1.6.2.3 Управление устройствами по SNMP

Некоторые сетевые устройства, сервисы и приложения можно контролировать по SNMP. Тип доступных операций и методы их реализации различаются от типа устройств/приложений/сервисов.

Примеры операций управления

Существуют два вида операций управления, которые можно выполнять по SNMP: **изменение настройки** и **выполнение прямого действия**.

ИЗМЕНЕНИЕ НАСТРОЙКИ

Изменяемые параметры настройки для управляемого SNMP-устройства/сервера/приложения представлены в виде доступных для записи переменных (объектов с ID).

Один из простейших примеров настройки по SNMP - установка стандартного значения, например, одного из описанных в *RFC 1213*. Так, для изменения **описания устройства** можно изменить значение `sysDescr` (`1.3.6.1.2.1.1.1`) на нужное значение строки.

Изменения в настройках могут вовлекать и более сложные процедуры, включая соответствующий ряд изменений, создание рядов таблиц и пр.

ВЫПОЛНЕНИЕ ДЕЙСТВИЯ

Существуют некоторые действия, которые выполняются в зависимости от конкретного типа управляемых объектов. Все эти действия выполняются как **ряд SNMP-**операций для объектов с определенными ID.

Например, чтобы перезагрузить коммутатор Cisco, можно присвоить значение `2` (обозначающее операцию перезагрузки) для переменной `sysReset` (`1.3.6.1.4.1.9.2.9.9`).

Как активировать операции

Поскольку все контролирующие операции в SNMP - это фактически изменение переменных (ID объектов), чтобы выполнить операцию управления, менеджер должен установить значение определенной SNMP -переменной, которое отображает управляемое устройство. Переменные для них должны быть описаны в MIB-файлах. Обратитесь к документации устройства/сервиса/приложения, которое необходимо контролировать.

Чтобы записать новое значение отдельной SNMP-переменной для устройства, можно следовать стандартной процедуре:

1. Дважды щелкните мышью по элементу устройства в Системном дереве. Появится Окно **настройки устройства**.
2. Выберите переменную, которую необходимо изменить в соответствующей MIB-вкладке. Кроме того, Вы можете де-активировать разметку с вкладками и найти переменную в простом списке всех доступных переменных.
3. Установите новое желаемое значение и сохраните свойства. Это активирует процесс синхронизации 514, который отправит значение устройству.

Будьте осторожны при изменении параметров работающих устройств! Убедитесь, что Вы не изменяете параметр, важный для работоспособности системы.

Можно автоматизировать повторяющиеся операции управления, используя стандартные инструментальные средства AtomMind Network Manager, например, виджеты 943, скрипты 879, тревоги (их действия по автоматическому исправлению 793) и пр.

19.1.6.2.4 Отправка SNMP-ловушек и сообщений

Существуют два типа SNMP-уведомлений: запросы Ловушек и Сообщений. Основная разница между ними - разная степень надежности. Ловушки являются не подтвержденными и ненадежными, потому что отправитель не может знать, была ли получена ловушка. SNMPv2 позволил решить эту проблему, внедрив уведомления о сообщениях, которые можно считать подтвержденными Ловушками. Отправитель сообщений может теперь ожидать подтверждения от получателя. Если подтверждение не поступает в течение определенного периода времени, уведомление можно отправить еще раз.

В то же время Сообщения добавляют сети и вычислительным ресурсам дополнительные издержки: исходное извещение о сообщении должно хранится в памяти отравителя до получения ответа, ему требуется дополнительная обработка на стороне получателя и увеличение трафика при повторении.

Ловушки и Сообщения позволяют найти баланс между надежностью и ресурсами: для важных уведомлений можно использовать Сообщения, а "обычные" можно отправлять в виде Ловушек.

AtomMind Network Manager может генерировать оба типа SNMP-уведомлений, используя действие 87 **Отправить SNMP-ловушку** (`sendSnmpTrap`) из корневого контекста 1559.

Можно инициировать действие **Отправить SNMP-ловушку** вручную, активировав контекстное меню узла сервера () в Системном дереве, выбрав подменю Сетевое управление, а затем выбрав действие **Отправить SNMP-ловушку**. Кроме того, можно настроить так, чтобы ловушки отправлялись в ответ на тревогу - см корректирующие действия 800.

Формат ввода

Это действие можно настроить с параметрами следующего формата:

Имя	Тип	Описание	Информация
trapType	Строка	Тип уведомления	Тип уведомления: Ловушка или Сообщение (Inform).
targetAddress	Строка	IP-адрес или имя хоста	Адрес получателя ловушки.
port	Целое	Порт	Порт, на который будет отправлена ловушка (162 по умолчанию).
protocol	Строка	Протокол	Транспортный протокол, который используется до доставки ловушки (UDP или TCP).
timeout	Длинное	Таймаут	Время ожидания до момента, когда запрос отправляется повторно, или истекает время ожидания (5 секунд по умолчанию).
enterprise	Строка	ID объекта ловушки	Специфичный для производителя ID объекта ловушки для 1-ой версии SNMP или ID объекта ловушки/сообщения для 2-ой и 3-ей версий SNMP.
variableBindings	Таблица данных	Привязки переменных	Привязки переменных, который ассоциируются с данной ловушкой. Привязки задаются в таблице в следующем формате:

Имя	Тип	Описание
variableBindingsOID	Строка	ID объекта переменной
variableBindingsType	Целое	Тип переменной
variableBindingsValue	Строка	Значение переменной

snmpVersion	Целое	Версия протокола SNMP	Версия SNMP.
community	Строка	Доступ Community	Строка доступа (для уведомлений 1-й и 2-1 версий SNMP)
userName	Строка	Имя пользователя	Имя пользователя для уведомлений 3 версии. Обратите внимание, что система пытается обнаружить авторизацию и пароли доступа в Таблице пользователей SNMP 645 (используя заданное Имя пользователя и ID процессора). Т.о. запись о пользователе должна быть в таблице.
engineId	Данные	Authoritative Engine ID	ID обработчика в двоичном представлении (для уведомлений 3-й версии).
securityLevel	Целое	Уровень безопасности	Уровень безопасности для уведомлений 3-й версии.

Формат вывода

Действие ничего не возвращает Ловушкам. Для сообщений ответ возвращается в следующем формате:

Имя	Тип	Описание	Информация
variableBindings	Таблица данных	Привязки переменной	Данные возвращаются как ответ на Сообщение. Привязки заданы в таблице в следующем формате:

Имя	Тип	Описание
variable	Строка	Переменная
value	Строка	Значение

errorMessage	Строка	Ошибка	Текстовое описание статуса успеха/ошибки.

19.1.7 WMI (инструментарий управления Windows)

WMI **Инструментарий управления Windows** - это инфраструктура для управления данными и операциями на ОС с Windows.

WMI - это **Система управления предприятием, основанная на использовании Web-технологий** (**WBEM**), стандартная технология доступа к административной информации на предприятии, реализованная Microsoft. WMI использует **Общую информационную модель** (**CIM**) промышленного стандарта для отображения систем, приложений, сетей, устройств и других управляемых компонентов, как совокупности объектов и их взаимосвязи.

Информацию об архитектуре WMI, возможности реализации на различных ОС можно найти в разделе MSDN Microsoft.

Удаленное управление

WMI можно использовать для мониторинга и управления компьютерами с Windows удаленно. Удаленные WMI-соединения осуществляются через **Распределенную модель компонентных объектов** (**DCOM**). Удаленный компьютер следует правильно настроить с разрешением DCOM-соединений и выполнения WMI-запросов. См. разделы Настройка удаленного доступа к WMI 665 и Настройка DCOM для удаленного доступа 2162.

Универсальная модель для управляемых данных

WMI основан на общей информационной модели, которая извлекает и описывает управляемую среду в рамках объектно-ориентированной модели. Данная модель затрагивает, фактически, все управляемые элементы современной IT-среды, включая устройства и их компоненты, компьютерные системы, операционные системы, сети, программное обеспечение, пользователей, физическую среду, статистику выполнения. Т.о. доступ ко всем управляемым ресурсам может осуществляться обычным способом. В то же время CIM (общая информационная модель) позволяет производителям предоставлять свои особенные свойства.

WMI описывает все управляемые ресурсы как объекты, которые представляют данные элементов (свойства, доступные для чтения) и контролирующие средства (свойства и методы, доступные для записи). Каждый WMI-объект - это экземпляр определенного класса. Классы определяют ряд свойств и методов собственных экземпляров.

Например, экземпляры класса Win32_Printer представляют устройства принтеры. Этот класс определяет несколько десятков свойств принтера, включая его имя, описание, драйвер, возможности, приоритет, статус, различные атрибуты и т.д. Некоторые из этих свойств доступны только для чтения, в то время как другие (такие, как "Приоритет", "Прямой", "Скрытый", "Сохранить напечатанные файлы" и т.д.) доступны для записи и могут использоваться для настройки принтеров через WMI. Класс Win32_Printer также представляет несколько способов, позволяющих контролировать принтеры: "Пауза", "Возобновить", "Напечатать тестовую страницу", "Отменить все задачи" и др.

События

Управляемые ресурсы и WMI-инфраструктура используют события для уведомления систем мониторинга/управления об изменениях. Получатели событий должны подписаться на интересующие их события, задав соответствующие параметры в фильтре. Фильтр описывает состояния, при которых получатель хочет получать уведомления.

Язык запросов WMI

Язык запросов WMI (**WQL**) - это реализация (от Microsoft) языка запросов общей информационной модели (CQL) для Общей объектной модели. Это подкласс ANSI SQL с несколькими изменениями для поддержки WBEM/WMI. WQL позволяет запрашивать информацию и подписываться на уведомления о событиях. См. главы WQL в библиотеке MSDN.

Драйвер устройства WMI

В AtomMind Network Manager WMI-поддержку осуществляет драйвер устройства WMI 661. Драйвер позволяет:

- подключаться к запущенному на удаленных компьютерах WMI
- запрашивать WMI-объекты по факту исполнения WQL-запросов
- получать WMI-классы и их экземпляры

- выполнять методы, предоставляемые WMI-объектами и изменять их доступные для записи свойства
- подписываться на уведомления о WMI-событиях, используя WQL-запросы и получать события от WMI-машины
- запускать автоматические действия в ответ на WMI-события
- кэшировать информацию WMI на сервере
- хранить административную информацию в БД
- обнаруживать I811 компьютеры с подключенным WMI и настраивать их с целью мониторинга и управления.

19.1.7.1 Примеры использования WMI

В этом разделе представлены примеры того, как можно использовать WMI для мониторинга и управления компьютерами с ОС Windows в сети. Изучите и другие классы, их свойства и методы, которые можно использовать для управления Вашей системой.

19.1.7.1.1 Контроль сервисов Windows

WMI представляет сервисы Windows, как экземпляры класса **Win32_Service**. Вы можете добавить этот класс к активам устройства или же использовать WQL-запросы для получения информации и осуществления контроля за сервисами на удаленных Windows-компьютерах. Несколько примеров представлено ниже.

Перечисление неактивных сервисов

Для того, чтобы получить имена неактивных сервисов и их статусы, можно использовать следующий WQL-запрос:

```
SELECT DisplayName,State FROM Win32_Service WHERE State <> 'Running'
```

Для того, чтобы активировать его, выполните следующее:

- Выберите устройство в **Контекстном дереве**
- Щелкните правой кнопкой мыши и выберите элемент **Редактировать свойства устройства**
- Добавьте новую запись в таблицу, расположенную во вкладке **WQL-запросы**; введите имя, описание и текст запроса (как указано ранее) в соответствующих полях
- Нажмите OK

Устройство автоматически выполнит синхронизацию и запрос. Результат будет доступен в таблице **WQL-запросов** в **Настройках устройства**.

Определение сервисов, которые можно остановить

Используйте следующий запрос для получения только тех сервисов, которые можно остановить:

```
SELECT * FROM Win32_Service WHERE AcceptStop = True
```

Следуйте по аналогии с процедурой, описанной выше.

Сервисы Start и Stop

Для того, чтобы запустить сервис на определенном комьютере, следует выполнить следующие операции:

- Выберите соответствующее устройство в **Контекстном дереве**
- Кликните по нему правой кнопкой мыши и выберите в контекстном меню элемент **Win32_Service** (для Устройств хоста сети в подменю **WMI**)
- Выберите там метод **StartService**
- Выберите или введите путь к объекту, определяющий сервис, который необходимо запустить
- Нажмите OK

Для того, чтобы остановить сервис, выполните эту процедуру для метода **StopService**.

Очередность загрузки групп сервисов

Существуют другие классы, содержащие информацию о сервисах Windows. За дополнительной информацией обратитесь к документации по WMI.

Например, экземпляры класса **Win32_LoadOrderGroup** представляют группы сервисов, которые определяют зависимости выполнения. Можно добавить этот класс в активы устройств или использовать WQL:

```
SELECT * FROM Win32_LoadOrderGroup
```

19.1.7.1.2 Управление принтерами

Класс **Win32_Printer** WMI представляет устройство печати, связанное с компьютером, запущенным на ОС Microsoft Windows. Оно позволяет выполнять ряд задач мониторинга и управления; некоторые из них представлены в этом разделе.

Список установленных принтеров

Чтобы перечислить все принтеры, связанные с компьютером, можно либо добавить класс Win32_Printer к активам устройства и изучить содержание переменной Win32_Printer variable, или же выполнить следующий WQL-запрос:

```
SELECT * FROM Win32_Printer
```

Обратите внимание, что если Вы используете WQL, Вы можете выбрать лишь те принтеры, которые удовлетворяют определенному условию, путем добавления условий фильтрации в запрос.

Изменить свойства принтера

Некоторые из параметров принтера можно изменить как свойства доступные для записи. Для этого перейдите в раздел по Настройкам устройства, откройте таблицу Win32_Printer и измените свойства для требуемого принтера. Наиболее простой и безопасный путь - попытаться изменить комментарий к принтеру в поле **Комментарий**.

За дополнительной информацией об их свойствах и значениях обратитесь к документации WMI-класса Win32_Printer.

Переименовать принтер

С помощью класса Win32_Printer в активах устройства, Вы можете использовать их методы для контроля за принтерами. Например, чтобы переименовать принтер, можно активировать действие **RenamePrinter** (также как и все другие действия принтера) из подменю Win32_Printer в контекстном меню устройства.

Перечислить текущие задачи принтера

Существуют другие классы WMI, предоставляющие полезную функциональность для устройств печати, подключенных к Windows-компьютеру. Например, класс **Win32_PrintJob** WMI предназначен для работы конкретного принтера. Для того, чтобы получить список текущих задач для каждого принтера, необходимо выполнить следующий запрос:

```
SELECT Description, Document, Status, JobStatus, ElapsedTime, Priority  FROM
Win32_PrintJob
```

Запрос выбирает описание принтера, документы, находящиеся в состоянии печати или ожидания, фактическое время работы и свойства.

За дополнительной информацией обратитесь к документации по Вашим устройствам, а также изучите, какие WMI-объекты могут быть доступны в Вашей сети.

19.1.7.1.3 Пользователи операционной системы

Есть несколько WMI классов, представляющих информацию об авторизованных в операционной системе Microsoft Windows пользователях и их процессах.

Среди них:

- Win32_LogonSession
- Win32_LoggedOnUser
- Win32_SessionProcess
- Win32_Process

AtomMind Network Manager предоставляет коробочный мониторинг всех авторизованных пользователей и их процессов.

Инструментальная панель **Обзор сетевого устройства** включает в себя два запроса 829, которые предоставляют пользователям операционной системы следующую информацию:

- Данные об авторизованных пользователях (WMI)
- Данные о выполняемых процессах каждого пользователя (WMI)

19.1.7.1.4 Мониторинг журнала событий Windows через WMI

AtomMind Network Manager предлагает простой способ мониторинга Журнала Событий Windows через WMI.

Щелкните правой кнопкой мыши по устройству WMI и вызовите действие `Редактировать свойства устройства`. Затем выберите вкладку `Запросы события WQL` в открытом окне и добавьте запись к таблице. В поле `Запрос WQL` введите запрос, который выбирает события создания экземпляров для `Win32_NTLogEvents` с соответствующими параметрами фильтра по необходимости. Например, для мониторинга всех событий введите следующий текст:

```
SELECT * FROM __InstanceCreationEvent WHERE TargetInstance ISA 'Win32_NTLogEvent'
```

Вам нужно также заполнить поля `Имя` и `Описание`.

Щелкните на `OK`. С этого момента AtomMind Network Manager начнет извлекать свежие данные из Журнала Событий Windows при каждой синхронизации устройства. Объекты журнала представлены как события AtomMind. Вы можете также задать другой период опроса путем определения периода синхронизации для переменной `wmiEvents`.

Для отслеживания и контроля событий Windows вызовите фильтр `События WMI` в Журнале Событий AtomMind.

19.1.8 Мониторинг доступности

Существуют два определения термина *доступность* в контексте сетевого мониторинга, отвечающих на два вопроса: "Доступен ли сейчас хост/интерфейс/сервис?" и "как часто хост/интерфейс/сервис оказывается недоступным?" Ряд показателей, отвечающих на первый вопрос мы обозначаем, как **индикаторы текущей доступности**. Ответ на второй вопрос требует разносторонней оценки в разные моменты времени, и поэтому мы обращаемся к **статистическим метрикам доступности**. Два данных подхода описаны ниже.

Оба подхода используют ICMP -запросы (в основном, Пинг, иногда Трассировку). Если Ваше устройство не отвечает на ICMP-запросы, Вам потребуется найти другой способ отслеживания его доступности. См. сервисы Пинг[596] и Трассировка[607].

Текущая доступность

AtomMind Network Manager поддерживает следующие индикаторы текущей доступности:

- Статус **Online/Offline** отслеживается сервисом Пинг[596], доступ к нему осуществляется через Результаты мониторинга сервиса Ping[597] или при помощи следующих инструментальных средств: *тревога* Устройство Offline[1956], *запросы* Устройства Offline[832] и Сводка состояния устройства[832].
- **Статус сетевых интерфейсов узла** (отслеживается по SNMP). AtomMind Network Manager предоставляет соответствующие инструментальные средства: тревоги Интерфейс опущен[1867] и Административный интерфейс опущен[1867], *отчет/запрос* **опущенный интерфейсов.**
- **Время ответа** устройства отслеживает сервис Пинг[596] (доступ осуществляется через Результаты мониторинга сервиса Ping[597]). Соответствующие инструментальные средства: *тревога* Длительное время ожидания[1956], *диаграммы* Время пинга и Лучшие 10 по скорости ответа (на пинг) и *запросы/отчеты* Времени ответа[1956].
- **Коэффициент потери пакетов** отслеживает сервис Пинг[596] (доступ осуществляется также через Результаты мониторинга сервиса пинг[597]. См. *тревогу* Высокий коэффициент потери пакетов[1956], *диаграммы* Пинг потери пакетов и Первые 10 из потери пакетов, а *также отчеты/запросы* Коэффициенты потери пакетов[1956].
- **Достижимость узла** можно отследить через сервис Трассировка[607]. Сервис предоставляет подробную информацию о сетевом маршруте хоста.
- **Доступность виртуальной машины** можно отследить при помощи инструментальных средств Мониторинг сервера VMware, в частности, при помощи тревоги Состояние Виртуальной машины[1912].
- **Доступность принтера** можно отследить при помощи инструментальных средств мониторинга принтера**:** тревог(сообщений) о Состоянии принтера[1918], Метке маркера[1918], Обеспечении принтера[1918] и виджета Информация о принтере[1919].

- **Доступность сервисов** можно отследить при помощи тревоги Сервис Offline 1956 и **таблицы статусов сервисов**. **Таблица статусов сервисов** доступна в виде элемента Избранное 1524 и отображает статусы доступности/работоспособности 1798 всех сервисов для зарегистрированных устройств. Статус сервиса обозначен цветом: **серый** цвет означает, что сервис *отключен* (т.е. не настроен для мониторинга), **зеленый** -- сервис находится в режиме *online*, и **красный** -- в режиме *offline*.

Статистическая доступность

Доступность как статистическое значение - это доля времени, когда система находится в рабочем состоянии.

Доступность устройства измеряется как определенный временной промежуток в процентном соотношении: `100 - packetLossRate`, где `packetLossRate` - это процент неуспешных пинг-запросов. В AtomMind Network Manager значение статистическая доступность устройства для временного интервала отражена в *отчете* Доступность ниже 100% 1956.

Датчик времени отклика

AtomMind Network Manager включает датчик 2181 **Время отклика сервиса**, который отслеживает время отклика определенного сервиса. Этот датчик может создаваться, используя действие Установить профайл мониторинга 1808.

19.1.8.1 Тревоги доступности

Заранее настроенные тревоги доступности включают:

Имя тревоги	Условия вызова	Примечания
Долгое время отклика	Среднее время отклика превышает определенный порог.	Заранее настроена для всех контекстов сетевых устройств.
Высокий процент потери пакетов	Процент потери пакетов превышает определенный порог.	Заранее настроена для всех контекстов сетевых устройств.
Устройство офлайн	Хост отключен от сервера на определенное время (по умолчанию *10 минут*).	Встроена в базовый дистрибутив AtomMind Server (см. Встроенная тревога Устройство офлайн 781).
Сервис офлайн	Определенный сервис находится в режиме офлайн.	Для создания тревоги используйте группу **Доступность** в действии Настроить профайл мониторинга 1808.
Сетевой интерфейс отключен (SNMP) 1867	Рабочее состояние сетевого интерфейса *отключен*, но административно он *включен*.	Используйте действие Настроить профайл мониторинга 1808, для создания - группу **Сетевые интерфейсы**. Доступно только для устройств, совместимых с SNMP.
Административное завершение работы сетевого интерфейса (SNMP) 1867	Желаемое состояние интерфейса *отключен*.	Используйте действие Настроить профайл мониторинга 1808, для создания - группу **Сетевые интерфейсы**. Доступно только для устройств, совместимых с SNMP.
Изменение состояния сетевого интерфейса (SNMP) 1867	Сетевой интерфейс изменил свое рабочее состояние.	Используйте действие Настроить профайл мониторинга 1808, для создания - группу **Сетевые интерфейсы**. Доступно только для устройств, совместимых с SNMP.

19.1.8.2 Отчёты доступности

Встроенные отчеты доступности включают:

Класс отчета/запроса	Имя	Описание
Время отклика	Обзор времени отклика	Среднее время передачи сигнала для всех устройств
	Время отклика превышает 100 миллисекунд	Среднее время передачи сигнала для устройств с временем передачи свыше 100 миллисекунд
	Время отклика превышает 500 миллисекунд	Среднее время передачи сигнала для устройств с временем передачи свыше 500 миллисекунд
	Время отклика Топ 10	Среднее время передачи сигнала для топ 10 устройств
	Время отклика Топ 50	Среднее время передачи сигнала для топ 50 устройств
Процент потери пакетов	Обзор процента потери пакетов	Процент потери пакетов для всех устройств
	Процент потери пакетов больше 10%	Устройства с процентом потери пакетов свыше 10%
	Процент потери пакетов больше 50%	Устройства с процентом потери пакетов свыше 50%
	Потеря пакетов Топ 10	10 устройств с наивысшим процентом потери пакетов
	Потеря пакетов Топ 50	50 устройств с наивысшим процентом потери пакетов
Доступность	Доступность не 100%	Статистическая доступность 1854 для устройств с ненулевым процентом потери пакетов
Сетевые ресурсы	Интерфейсы по типу	Количество интерфейсов по типу
Отключенные интерфейсы	Отключенные интерфейсы	Сетевые интерфейсы, находящиеся в Отключенном или Административно Отключенном состоянии

19.1.8.3 Графики доступности

Имя графика	Описание	Примечания
Время проверки связи	Представляет минимальное, среднее и/или максимальное время отклика (передачи сигнала) проверки связи.	График может быть создан для конкретного контекста по SNMP с использованием действия Установить профайл мониторинга 1808.
Потеря пакетов проверки связи	Представляет средние значения процента потери пакетов.	График может быть создан для конкретного контекста по SNMP с использованием действия Установить профайл мониторинга 1808.
Время отклика Топ 10	Представляет устройства с наибольшим средним временем передачи сигнала.	График автоматически создается соответствующим действием автозапуск 941 в группе **Топ 10** и доступен в **инструментальной панели Топ 10**.
Процент потери пакетов Топ 10	Представляет устройства с наибольшим процентом потери пакетов.	График автоматически создается соответствующим действием автозапуск 941 в группе **Топ 10** и доступен в **инструментальной панели Топ 10**.
Состояние сервисов	Представляет состояние сервисов: "Включен" как 1, "Отключен" как 0	График может быть создан для конкретного контекста Хост Сети с использованием действия Установить профайл мониторинга 1808 (группа Доступности).

Доступность	Указывает средние проценты доступности устройства, сгруппированные по различным временным периодам (часы, дни и т.д.)	Часть инструментальной панели **Обзор сетевых устройств**.

19.1.9 Мониторинг производительности

Эта тема включает в себя несколько обособленных характеристик *производительности устройства*:

- центральный процессор (CPU) - см. Мониторинг производительности CPU 1856;
- запоминающие устройства (жесткие диски, оптические диски и пр.) - см. Мониторинг системы хранения данных 1857;
- программное обеспечение, запущенное на хосте (процессы) - см. Мониторинг процесса 1860.

Мониторинг эффективности сети 1863 и приложения/сервиса 1888 (часто считаются частью мониторинга производительности) описан отдельно.

Некоторые аспекты производительности устройства, включая датчики **загрузки CPU**, **Использование физической памяти**, **Использование виртуальной памяти**, могут отображаться в реальном времени в одном окне, при использовании виджета информации об IP-хосте.

Следует затронуть еще один тип мониторинга производительности, относящийся к технологиям создания виртуальной среды. AtomMind Network Manager отслеживает различные метрические данные *производительности виртуальных машин сервера VMware ESX*. Описанные в отдельной главе (см Мониторинг сервера VMware), инструментальные средства мониторинга производительности виртуальной инфраструктуры тоже упоминаются в соответствующем подразделе ниже.

19.1.9.1 Мониторинг нагрузки на центральный процессор

Мониторинг **производительности процессора** выполняется по SNMP и WMI.

Мониторинг производительности процессора по SNMP

AtomMind Network Manager собирает и обрабатывает данные загрузки **процессора** по SNMP.

Согласно RFC 1514 **загрузка процессора** - это показатель времени загруженности процессора в течение последней минуты его работы.

AtomMind Network Manager использует SNMP-переменную `hrProcessorLoad`, чтобы получить среднее значение загрузки процессора. Переменная - это часть `hrProcessorTable`, предоставляющая отдельные данные для всех процессоров хоста.

Инструментарий мониторинга процессора по SNMP включает в себя:

- Диаграммы нагрузки на процессор
- Топ 10 нагрузки на процессор
- Отчеты по нагрузке на процессор
- Действие автозапуска 941 10 самых загруженных устройств, отображающее 10 устройств с наиболее высокой нагрузкой на процессор

Диаграмму нагрузки на процессор можно создать и настроить при помощи действия **Установить профиль мониторинга** -- см. группу **Поцессор** в разделе Действие Установить профиль мониторинга 1808 .

Системные администраторы могут создавать другой инструментарий (например, тревоги, виджеты и пр.) для интерпретации данных SNMP-переменной `hrProcessorTable` .

Виртуальные машины могут осуществлять мониторинг процессора на *серверах VMware ESX* при помощи тревоги нагрузки на процессора на виртуальной машине 1912 и диаграммы использования процессора виртуальной машиной 1914.

Мониторинг нагрузки на процессор по WMI

AtomMind Network Manager использует WMI для получения данных о нагрузке на процессор от экземпляров класса `Win32_PerfFormattedData_Counters_ProcessorInformation`. Инструментарий мониторинга нагрузки на процессор по WMI включает тревогу активности процессора 1956.

Тревоги по нагрузке на процессор

Имя тревоги	Условие вызова	Примечания
Активность процессора (WMI)	**Индикатор активности процессора** удовлетворяет *условию,* состоящему из *порога* и *операции сравнения*. Доступные индикаторы - это процент времени, когда процессор находился в следующих состояниях: в режиме ожидания, не в режиме ожидания, в режиме пользователя, работа с аппаратными прерываниями, использование максимальной частоты.	Используйте действие Установить профайл мониторинга 1808, группу **Процессор** для создания; доступно только для устройств, совместимых с WMI.

Отчеты по нагрузке на процессор

Имя отчета	Описание
Обзор нагрузки на процессор	Нагрузка на процессор 1856 для всех устройств
Нагрузка на процессор свыше 50%	Устройства с нагрузкой на процессор свыше 50%
Нагрузка на процессор свыше 90%	Устройства с нагрузкой на процессор свыше 90%
Нагрузка на процессор Топ 10	10 устройств с наивысшей нагрузкой на процессор
Нагрузка на процессор Топ 50	50 устройств с наивысшей нагрузкой на процессор

Графики по нагрузке на процессор

Имя графика	Описание	Примечания
Нагрузка на процессор (SNMP)	Отображает нагрузку на процессор(ы) 1856.	График может создаваться для определенного контекста SNMP, используя действие Установить профайл мониторинга 1808.
Нагрузка на процессор Топ 10 (SNMP)	Представляет устройства с наибольшей утилизацией процессора.	График автоматически создается соответствующим действием автозапуска 941 в группе **Топ 10** и доступен в **инструментальной панели Топ 10**.

19.1.9.2 Мониторинг систем хнанения данных

В этом разделе говорится о двух отличительных свойствах средств хранения данных компьютера: оперативной памяти (RAM, и пр.) и постоянной памяти (жесткие диски, оптические диски и пр.).

Данные **мониторинга хранилища** можно получить по SNMP и WMI.

Протокол **SNMP** (SNMP-таблица `hrStorageTable`) используется для получения подробной информации о *хранилищах*, включая их *типы, описание, общий* и *свободный объем, единицы измерений* и пр.

AtomMind Network Manager использует следующие классы **WMI**:

- `Win32_PerfFormattedData_PerfOS_Memory` для отслеживания памяти
- `Win32_LogicalDisk` для отслеживания постоянного хранилища.

AtomMind Network Manager включает следующий инструментарий мониторинга хранилища:

- График объем памяти (SNMP) 1956
- График использования хранилища (SNMP)
- Диаграммы Топ 10 по использованию памяти (SNMP)
- График Память (WMI) 1956
- График Свободное дисковое пространство (WMI) 1956
- Отчеты/запросы Использование памяти 1956
- Диаграммы Топ 10 по использованию томов дисков (SNMP)

- Отчеты/запросы использования томов диска 1956
- Отчеты/запросы Свободное пространство томов диска 1956
- Действие автозапуска 941 Топ 10 по использованию памяти, которое отображает 10 больших по использованию памяти процессов.

График объема хранилища и график использования хранилища можно создать и настроить при помощи действия **Настроить профиль мониторинга** -- см. группу **Тома Хранилища** в Настроить профиль мониторинга 1808.

Использование хранилища виртуальных машин VMware можно отследить при помощи инструментария Мониторинг сервера VMware, а именно, используя:

- диаграммы Производительности диска виртуальной машины 1915
- диаграммы Использование памяти виртуальной машины 1915
- тревоги Абсолютное использование памяти виртуальной машины 1912
- тревоги Относительное использование памяти виртуальной машины 1912
- тревоги Скорость чтения HDD виртуальной машины 1912
- тревоги Скорость записи HDD виртуальной машины 1913

Тревоги памяти

Имя тревоги	Условие вызова	Примечания
Память (WMI)	**Индикатор использования памяти** удовлетворяет *условию,* состоящему из *порога* и *операции сравнения*. Имеются следующие индикаторы: Доступная физическая память (байты), Размер кэша файловой системы (байты), Количество ошибок кэша в секунду, Количество ошибок страницы в секунду, Количество прочитанных страниц с диска для разрешения сложных ошибок страниц, Количество записанных на диск страниц в свободной физической памяти, Операции чтения диска в свободной физической памяти, Операции записи диска для разрешения сложных ошибок страниц.	Используйте действие Установить профайл мониторинга 1808, группу **Объемы хранения** для создания; доступна только для устройств, совместимых с WMI.

Тревоги хранения

Имя тревоги	Условия вызова	Примечания
Место хранения (SNMP)	**Процент использованного места в данном объеме хранения** превышает заданный порог.	Тревогу можно создать, используя действие Установить профайл мониторинга 1808 (группа **Объемы хранения**); доступна только для устройств, совместимых с SNMP.
Свободное место на диске (WMI)	**Процент свободного места на диске** удовлетворяет *условию,* состоящему из *порога* и *операции сравнения*.	Используйте действие Установить профайл мониторинга 1808, группу **Объемы хранения** для создания; доступна только для устройств, совместимых с WMI.

Отчеты памяти

Класс отчета/запроса	Имя отчета/запроса	Описание
Использование памяти	Обзор использования памяти	Процент использования физической памяти для всех устройств
	Использование памяти свыше 75%	Устройства с процентом использования памяти выше 75%
	Использование памяти свыше 90%	Устройства с процентом использования памяти выше 90%

	Использование памяти Топ 10	10 устройств с наибольшим процентом использования памяти
	Использование памяти Топ 50	50 устройств с наибольшим процентом использования памяти

Отчеты хранения

Класс отчета/запроса	Имя отчета/запроса	Описание
Утилизация томов диска	Обзор утилизации томов диска	Процент использования дискового пространства для всех устройств
	Утилизация томов диска выше 75%	Устройства с процентом использования дискового пространства выше 75%
	Утилизация томов диска выше 90%	Устройства с процентом использования дискового пространства выше 90%
	Утилизация томов диска Топ 10	Процент использования дискового пространства для топ 10 устройств
	Утилизация томов диска Топ 50	Процент использования дискового пространства для топ 50 устройств
Свободное место томов диска	Свободное место томов диска Топ 10	10 устройств с наибольшим процентом использования дискового пространства
	Свободное место томов диска Топ 50	50 устройств с наибольшим процентом использования дискового пространства

График использования памяти

Имя графика	Описание	Примечания
Использование памяти и дискового пространства (SNMP)	Отображает процент использования пространства хранения на хосте, включая физическую память, виртуальную память и дисковое пространство.	График может создаваться для определенного контекста SNMP, используя действие Установить профайл мониторинга 1808.
Использование памяти Топ 10 (SNMP)	Представляет устройства с наибольшим использованием памяти.	График автоматически создается соответствующим действием автозапуска 941 в группе **Топ 10** и доступен в **инструментальной панели Топ 10**.

График использования хранения

Имя графика	Описание	Примечания
Утилизация томов диска Топ 10 (SNMP)	Представляет тома диска и соответствующие устройства с наибольшим процентом утилизации.	График автоматически создается соответствующим действием автозапуска 941 в группе **Топ 10** и доступен в **инструментальной панели Топ 10**.

Модель утилизации памяти

Эта модель 810 прикрепляется к каждому сетевому устройству. Она описывает общие переменные **Физическая память**, **Виртуальная память**, **Область подкачки** и набор правил, рассчитывающих их значение из доступных метрик памяти, специфичных для устройства.

Все другие инструменты анализа утилизации памяти (графики, отчеты, тревоги и др.) основаны на этой модели.

Модель утилизации диска

Эта модель 810 прикрепляется к каждому сетевому устройству. Она описывает основную переменную **Статистика диска** и набор правил, который рассчитывает использование диска из доступных метрик диска, специфичных для устройства.

Все другие инструменты анализа утилизации диска (графики, отчеты, тревоги и др.) основаны на этой модели.

19.1.9.3 Мониторинг процессов

Мониторинг процесса представляет собой отслеживание метрических данных программного обеспечения, запущенного на узле. Информацию об активности процесса можно получить по SNMP или WMI.

Мониторинг процесса по SNMP

Мониторинг процесса по SNMP осуществляется при помощи таблиц `hrSWRunTable` и `hrSWRunPerfTable`, которые предоставляют доступ к информации, связанной с каждым отдельным процессом, запущенным или загруженным в память или находящемся в процессе загрузки. Он включает в себя мониторинг операционной системы, драйверов и приложений.

Данные мониторинга, полученные по SNMP, обрабатываются и сохраняются в табличной переменной `processList` AtomMind Network Manager; для каждого процесса предоставляется следующая информация:

Значение	**Источник (имя SNMP-переменной или описание выражения вычисления**	**Описание**
Имя процесса	`hrSWRunName`	Текстовое описание запущенного программного обеспечения, включая производителя, новую версию и чаще всего название используемого ПО.
Путь процесса	`hrSWRunPath`	Описание расположения долгосрочного хранилища (например, дисковод), с которого это ПО было загружено.
Параметры процесса	`hrSWRunParameters`	Описание параметров этого ПО, когда оно загружено в первый раз.
Тип процесса	`hrSWRunType`	Тип ПО (с числовым кодом): • Неизвестно (1) • Операционная система (2) • Драйвер устройства (3) • Приложение (4)
Статус процесса	`hrSWRunStatus`	Статус запущенного ПО (с числовым кодом): • Запущено (1) • Работоспособно (2) -- ожидает ресурс (т.е. CPU, накопитель, IO (ввод-вывод)) • Не работоспособно (3) -- загружено, но ожидает событие • Неверно (4) -- не загружено
Нагрузка на процессор, %	Производная 1801 от `hrSWRunPerfCPU`, разделенная на число CPU	Среднее количество сантисекунд общих ресурсов системы CPU, затрачиваемых процессом в секунду. Количество CPU узла, предоставленных SNMP-переменной `hrProcessorTable`.
Использование памяти, в килобайтах	`hrSWRunPerfMem`	Общий объем памяти системы, назначенной для данного процесса.
Использование памяти, %	`hrSWRunPerfMem * 100 / hrMemorySize`	Процент от общего объема ОЗУ узла (`hrMemorySize`), назначенной для данного процесса (`hrSWRunPerfMem`).

Для того, чтобы получить доступ к информации о процессах хоста, выберите в контекстном меню устройства элемент **Просмотр Состояния устройства** и перейдите к вкладке **Список процессов**.

Мониторинг процессов на основе WMI

WMI предоставляет несколько классов, выдающих информацию о процессах и их активности. AtomMind Network Manager использует данные классов `Win32_Process` и `Win32_PerfFormattedData_PerfProc_Process`.

Диаграммы и тревоги мониторинга процессов

AtomMind Network Manager предоставляет тревогу процессов 1956, которая уведомляет пользователя, когда процесс использует ресурсы (загрузка CPU или памяти) больше установленной предельной величины.

Существуют также диаграммы, относящиеся к мониторингу процесса: **Диаграмма подсчета процессов** и **Диаграмма подсчета экземпляров процесса**.

Эти диаграммы и тревогу можно создать для отдельного устройства, используя действие Настроить профиль мониторинга 1808 (см. здесь **Группу процессов**).

Тревоги мониторинга процессов

AtomMind Network Manager предоставляет тревогу процессов 1956, которая уведомляет пользователя, когда процесс использует ресурсы (нагрузка на процессор или память) больше установленной предельной величины.

Имя тревоги	Условие вызова	Примечания
Высокое использование процессора процессами (SNMP)	**Время процессора, использованное процессом(ами)** превышает заданное пороговое значение.	Тревога может создаваться, используя действие Установить профайл мониторинга 1808 (группа **Процессы**); доступна только для устройств, совместимых с SNMP.
Высокое использование памяти процессами (SNMP)	**Объем памяти, использованный процессом(ами)** превышает заданное пороговое значение.	Тревога может создаваться, используя действие Установить профайл мониторинга 1808 (группа **Процессы**); доступна только для устройств, совместимых с SNMP.
Количество экземпляров процессов (SNMP)	**Количество экземпляров процессов** превышает заданное пороговое значение.	Тревога может создаваться, используя действие Установить профайл мониторинга 1808 (группа **Процессы**); доступна только для устройств, совместимых с SNMP.
Количество экземпляров процессов (WMI)	**Количество заданных экземпляров процессов** удовлетворяет *условию,* состоящему из *порога* и *операции сравнения*.	Используйте действие Установить профайл мониторинга 1808, группу **Процессы** для создания; доступна только для устройств, совместимых с WMI.
Активность процессов (WMI)	**Индикатор активности заданного процесса** удовлетворяет *условию,* состоящему из *порога* и *операции сравнения*. Доступные индикаторы: процент времени процессора, используемого процессом, количество физической памяти, используемой процессом, или количество пространства виртуального адреса, используемого процессом.	Используйте действие Установить профайл мониторинга 1808, группу **Процессы** для создания; доступна только для устройств, совместимых с WMI.

Графики производительности процессов

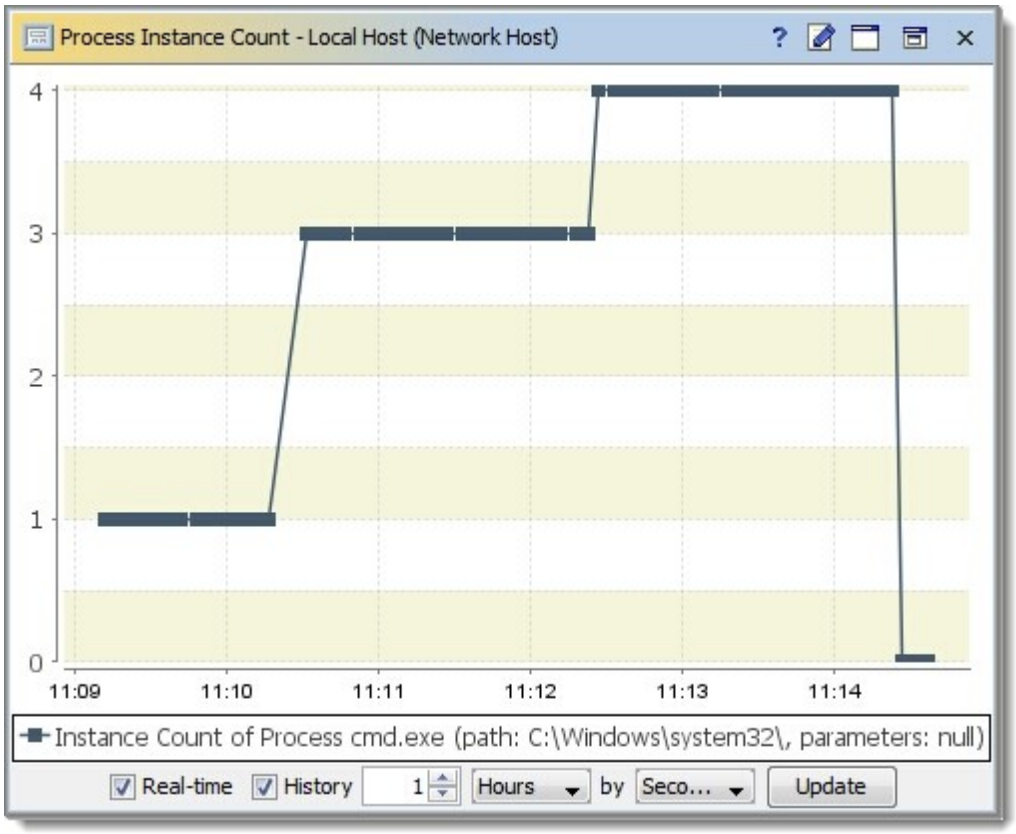
Имя графика	Описание	Примечания
Количество процессов (SNMP)	Отображает общее количество процессов, запущенных на определенном компьютере или устройстве. Данные извлекаются через SNMP.	График может создаваться для определенного контекста SNMP, используя действие Установить профайл мониторинга 1808.
Количество экземпляров процессов (SNMP) 1862	Отображает количество экземпляров определенного процесса. Экземпляр процесса определяется именем, путем и параметрами процесса. Данные извлекаются через SNMP.	График может создаваться для определенного контекста SNMP, используя действие Установить профайл мониторинга 1808.
Количество процессов (WMI)	Отображает общее количество процессов, запущенных на определенном компьютере или устройстве. Данные извлекаются через WMI.	График может создаваться для определенного контекста WMI, используя действие Установить профайл мониторинга 1808.

Количество экземпляров процессов (WMI)	Отображает количество экземпляров определенного процесса, определяемого по имени процесса. Данные извлекаются через WMI.	График может создаваться для определенного контекста WMI, используя действие Установить профайл мониторинга 18081.

19.1.9.3.1 Диаграмма Подсчет экземляров процесса

Диаграмма **Подсчет экземпляров процесса** - это инструментарий мониторинга процессов 18601, который графически отображает количество процессов экземпляров.

Экземпляр процесса определяется по имени и (дополнительно) по связанному пути и параметрам.



Во время настройки диаграммы для хоста, можно выбрать **Имя процесса** из выпадающего списка процессов или ввести его самостоятельно.

Поля **Путь процесса** и **Параметры процесса** можно оставить незаполненными (т.е. с нулевым значением - NULL). В этом случае, включены процессы с заданным именем и **любые** пути/параметры.

Обратите внимание, что "*установка в состояние 0*" ("NULL") и "*пустая строка*" - это разные значения. Если Вы не хотите отфильтровывать процессы по пути и/или другим параметрам, убедитесь, что соответствующие поля обозначены как **<Not set>** , что соответствует нулевому значению. Пустое поле означает, что введена пустая строка, и это свойство будет использовано для фильтрации процессов.

Например, если Путь процесса обозначен как <Not set>, а поле Параметры процесса пустое, диаграмма будет считать процессы без указанных параметров (с пустой строкой) и проигнорирует их пути.

Чтобы установить значение NULL для поля, можно использовать элемент "Удалить значение", расположенный в контекстном меню поля.

19.1.9.4 Мониторинг общей производительности сервиса

Каждое устройство AtomMind Network Manager предоставляет таблицу Статусы синхронизации настроек 516, имеющую поле **Длительность ввода/вывода**. Это поле предоставляет отчет о времени, используемом для чтения/записи переменной во время последней синхронизации устройства 514.

Поле Длительность ввода/вывода может анализироваться выражением триггера 782 тревоги 779, чтобы:

- Измерить время выполнения любого **запроса** со стороны драйвера устройства базы данных 647
- Измерить время выполнения скрипта SSH 1903, выполняемого на удаленной машине Linux
- Измерить время загрузки веб-страницы, извлекаемой драйвером устройства HTTP 569
- Измерить время задержки любого другого сервиса, такого как DNS или DHCP

См. описание выражения триггера тревоги, которое анализирует длительность ввода/вывода переменной устройства, здесь 808.

19.1.10 Мониторинг трафика и пропускной способности

Мониторинг эффективности сети - это важный момент для ряда административных задач, включая:

- сетевые задачи обнаружения, идентификации, обработки и превентивные меры
- обнаружение неэффективности и узких мест, выполнение анализа загрузки, подтверждение необходимого качества сервиса
- распознавание вида и направления трафика, планирование сетевой мощности.

AtomMind Network Manager поддерживает два основных подхода к мониторингу сетевого потока и пропускной способности сети:

- Мониторинг сетевых интерфейсов по SNMP (см. ниже)
- Мониторинг по NetFlow.

Мониторинг использования сетевого трафика и пропускной способности по SNMP

AtomMind Network Manager можно настроить на периодический опрос сетевых узлов по SNMP 1828 и получение следующей информации об их сетевых интерфейсах, используя SNMP-переменную `ifTable` :

- общее количество байт, полученных и переданных из интерфейса
- текущая пропускная способность (бит в секунду)
- скорость интерфейса (максимальная пропускная способность)
- количество пакетов, содержащих ошибки, которые мешают доставке пакетов
- количество отброшенных пакетов

AtomMind Network Manager использует эти значения для подсчета использования *пропускной способности* и *трафика* интерфейса. Для устройств с переменной `ifTable` создается переменная `Interface Traffic` (`interfaceTraffic`). Она содержит подсчитанные значения, как описано в следующей таблице:

Имя поля	**Комментарии**
Имя интерфейса	Извлеченное из поля `ifDescr` переменной `ifTable`.
Входящий трафик, б/с	Посчитано как производная 1801 `ifInOctets`.
Исходящий трафик, б/с	Посчитано как производная 1801 `ifOutOctets`.
Использование входящей полосы, %	Посчитано как коэффициент между `Входящим трафиком` и скоростью его интерфейса (поле `ifSpeed`).
Использование исходящий полосы, %	Посчитано как коэффициент между `Исходящим трафиком` и скоростью его интерфейса (поле `ifSpeed`).

Можно просмотреть эту информацию для отдельного устройства, например, в Исполнителе запроса 839:

```
SELECT * FROM users.*.devices.deviceContext:ifTable as data
```

(Замените `deviceContext` именем контекста Вашего устройства)

Данные мониторинга сетевого интерфейса можно визуализировать, используя следующие графики:

- График Трафик интерфейса 1866
- Графики **Трафик интерфейса Топ 10** и **Использование пропускной способности Топ 10**
- График Ошибки интерфейса 1866
- График Отброшенные пакеты интерфейса 1866

Вы также можете видеть эти данные в форме таблицы, используя отчеты, а именно:

- Отчет Обзор трафика интерфейса 1867
- Отчеты 10 самых используемых интерфейсов и 50 самых используемых интерфейсов 1867
- Отчеты Использование полосы интерфейсом более чем на 50% и Использование полосы интерфейсом более чем на 90% 1868
- Отчеты 10 самых загруженных интерфейсов и 50 самых загруженных интерфейсов 1868

Функциональные характеристики отдельных интерфейсов на маршрутизаторах и коммутаторах можно использовать для диагностики сетевых проблем. AtomMind Network Manager предлагает для этого ряд тревог:

- Тревога Использование сетевого интерфейса 1867
- Тревога Сетевой интерфейс отключен 1867
- Тревога Административное отключение сетевого интерфейса 1867

Вы можете создать пользовательские инструментальные средства для мониторинга и анализа деятельности сетевых интерфейсов.

Тревоги трафика и утилизации интерфейса

Имя тревоги	Условия вызова	Примечания
Утилизация сетевого интерфейса (SNMP) 1867	Использование входящей и исходящей пропускной способности превышает заданный порог.	Используйте действие Установить профайл мониторинга 1808, группу **Сетевые интерфейсы** для создания. Доступна только для устройств, совместимых с SNMP.

Отчеты трафика и утилизации интерфейса

Класс отчета/запроса	Имя отчета/запроса	Описание
Трафик интерфейса	Обзор трафика интерфейса	Обзор трафика по каждому интерфейсу (включая входящий и исходящий трафик) для всех устройств
	Трафик интерфейса Топ 10	10 интерфейсов с максимальным общим трафиком (сумма входящего и исходящего трафика)
	Трафик интерфейса Топ 50	50 интерфейсов с максимальным общим трафиком (сумма входящего и исходящего трафика)
Утилизация пропускной способности интерфейса	Обзор пропускной способности интерфейса	Обзор пропускной способности по каждому интерфейсу для всех устройств
	Утилизация пропускной способности интерфейса выше 50%	Проценты использования входящей и исходящей пропускной способности для устройств с каждым значением выше 50%
	Утилизация пропускной способности интерфейса выше 90%	Проценты использования входящей и исходящей пропускной способности для интерфейсов с каждым значением выше 90%
	Утилизация пропускной способности интерфейса Топ 10	Проценты использования входящей и исходящей пропускной способности для 10 интерфейсов с максимальным общим использованием пропускной способности
	Утилизация пропускной способности интерфейса Топ 50	Проценты использования входящей и исходящей пропускной способности для 50 интерфейсов с максимальным общим использованием пропускной способности

Графики трафика, утилизации и статуса интерфейса

Имя графика	Описание	Примечания
Трафик сетевого интерфейса (SNMP) 1866	Показывает входящий и исходящий трафик указанных сетевых интерфейсов.	График может быть создан для определенного контекста SNMP, используя действие Установить профайл мониторинга 1808.
Трафик интерфейса Топ 10 (SNMP)	Представляет сетевые интерфейсы и соответствующие устройства с наибольшим общим (входящим и исходящим) трафиком.	График автоматически создается соответствующим действием автозапуска 941 в группе **Топ 10** и доступен в **инструментальной панели Топ 10**.
Утилизация пропускной способности интерфейса Топ 10	Представляет сетевые интерфейсы и соответствующие устройства с наибольшим общим процентом использования пропускной способности (входящей и исходящей).	График автоматически создается соответствующим действием автозапуска 941 в группе **Топ 10** и доступен в **инструментальной панели Топ 10**.
Ошибки сетевого интерфейса (SNMP) 1866	Показывает количество ошибочных пакетов. Ошибочные пакеты - это входящие пакеты, которые содержат ошибки, препятствующие их доставке в протокол более высокого уровня.	График может быть создан для определенного контекста SNMP, используя действие Установить профайл мониторинга 1808.
Отброшенные пакеты сетевого интерфейса (SNMP) 1866	Отображает количество отброшенных пакетов: входящие пакеты, которые были выбраны для отбрасывания, хотя не было обнаружено никаких ошибок, препятствующих их доставке в протокол более высокого уровня. Например, если было получено слишком много пакетов и стек протоколов не имеет достаточно ресурсов для надлежащей обработки пакетов, они могут быть отброшены.	График может быть создан для определенного контекста SNMP, используя действие Установить профайл мониторинга 1808.

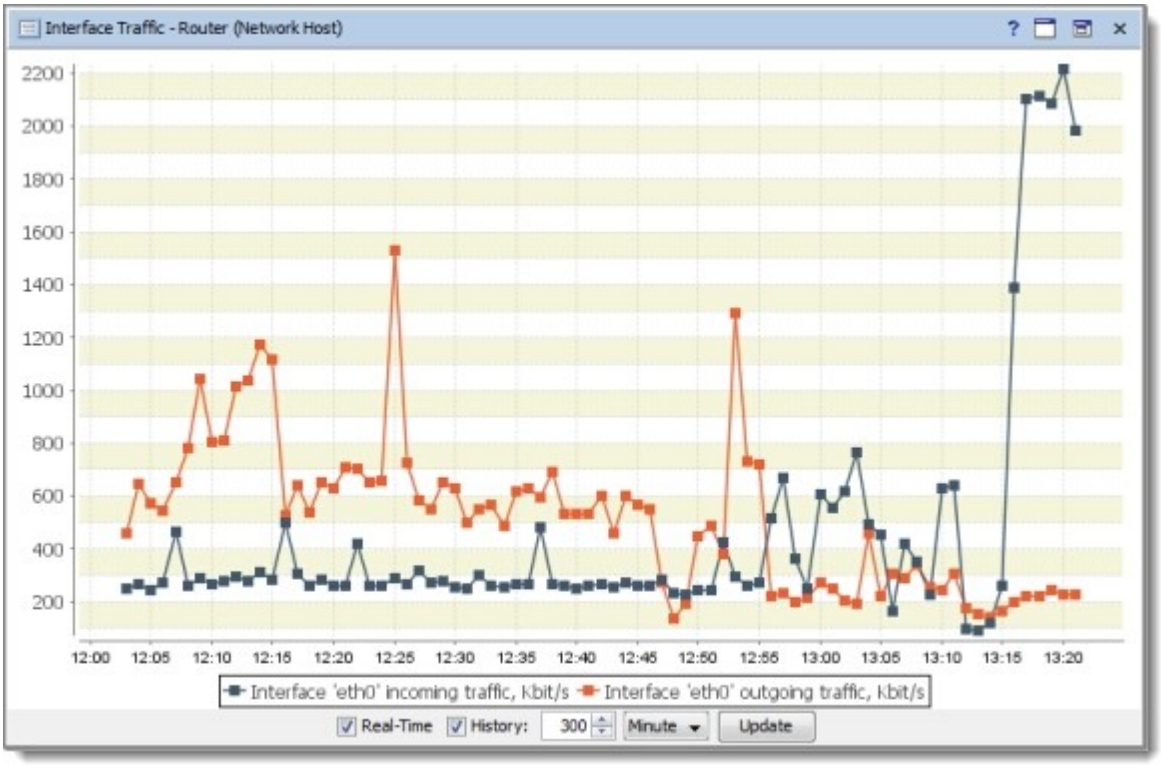
Модель трафика интерфейса

Эта модель 810 прикрепляется к каждому сетевому устройству. Она описывает побайтовую таблицу счетчиков для каждого интерфейса. Эта таблица заполняется, используя 64-битные счетчики трафика, если такие счетчики предоставляются устройством, в ином случае используются 32-битные счетчики.

Все другие инструменты анализа утилизации диска (графики, отчеты, тревоги и др.) основаны на этой модели.

19.1.10.1 Диаграмма трафика сетевого интерфейса

Диаграмма трафика сетевого интерфейса отображает входящий и исходящий трафик заданных интерфейсов сети.



Диаграмму можно открыть при помощи инструментальной панели 912 **Обзор трафика интерфейса**. Кроме того, можно создать отдельный виджет для контекста, используя действие Настроить профиль мониторинга 1808.

19.1.10.2 Утилизация пропускной способности сетевого интерфейса

График Утилизация пропускной способности сетевого интерфейса показывает процент использования входящей и исходящей пропускной способности определенных сетевых интерфейсов.

График доступен в инструментальной панели **Обзор сетевых устройств**. Кроме того, вы можете создать свой собственный виджет для любого устройства, используя действие Установить профайл мониторинга 1808.

19.1.10.3 Диаграмма ошибок сетевого интерфейса

Диаграмма ошибок сетевого интерфейса отображает количество пакетов ошибок. Пакеты ошибок - это входящие пакеты, которые содержат ошибки, мешающие их отправке протоколу более высокого уровня.

Можно создать виджет с **диаграммой ошибок интерфейса** для отдельного контекста с активированным SNMP при помощи действия Настроить профиль мониторинга 1808.

19.1.10.4 Диаграмма сбросов сетевого интерфейса

Диаграмма сбросов сетевого интерфейса показывает отвергнутые пакеты для отдельного Сетевого интерфейса.

Отвергнутые пакеты - это входящие пакеты, которые были отвергнуты, несмотря на то, что не были обнаружены ошибки, которые могли бы помешать доставке пакетов протоколу более высокого уровня. Это происходит, например, в том случае, если получено слишком много пакетов, а у стека протоколов недостаточно ресурсов для обработки пакета, и он может быть отвергнут.

Вы можете создать виджет с **диаграммой сбросов сетевого интерфейса** для отдельного контекста с активированным SNMP, используя действие Настроить профиль мониторинга 1808.

19.1.10.5 Тревога Использование сетевого интерфейса

Тревога Использование сетевого интерфейса отслеживает использование пропускной способности сетевого интерфейса. Тревога активизируется, если использование входящей или исходящей пропускной способности превышает предельную величину. Интерфейс и предельная величина задаются как параметры тревоги. Вы можете создать **тревогу Использования сети**, запустив действие Установить профиль мониторинга 1808.

19.1.10.6 Тревога Сетевой интерфейс отключен

Тревога **Сетевой интерфейс отключен** отслеживает рабочее состояние сетевого интерфейса (доступно как поле `ifOperStatus` в переменной `ifTable`). Тревога активируется, когда рабочее состояние определено как **"отключено"** (2), а административный статус как **"включено"** (1). Отслеживаемый интерфейс задается в параметрах тревоги. Вы можете создать **тревогу Сетевой интерфейс отключен**, активировав действие Настроить профиль мониторинга 1808.

19.1.10.7 Тревога Административное отключение сетевого интерфейс

Тревога Административное отключение сетевого интерфейса отслеживает желаемое состояние сетевого интерфейса (доступно как поле `ifAdminStatus` в переменной `ifTable`). Тревога активизируется, когда искомое состояние **"отключено"** (2). Отслеживаемый интерфейс задан в параметрах тревоги. Вы можете создать **тревогу Сетевой интерфейс отключен**, вызвав действие Настроить профиль мониторинга 1808.

19.1.10.8 Тревога Изменение состояния сетевого интерфейса

Тревога Изменение состояния сетевого интерфейса контролирует *рабочее состояние* сетевого интерфейса (доступна как поле `ifOperStatus` в переменной `ifTable`). Тревога активируется, когда меняется рабочий статус (с рабочего на нерабочий и наоборот). Отслеживаемый интерфейс указывается как параметры тревоги. Вы можете создать **тревогу Изменение состояния сетевого интерфейса** путем вызова действия Установить профайл мониторинга 1808.

19.1.10.9 Отчет о трафике интерфейса

Отчет о трафике интерфейса представляет входящий и исходящий трафик с информацией о пропускной способности для всех интерфейсов всех зарегистрированных устройств. Этот отчет включает следующие поля:

Имя поля	**Описание**
Устройство	Тип и имя устройства
Интерфейс	Имя сетевого интерфейса
Входящий трафик, бит/с	Входящий трафик (в байтах в секунду)
Исходящий трафик, бит/с	Исходящий трафик (в байтах в секунду)
Использование входящей пропускной способности, %	Коэффициент использования входящей пропускной способности в процентах
Использование исходящей пропускной способности, %	Коэффициент использования исходящей пропускной способности в процентах

19.1.10.10 Отчеты о топ 10 и 50 интерфейсов по трафику интерфейсов

Отчеты о **первых 10** и **50 по трафику интерфейсов** представляют интерфейсы с наиболее высоким трафиком (входящим и исходящим) среди всех зарегистрированных устройств. Эти отчеты включают следующие поля:

Имя поля	**Описание**
Устройство	Тип и имя устройства
Интерфейс	Имя сетевого устройства
Входящий трафик, бит/с	Входящий трафик в байтах в секунду

Исходящий трафик, бит/с	Исходящий трафик в байтах в секунду.

19.1.10.11 Отчеты об использовании пропускной способности интерф

Отчеты об **использовании пропускной способности интерфейса более чем на 50%** и **90%** представляют интерфейсы с использованием входящей и исходящей пропускной способности более установленной предельной величины. Эти отчеты включают следующие поля:

Имя поля	**Описание**
Устройство	Тип и имя устройства
Интерфейс	Имя сетевого интерфейса
Использование входящей полосы, %	Использование входящей полосы в процентах
Использование исходящей полосы, %	Использование исходящей полосы в процентах.

19.1.10.12 Отчеты о 10 и 50 интерфейсах с наибольшим использовани

Отчеты о **10** и **50 интерфейсах с наибольшим использованием пропускной способности**, представляют интерфейсы с наиболее высоким использованием пропускной способности (входящей и исходящей) среди всех зарегистрированных устройств. Эти отчеты включают следующие поля:

Имя поля	**Описание**
Устройство	Тип и имя устройства
Интерфейс	Имя сетевого интерфейса
Использование входящей полосы, %	Коэффициент использования входящей полосы в процентах
Использование исходящей полосы, %	Коэффициент использования исходящей полосы в процентах

19.1.11 Использование NetFlow для анализа трафика

NetFlow обеспечивает мощный, гибкий и эффективный способ сбора данных, необходимых для подробного анализа производительности IP-сети. Он может быть использован для ряда задач, таких как следующие:

- Мониторинг использования сети, включая текущие и исторические данные; поиск самых активных потребителей полосы пропускания: пользователи, хосты, приложения, протоколы и т.д.
- Измерение фактического трафика; получение подробной информации об использовании полосы пропускания по конкретным хостам, сетевым интерфейсам приложений, протоколам и т.д.
- Отслеживание сетевых проблем и выявление их первопричин; обеспечение безопасности и обнаружение угроз, таких как черви, вирусные атаки и другая несанкционированная и нежелательная деятельность.
- Настройка сети для повышения производительности в наиболее распространенных случаях использования, специфичных для конкретной организации.
- Обеспечение требуемого качества сетевой связи для критически важных приложений.
- Отслеживание тенденций сети и планирование мощностей.

NetFlow сегодня - это наиболее широко используемая технология для сбора статистических данных в больших сетях с высокой загруженностью. NetFlow и иные родственные технологии поддерживаются рядом поставщиков сетевой инфраструктуры, включая Cisco, HP, Juniper и многих других.

19.1.11.1 Основы NetFlow

NetFlow и аналогичные протоколы предназначены для передачи информации об IP-трафике от точек наблюдения блокам собора данных. **Точка наблюдения** - это место в сети, где можно наблюдать IP-пакеты. Например, это может быть интерфейс маршрутизатора. Пакеты, поступающие в точку наблюдения, контролируются **экспортером**. Экспортер обрабатывает пакеты, накапливает информацию о трафике и периодически отправляет ее в **блок сбора данных** NetFlow. Блок сбора данных получает информацию, анализирует ее и хранит данные для дальнейшего использования аналитическими инструментами. Один блок может собирать информацию у нескольких экспортеров.

Экспортер не отправляет информацию о каждом конкретном пакете, который он наблюдает. Вместо этого, пакеты объединяются в несколько **IP-потоков**. Каждый поток накапливает данные (количество пакетов и байтов) для пакетов с определенными общими свойствами. Например, поток обычно включают следующую информацию о трафике:

- IP-адрес и номер порта источника
- IP-адрес и номер порта места назначения
- Тип IP-протокола
- Значение Типа Сервиса
- Индексы SNMP интерфейсов ввода и вывода (см. ifIndex в IF-MIB)
- Количество пакетов и байтов (октеты Уровеня 3), наблюдаемых в потоке
- Временные метки для моментов, когда в потоке наблюдались первый и последний пакеты
- Флажки TCP
- Информация о маршрутизации
- И другое

Экспортер периодически посылает накопленные данные о потоке блоку сбора данных. Потоки для отправки группируются в пакеты экспорта (дейтаграммы). Пакет экспорта включает в себя основную информацию, такую как версия NetFlow, число потоков, содержащихся в пакете, и нумерация последовательности. Блок сбора данных анализирует пакеты экспорта, извлекает информацию о потоке и сохраняет ее. После этого статистика трафика может использоваться для анализа.

AtomMind Network Manager реализует сервис сбора данных о потоках, обеспечивает сохранение данных netflow, предлагает готовые инструменты для анализа утилизации сети. Более того, AtomMind Network Manager позволяет создавать пользовательскую аналитику и средства визуализации для ваших специфичных нужд.

В AtomMind Network Manager все средства декомпозиции трафика поддерживаются плагином 207 **NetFlow**. Он обеспечивает сервис сбора потоков и средства для обработки, сбора, анализа и визуализации данных об IP-трафике.

19.1.11.2 Поддерживаемые версии NetFlow

Первоначально представленный Cisco, NetFlow развивался с течением времени. Есть несколько версий, включая устаревшие (версии 1 и 6), внутренние неизданные версии Cisco (версии 2, 3 и 4), относительно редко используемые, такие как версии 7 (только для коммутаторов, не маршрутизаторов) и 8 (предлагает возможность сбора для сокращенных объемов данных NetFlow). Две версии, которые на самом деле широко используются в практике - это версии 5 и 9.

Версия NetFlow 5 - все еще наиболее широко используемая версия NetFlow для анализа трафика. В ней есть все функции, необходимые для большинства реальных обстоятельств, также она поддерживается многими маршрутизаторами различных марок.

Версия NetFlow 9 (см. RFC 3954 "Системы Cisco, Экспорт Сервисов NetFlow, версия 9") является наиболее мощной, гибкой и расширяемой версией протокола потока Cisco IP. Записи потоков версии 9 можно настроить и описать в шаблонах. Это позволяет собрать большее число показателей. Эта версия NetFlow в основном используется для продвинутых обстоятельств, таких как мониторинг IPv6, MPLS и т.д.

Версия NetFlow 9 стала основой для недавнего **стандарта IPFIX** (RFC 3917, RFC 5101, RFC 5102 и т.д.). Хотя IPFIX (иногда обозначается как версия NetFlow 10) заменяет NetFlow 9 в нескольких аспектах, он пока широко не принят в качестве отраслевого стандарта продавцами сетевой инфраструктуры.

Есть также несколько протоколов "типа NetFlow", которые обеспечивают сходные функции с некоторыми вариациями или поддерживаются другими производителями. Например, sFlow основана на выборке и, следовательно, является весьма масштабируемой технологией, применимой к высоко скоростным сетям; с другой стороны, выборка значительно снижает точность.

19.1.11.3 Системные требования

Обработка NetFlow, как правило, потребляет много системных ресурсов из-за огромного количества данных, которые получаются, сохраняются, визуализируются блоком сбора данных. В больших сетях должны быть приняты специальные меры предосторожности, чтобы получить оптимальную производительность обработки потока. В некоторых случаях может потребоваться выделенный сервер обработки потока.

Ряд факторов может повлиять на системные требования сервера NetFlow. Первым фактором является скорость потока (число потоков в секунду), который получает AtomMind. Это зависит от количества датчиков, объема сетевого трафика, а также настроек экспорта потоков датчика. Вторым фактором является установка обработки потоков AtomMind Servera: интервалы агрегации, проценты выброшенного трафика, настройки разрешения имени DNS и т.д.

При планировании установки AtomMind Network Manager для обработки потоков, имейте в виду, что фактические потребности определяются конфигурацией вашей сети, объемом и уникальностью получаемых данных NetFlow. Фактическое использование ресурсов может сильно варьироваться в зависимости от этих факторов.

Ниже приведенная таблица помогает найти конфигурацию системы для выделенного сервера обработки потоков. Если обработка потока сочетается с регулярным мониторингом сети, системные требования должны быть также совмещены с обычными требованиями системы AtomMind Server 146. Обратите внимание, что обработка NetFlow особенно прихотлива в отношении места на жестком диске и производительности ввода/вывода.

Скорость потока в секунду	Центральный процессор	Ядра процессора	Оперативная память	Место на диске	Комментарии
До 100	3 ГГц	2	4 ГБ	500 ГБ	
От 100 до 1000	3 ГГц	4	8 ГБ	1 ТБ	
От 1000 до 10000	2 * 3 ГГц	8	16 ГБ	2 ТБ	Настоятельно рекомендуется конфигурация 10 000 или 15 000 RPM SAS накопители в RAID 0 или 10
Свыше 10000	2 * 3 ГГц	16	32 ГБ	4 ТБ	Настоятельно рекомендуется конфигурация 15 000 RPM SAS накопители в RAID 0 или 10

Если у вас есть дополнительные вопросы или вы испытываете проблемы с производительностью, пожалуйста, свяжитесь с ТВЭЛ для получения рекомендаций у наших системных инженеров.

19.1.11.4 Настройка экспортеров NetFlow

Сетевые устройства и сервисы должны быть правильно настроены, чтобы представлять статистические данные о пакетах, проходящих через них. Сбор и экспорт NetFlow выполняются и настраиваются независимо от различных сетевых элементов. Процедура конфигурирования существенно отличается для разных типов устройств и сервисов сбора данных. Здесь мы предлагаем лишь несколько точек входа для некоторых из наиболее широко используемых типов устройств и услуг. Обратитесь к документации вашего устройства/услуги для ознакомления с более подробной инструкцией по использованию и конфигурации.

Устройства Cisco

Для информации о конфигурации NetFlow на устройствах Cisco см. документацию на официальном сайте (www.cisco.com); хорошая стартовая статья: Конфигурирование NetFlow и экспорт данных NetFlow.

Другие производители

Существуют другие производители сетевых устройств, поддерживающие NetFlow, включая 3Com (HP Networking), Enterasys Networks, Extreme Networks, Juniper Networks и другие. Обратитесь к непосредственной документации производителя для получения информации о конфигурировании NetFlow.

Экспортеры программного обеспечения

Не все сетевые устройств поддерживают NetFlow. Если ни один из ваших маршрутизаторов или коммутаторов не поддерживает Netflow или один из его "родственных" протоколов, вы можете использовать экспортер программного обеспечения.

Например, если у вас есть коммутатор, поддерживающий зеркалирование портов / SPAN, вы можете настроить его для отправки копии сетевых пакетов в определенное место сети. Это может быть, например, компьютер, на котором запущен экспортер программного обеспечения, прикрепленный к зеркальному порту.

Кроме того, если коммутатор не позволяет отражать ваш трафик, вы все же можете контролировать данные NetFlow с помощью концентратора (не коммутатора), чтобы ваши сетевые пакеты достигали экспортера программного обеспечения. Например, при наличии концентратора между вашим внешним маршрутизатором и магистральным оборудованием, весь трафик, который проходит через соединение, будет доступен для экспортера программного обеспечения, прикрепленного к этому концентратору.

Существует ряд проприетарных и бесплатных экспортеров программного обеспечения NetFlow:

- fprobe: бесплатный инструмент на базе libpcap, переносная библиотека C/C++ для сбора сетевого трафика; fprobe может компилироваться для любой **операционной системы, совместимой с POSIX** (типа **Linux**/**BSD**/**Unix**).
- nProbe: коммерческий продукт nTop, способный играть роль экспортера и блока сбора данных; существует бесплатная версия для некоммерческих институтов и универсистетов; доступен на **Unix** (включая **MacOS X** и **Solaris**), **Windows** и **встраиваемых средах**.
- ng_netflow: реализует протокол NetFlow на **FreeBSD**; бесплатный**.**
- NDSAD: бесплатный экспортер для платформ **BSD** и **POSIX** (основан на библиотеке *libpcap*) и **Windows** (на основе *winpcap*).

19.1.11.5 Конфигурирование обработки потоков

Этот раздел описывает основные шаги, требуемые для настройки обработки NetFlow:

- Настроить плагин NetFlow 1871
- Добавить устройства датчики 1870 NetFlow
- Запустить обработку потоков 1875 и выбрать интерфейсы

19.1.11.5.1 Конфигурирование плагина NetFlow

Настройки базовой конфигурации NetFlow доступны в глобальной конфигурации плагина 207 NetFlow. Для получения доступа к этой конфигурации:

- Разверните узел **Драйвера/Плагины** ( ) в системном дереве 370.
- Щелкните дважды на плагине **NetFlow** ( ).

Доступны следующие настройки базовой конфигурации NetFlow:

Свойство	**Описание**
Включено	Включает/выключает сбор потоков.
Порт NetFlow	Порт прослушивания для данных NetFlow.
Порт sFlow	Порт прослушивания для данных sFlow.
Размер буфера, байты	Максимальный размер потока, т.е. размер буфера памяти, отведенный для получения потока.

19.1.11.5.2 Конфигурирование учетных записей устройств для датчиков NetFlow

Если плагин NetFlow развернут в определенной инсталляции AtomMind Network Manager, каждая учетная запись устройства 497 имеет свойство **Настройки сбора потоков**, используемое для настройки обработки потоков для этого устройства (датчика).

К свойству **Настройки сбора потоков** можно получить доступ через диалоговое окно **Редактировать свойства устройства**. Это устройство имеет следующие поля:

- **Включить сбор потоков**. Снятие этого флажка заставит сервер игнорировать все потоки, получаемые с адреса устройства.
- **Включить интерфейсы**. Позволяет обрабатывать только потоки, получаемые с определенных сетевых интерфейсов датчика. Потоки, соответствующие отключенным интерфейсам, будут незаметно сброшены.

19.1.11.5.2.1 Конфигурирование DNS и разрешения NetBIOS

AtomMind Network Manager определяет имена хостов через сервисы DNS и NetBIOS для более комфортной визуализации собранных данных.

Настройки разрешения DNS доступны в глобальной конфигурации плагина 207 NetFlow :

Свойство	Описание
Включено	Включает/выключает разрешение имени DNS. Отключение разрешения имени хоста обеспечивает максимальную производительность.
Разрешение DNS	Определяет метод связи для запросов разрешения имени: • **По запросу**. Имена DNS будут разрешаться только во время визуализации данных потока. Это ускоряет сбор потоков и улучшает производительность сервера во время сбора потоков, но визуализация потока будет всегда занимать дополнительное время, требуемое для обработки всех имен DNS адресов, которые нужно отобразить. • **Постоянный**. Имена DNS будут разрешаться сразу после получения данных и постоянно храниться в базе данных 692 сервера. Это обеспечивает быструю визуализацию потоков, но может снизить производительность сбора потоков. Этот режим подходит для небольших сетей.
Количество IP-адресов для определения как пакета	Определяет количество IP-адресов, отправленных одним пакетом в ответ на запрос DNS.
Количество потоков для сохранения в пуле	Определяет минимальное количество потоков обработки запросов DNS.
Максимальное количество потоков, разрешенных в пуле	Определяет максимальное количество потоков обработки запросов DNS.

Настройки разрешения NetBIOS также доступны в глобальной конфигурации плагина 207 NetFlow:

Свойство	Описание
Включить разрешение NetBIOS	Включает/выключает разрешение NetBIOS.

19.1.11.6 Просмотр статистики обработки NetFlow

Если плагин NetFlow развернут в определенной инсталляции AtomMind Network Manager, каждая учетная запись устройства 497 имеет переменную **Статистика сбора потоков**, используемую для отслеживания обработки потоков этого устройства (датчика).

Получить доступ к переменной **Статистика сбора потоков** можно через диалог овое окно **Просмотреть статус** учетной записи устройства. Эта переменная имеет следующие поля:

- **Общее количество потоков**. Количество потоков, полученных от датчика.
- **Количество сброшенных потоков**. Количество потоков, полученных от датчика и сброшенных из-за их соответствия интерфейсам, отключенным для сбора потоков.
- **Процент потери**. Процент потоков, которые не были получены (по большей части из-за перегрузки сети). Это количество рассчитывается путем отслеживания номеров последовательностей потоков.

19.1.11.7 Агрегирование данных потока

Сервер сбора NetFlow в типичной крупномасштабной сети получает тысячи потоков в секунду. Сохранение всех потоков в базе данных как постоянных в течение значительного времени и быстрая загрузка их по требованию в целях визуализации потребует огромных ресурсов сервера (мощность процессора, память и место на жестком диске).

Чтобы найти компромисс между подробной статистикой данных потока и потреблением ресурсов сервера, AtomMind Network Manager использует сложное многоуровневое кэширование потока и систему агрегации. Эта система включает в себя два основных уровня:

- Кэш потоков в памяти
- Многоуровневая агрегация потоков в базе данных сервера 692

Настройки агрегации данных потока можно осуществить в глобальной конфигурации плагина NetFlow. Для получения доступа к этой конфигурации:

- Разверните узел **Драйвера/Плагины** () в системном дереве 370.
- Щелкните дважды на плагине **NetFlow** ().

Кэш потоков в памяти

Кэш потоков в памяти - это первый кэш, куда попадает полученный поток. Это также самый быстрый кэш, он гораздо быстрее по сравнению с системами хранения потоков базы данных более низкого уровня. При этом оговоркой будет использование большой оперативной памяти сервера.

Кэш-память контролируется тремя параметрами:

Свойство	Описание
Максимальный размер кэш-памяти	Определяет максимальное количество потоков, которые могут быть собраны в памяти до начала этапа агрегации первого уровня.
Период кэш-памяти	Определяет период агрегации первого уровня. Агрегация будет производиться в каждый период, независимо от того, достигло ли количество потоков в памяти **Максимального размера кэш-памяти**.
Объем отброшенного трафика, %	Определяет, сколько потоков будет отброшено во время этапа агрегации первого уровня.

Все потоки, полученные от устройств, изначально хранятся в кэш-памяти. Все элементы кэша обрабатываются сервером на так называемом *этапе агрегации*, который начинается:

- В конце каждого Периода кэша-памяти
- Если количество потоков в кэше превышает Максимальный размер кэш-памяти

Этапы агрегации потоков

Этап агрегации потоков включает несколько шагов:

- **Объединение "равных" потоков в единый поток**. Все потоки в кэш-памяти или хранения баз данных обрабатываются. Сервер находит потоки с такими же адресами, портами, типами трафика, а также другими показателями источников/адресатов. Эти потоки объединяются в единый поток путем суммирования их объемов перевозок. Общее количество потоков в кэше, таким образом, уменьшается.
- **Отбрасывание наименее важных потоков**. Потоки, отражающие небольшое количество переданных байт, считаются менее важными, чем потоки, представляющие большие объемы трафика. Сервер вычисляет общий объем трафика, представленный всеми потоками в кэше, сортирует агрегированные потоки по объему трафика и отбрасывает потоки с наименьшими счетчиками трафика один за другим, пока общий объем трафика, представленный оставшимися потоками, не снизится до процентов **Объема отброженного трафика** по сравнению с первоначальным объемом трафика. На практике отбрасывание всего 1-3 процентов объема трафика будет на самом деле причиной падения 80-90% потоков. Эти потоки соответствуют приложениям, которые не генерируют много трафика и, таким образом, часто не являются важными для анализа (почтовый трафик, мессенджер приложений, трафик, вызванный мониторингом сети и т.д.)
- **Перемещение потоков в кэш более низкого уровня**. После того, как потоки объединяются и менее важные потоки отбрасываются, число потоков в большинстве случаев сильно снижается. Остальные потоки затем переносятся в кэш более низкого уровня:
  - Потоки, содержащиеся в кэше памяти, переносятся в систему хранения базы данных первого уровня
  - Потоки, содержащиеся в системе хранения базы данных, переносятся в систему хранения более низкого уровня (т.е. с первого уровня на второй уровень)

Системы хранения потоков базы данных

Системы хранения потоков базы данных - это выделенные таблицы баз данных, содержащие постоянно хранимые потоки. Существует немало систем хранения, называемых "уровни", каждая система хранения во многих смыслах подобна кэшу потоков в памяти.

Каждая система хранения конфигурируется по:

- **Периоду хранения**, т.е. периоду этапов агрегации потоков для этой системы хранения. Эта опция подобна **Периоду кэш-памяти** кэша потоков в памяти.
- Проценту **Объема отброшенного трафика**, используемого для отбрасывания менее важных потоков на этапе агрегации.

Системы хранения потоков базы данных настраиваются через таблицу **Конфигурация системы хранения базы данных**. Вот как может выглядеть таблица конфигурации системы хранения по умолчанию:

Период хранения	Объем отброшенного трафика, %
5 минут	1
1 день	5
1 месяц	5
1 год	5

После завершения этапа агрегации кэша потоков в памяти, все потоки отправляются в систему хранения базы данных первого уровня. Каждые пять минут все потоки агрегируются, 1% из наименее важных потоков отбрасывается, а оставшиеся потоки переносятся в кэш второго уровня, где они остаются в течение 1 дня, и так далее.

Агрегирование данных потока в настоящее время не работает с встроенным NoSQL хранилищем из-за использования родного функционала движков БД.

19.1.11.8 Визуализация потоков трафика

Основной задачей модуля декомпозиции трафика NetFlow является визуализация различных аспектов трафика сети.

Главной точкой входа визуализации трафика NetFlow является инструментальная панель 912 **Обзор NetFlow трафика**. Эта инструментальная панель включает в себя виджеты и таблицы, отображающие структуру и объем сетевого трафика. Инструментальная панель изначально представляет обзор всего сетевого трафика. Она предлагает фильтр для отбора и отображения типов трафика в соответствии с их датой/временем, направлением, сенсором, интерфейсом, приложением, страной, адресом источника/назначения, протоколом, типом сервиса, автономной системой и так далее.

Инструментальная панель обзора трафика включает несколько круговых диаграмм и таблиц:

- **Фильтр** отображает текущие настройки всех Топ 10 виджетов. Позволяет задать выборку по различным критериям. Например: по нескольким портам(22 80-100), по подсетям(192.168.75.0/24) или диапазону(192.168.1.1-192.168.1.10).
- **Топ 10 сенсоров** отображает статистику сенсоров по объему переданного трафика.
- **Топ 10 интерфейсов** показывает интерфейсы сенсоров, передавших много трафика.
- **Топ 10 сеансов связи** показывает пары узлов с наиболее активным обменом трафика.
- **Топ 10 приложений** показывает приложения/порты, которые произвели/потребили много трафика.
- **Топ 10 стран** отображает статистику трафика по сенсорам. Она отдельно показывает частный сетевой трафик.
- **Топ 10 узлов** отображает объем трафика для самых активных узлов сети (независимо от их расположения в локальной, глобальной сети или Интернете).
- **Топ 10 протоколов** показывает распределение трафика по протоколам.
- **Топ 10 типов сервиса** отображает статистику трафика, сгруппированную по типам сервисов, заданным в заголовках IP-пакетов.
- **Топ 10 автономных систем** отображает распределение объема трафика по автономным системам. Эта информация предоставляется только определенными моделями экспортеров потоков и протоколов сбора потоков.

Вторая инструментальная панель, используемая для анализа трафика - это инструментальная панель **Конечная точка NetFlow**. Она позволяет просматривать подробную информацию о трафике для отдельного узла сети. Инструментальная панель визуализирует темпоральное распределение трафика при помощи нескольких комбинированных диаграмм:

- Количество переданных данных
- Количество переданных пакетов
- Топ 5 приложений
- Топ 5 протоколов
- Топ 5 сеансов связи

- Топ 5 типов сервиса
- Топ 5 стран источников
- Топ 5 стран назначения
- Топ 5 автономных систем

19.1.11.9 Оптимизация производительности обработки NetFlow

Обработка NetFlow в больших сетях требует правильного подхода как к экспорту потоков сенсоров, так и к конфигурации обработки потоков точек сбора данных. Этот раздел описывает несколько советов по конфигурации, которые могут повысить производительность точек сбора данных, т.е. экземпляр AtomMind Network Manager, используемый для обработки потоков.

Переключение на разрешение имени хоста по запросу

Включение этой опции активирует выполнение разрешения имени хоста только во время визуализации данных потока. Это уменьшает как загрузку сетевой инфраструктуры, так и использование ресурсов сервера сбора данных.

См. подробности в Конфигурирование DNS и разрешение имени NetBIOS 1872.

Ограничение сбора потоков

AtomMind Network Manager обеспечивает различные опции для гибкой конфигурации сбора потоков. Например, вы можете настроить, с каких сенсоров и их индивидуальных сетевых интерфейсов нужно собирать данные.

См. подробности в Конфигурирование учетных записей устройств сенсоров NetFlow 1871.

Тонкая настройка агрегации данных

Возможно управлять процентами отброшенных пакетов на каждом шаге агрегирования потоков. Это позволяет уменьшать общее количество данных потоков, сохраняемых в базе данных путем отбрасывания потоков, которые не соответствуют большим объемам трафика.

См. подробности в Агрегация данных потоков 1872.

19.1.12 Мониторинг баз данных

AtomMind Network Manager обеспечивает мониторинг любой системы управления базами данных, совместимой с технологией JDBC. Мониторинг баз данных реализуется при помощи драйвера баз данных SQL 647. Драйвер обеспечивает мониторинг доступности баз данных путем проверки статуса их соединения. Более того, он поддерживает выполнение SQL запросов вставить/обновить/удалить, обеспечивая богатые возможности для тщательного мониторинга реляционных баз данных.

В качестве примера рассмотрите следующую задачу мониторинга баз данных Firebird/Interbase. Допустим, мы хотим проверить кодировки символов и последовательность строковых полей в таблицах.

Соответствующий запрос SQL выглядит следующим образом:

```
select rf.rdb$relation_name relation, rf.rdb$field_name table_field,
       f.rdb$field_name field_domain, f.rdb$field_type field_type,
       cs.rdb$character_set_name character_set,
       c.rdb$collation_name collation_name
from
  rdb$fields f, rdb$character_sets cs,
  rdb$collations c, rdb$relation_fields rf
where
   cs.rdb$character_set_id = f.rdb$character_set_id and
   c.rdb$collation_id = f.rdb$collation_id and
   c.rdb$character_set_id = cs.rdb$character_set_id and
```

```
    f.rdb$field_name = rf.rdb$field_source
    and f.rdb$character_set_id is not null
    and rf.rdb$relation_name not starting with 'RDB$'
order by 1, 2
```

Предположим, имеется зарегистрированное устройство **Драйвер SQL** с `org.firebirdsql.jdbc.FBDriver` в качестве **драйвера базы данных**, `jdbc:firebirdsql:<dbHost>/3050:<dbPath>` в качестве **URL базы данных** (где `<dbHost>` и `<dbPath>` замещаются *хостом* и *путем* вашей базы данных) и правильные значения **имени пользователя базы данных** и **пароля базы данных**. Затем запрос может добавляться к **вкладке Запросы**, предоставляя, скажем, `charsets` как **Имя** и `Charsets and Collations` как **Описание**.

Как только конфигурируется запрос, результаты мониторинга базы данных (в таблице **Конфигурация устройств**) будут иметь свойство **Кодировка символов и последовательность**, содержащее результат запроса. Теперь вы можете определять неправильные кодировки данных и последовательности в соответствии с вашими собственными парвилами и добавлять тревоги или обработать их каким-либо другим образом.

AtomMind Network Manager обеспечивает эффективный коробочный мониторинг распространенных СУБД, таких как Microsoft SQL Server, Oracle Database Server, MySQL, PostgreSQL.

Конфигурирование мониторинга баз данных

Для настройки мониторинга:

- Заведите аккаунт базы данных в узле **Устройства** в Системном дереве 370
- Щелкните правой кнопкой на созданном аккаунте
- Выберите пункт **Настроить мониторинг** в контекстном меню
- Отметьте **Создать запросы** в открывшейся таблице и нажмите OK
- Удалите при необходимости не нужные запросы из таблицы или оставьте значения по умолчанию и нажмите OK
- Откройте инструментальную панель двойным щелчком по аккаунту устройства

19.1.12.1 Мониторинг MySQL

AtomMind Network Manager предлагает высококачественное коробочное решение для безагентного мониторинга MySQL без дополнительной нагрузки на вашу базу данных. Неважно, сколько баз данных вам нужно администрировать, Менеджер будет отслеживать их производительность из единой консоли, отображая всю статистику MySQL. Более того, Менеджер позволяет вам установить собственные запросы SQL для отслеживания задержек и их надлежащего выполнения.

Настройка прав доступа для мониторинга MySQL

Для мониторинга базы данных MySQL необходимо дать права на чтение таблиц схемы **information_schema**.

Метрики, доступные для мониторинга

AtomMind Network Manager отслеживает следующие метрики MySQL:

- Полная статистика подключений
- Размер баз данных
- Трафик СУБД

- Текущий статус функции
- Статистика запросов
- Эффективность кэша, запросов, ключей
- Список и состояние процессов
- Статистика блокировок
- Статус таблиц
- И многое другое

19.1.12.2 Мониторинг Oracle

AtomMind Network Manager обеспечивает непрерывный коробочный мониторинг Oracle без агентов. Менеджер отслеживает метрики Oracle в реальном времени, не оказывая дополнительной нагрузки на ваши сервера. Он имеет все желаемые инструменты для основательного мониторинга базы данных, используя коробочное решение или давая возможность визуально разрабатывать собственные инструменты мониторинга. Администраторы базы данных извлекут пользу из легкой установки контроля производительности Oracle и обнаружения узких мест.

Настройка прав доступа для мониторинга Oracle

Oracle содержит набор счетчиков производительности, доступный для администратора базы данных **SYS**, которые называются динамическими представлениями производительности. AtomMind Network Manager использует эти данные для получения метрик производительности. Только пользователь **SYS** или другой пользователь с ролью **SYSDBA** имеет доступ к динамическим таблицам производительности. Хотя эти views кажутся обычными таблицами базы данных, они ими не являются. Эти представления отдают данные о внутренних структурах диска и структур памяти. Вы можете выбрать из этих взглядов, но вы никогда не можете обновить или изменить их.

Метрики, доступные для мониторинга

Метрики базы данных Oracle, контролируемые AtomMind Network Manager, включают:

- Общую информацию о сервере и базе данных
- Физический и логический ввод-вывод
- Утилизация табличного пространства
- Использование Redo Logs
- Сегмент Rollback
- Блокировки
- Активные пользователи, сессии и занимаемые ими ресурсы
- Процессы сервера
- Эффективность буфера
- Использование глобального системного пространства (SGA)
- Использование хранилища метаданных SUSAUX
- И многое другое

19.1.12.3 Мониторинг Microsoft SQL Server

AtomMind Network Manager обеспечивает эффективный коробочный мониторинг Microsoft SQL Server.

Настройка прав доступа для мониторинга MS SQL Server

Необходимо разрешение **SELECT**, а также разрешения **VIEW SERVER STATE** и **VIEW DATABASE STATE**.

Создание устройства для мониторинга MS SQL Server

Для мониторинга Microsoft SQL Server выполните следующие действия:

1. Загрузите последнюю версию **Microsoft SQL Server JDBC Driver**. На момент написания этого раздела, она доступна на сайте http://msdn.microsoft.com/data/jdbc/.
2. Поместите `sqljdbc4.jar` в подпапку `/jar` папки установки AtomMind Server.

Это альтернативный **Драйвер jTDS JDBC** драйвера JDBC**.** Он поддерживает более ранние версии Microsoft SQL Server, заканчивая версией 10.0 (SQL Server 2005, SQL Server 2000) и Аутентификацию домена 701.

На момент написания этого раздела, он доступен на сайте http://jtds.sourceforge.net/. В случае использования **jTDS JDBC Driver,** два первых шага должны быть следующими:

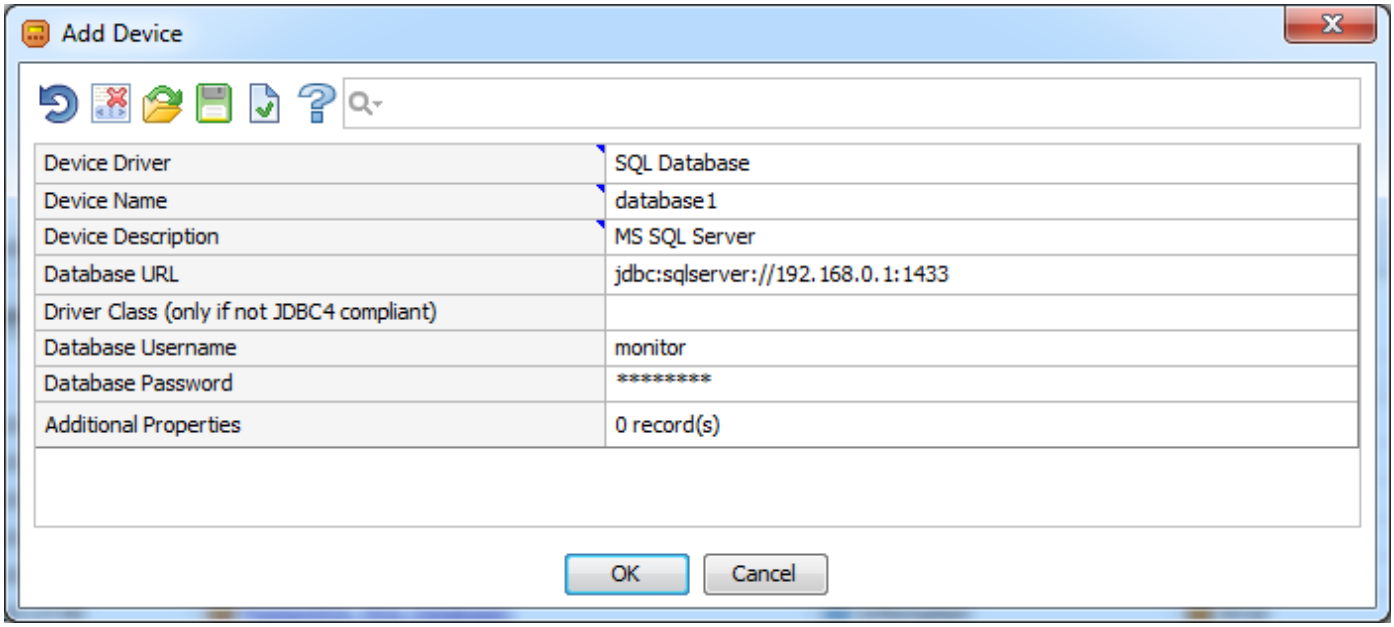
1. Загрузите последнюю версию **jTDS JDBC Driver**.
2. Поместите `jtds-X.X.X.jar` в подпапку `/jar` папки установки AtomMind Server.

3. Перезагрузите AtomMind Server.
4. Создайте новое устройство с драйвером Базы данных SQL 647 двойным щелчком по контейнеру `Devices` в системном дереве.
5. Измените имя устройства в поле `Device Name` и введите описание устройства в поле `Device Description`, если необходимо.
6. Настройка `Database URL` для базы данных Microsoft SQL Server имеет следующий формат: `jdbc:sqlserver://[serverName[\instanceName][:port]];databasename=dbname[;property=value[;property=value]]`, где `jdbc:sqlserver://` известен как подпротокол, и он постоянен, `serverName` – имя DNS или IP-адрес сервера, к которому необходимо подключиться (может быть `locahost`), `instanceName` – это экземпляр, к которому необходимо подключиться при помощи `serverName` если не определено, устанавливается подключение к экземпляру по умолчанию), `portNumber` является портом, к которому необходимо подключиться при помощи `serverName` (по умолчанию – 1433). Например, если ваш Microsoft SQL Server запущен на 192.168.0.1, на порте по умолчанию используйте данный URL для подключения к экземпляру по умолчанию: `jdbc:sqlserver://192.168.0.1:1433`. Вы также можете не пользоваться портом по умолчанию и использовать вместо него порт `jdbc:sqlserver://192.168.0.1`.

В случае использования **Драйвера jTDS JDBC**, `Database URL` должен быть следующим:

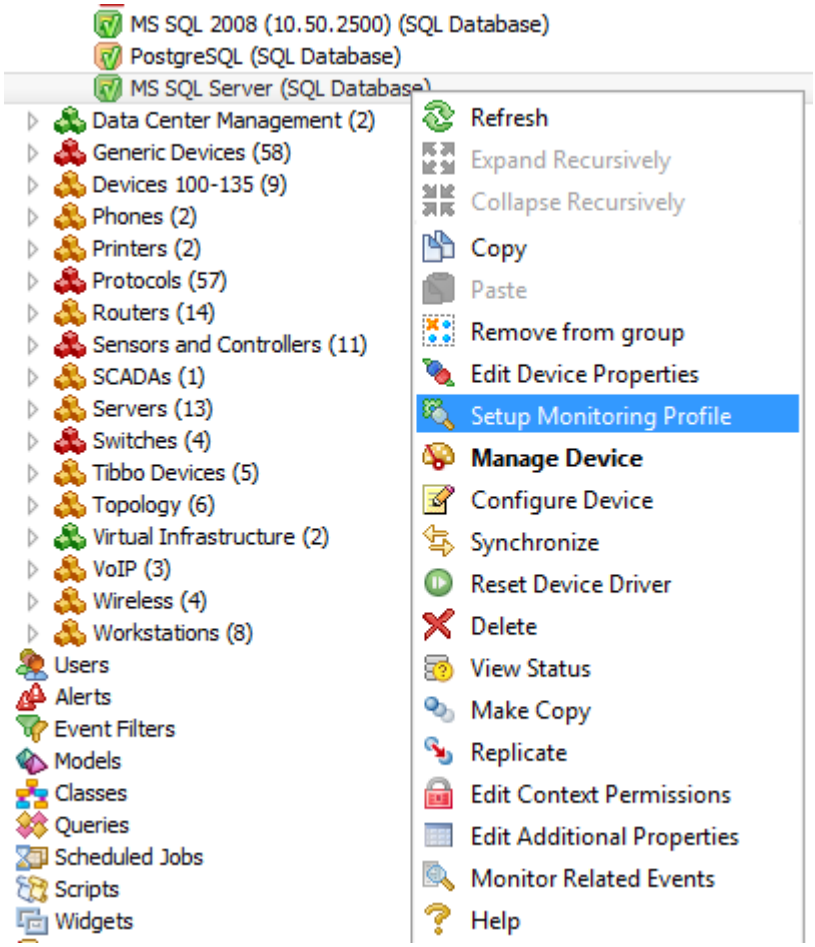
`jdbc:jtds:sqlserver://192.168.0.1:1433`

7. Введите имя и пароль пользователя базы данных с необходимыми правами доступа в соответствующих полях.

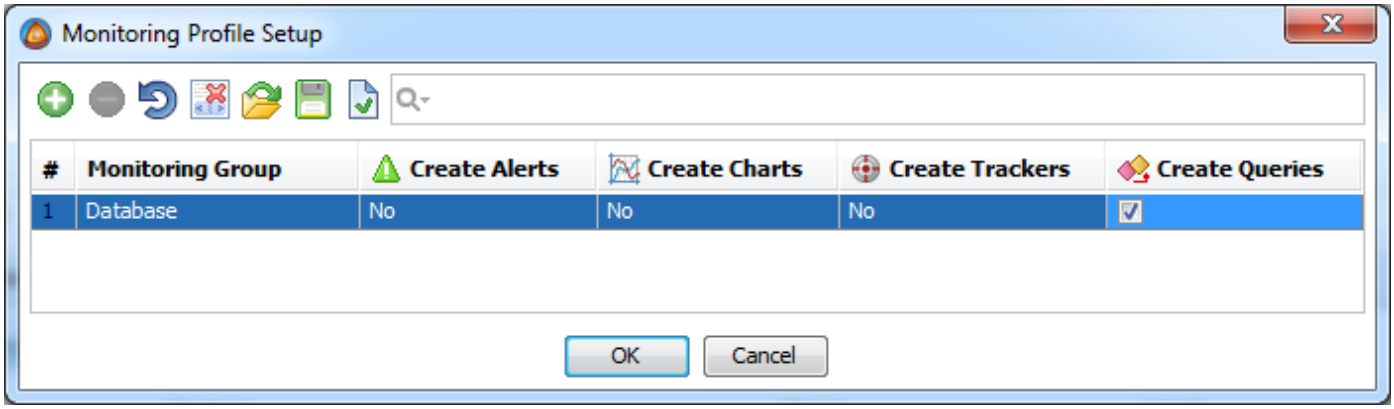


8. Нажмите кнопку `OK` и в меню `Device Properties` снова нажмите кнопку `OK`. Если подлючение прошло успешно, новое устройство будет доступно в системном дереве с иконкой зеленого цвета.

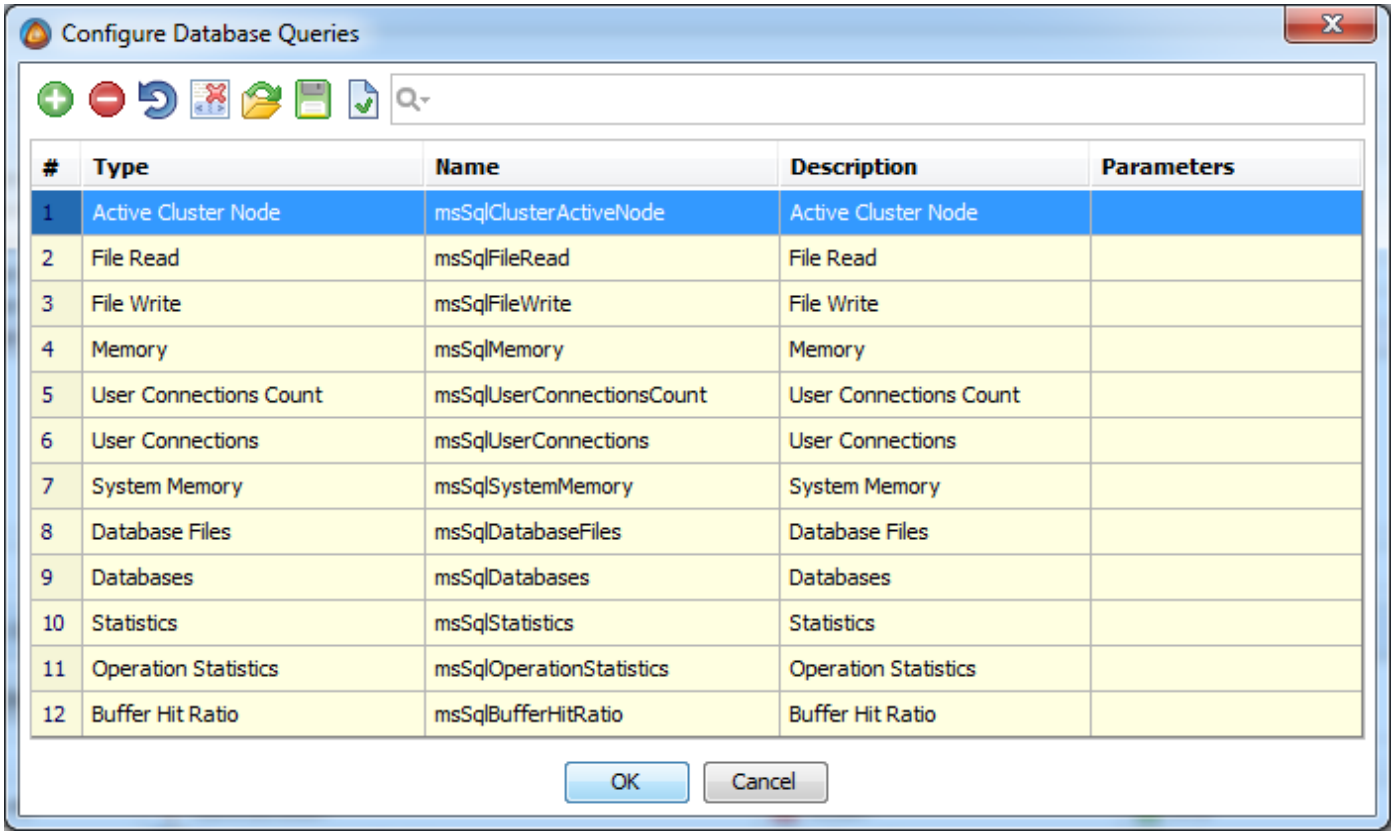
9. Нажмите правую кнопку мыши по устройству в системном дереве и выберите `Setup Monitoring Profile` из списка доступных действий устройства.



10. В новом окне поставьте галочку в поле `Create Queries` и нажмите кнопку `OK`.



11. При необходимости удалите некоторые запросы или просто нажмите кнопку `OK`.



#	Type	Name	Description	Parameters
1	Active Cluster Node	msSqlClusterActiveNode	Active Cluster Node	
2	File Read	msSqlFileRead	File Read	
3	File Write	msSqlFileWrite	File Write	
4	Memory	msSqlMemory	Memory	
5	User Connections Count	msSqlUserConnectionsCount	User Connections Count	
6	User Connections	msSqlUserConnections	User Connections	
7	System Memory	msSqlSystemMemory	System Memory	
8	Database Files	msSqlDatabaseFiles	Database Files	
9	Databases	msSqlDatabases	Databases	
10	Statistics	msSqlStatistics	Statistics	
11	Operation Statistics	msSqlOperationStatistics	Operation Statistics	
12	Buffer Hit Ratio	msSqlBufferHitRatio	Buffer Hit Ratio	

12. В `Device Properties` созданные запросы доступны во вкладке `Queries`. Настройки опций синхронизации 502 и Каналы статистики 718 переменной уже сконфигурированы и доступны также в соответствующих вкладках.

Метрики, доступные для мониторинга

Ключевые метрики Microsoft SQL Server:

- Эффективность кэша и буфера
- Утилизация памяти и эффективность использования
- Работа со страницами
- Общая статистика Microsoft SQL Server, статистика операций, блокировок
- Статистика методов доступа
- Активные подключения пользователей и сессии
- Подсистема ввода – вывода
- Утилизация дискового пространства
- И другие

19.1.12.4 Мониторинг PostgreSQL

AtomMind Network Manager включает коробочные инструментальные панели мониторинга PostgreSQL. Он отправляет заблаговременные предупреждения о потенциальных проблемах и собирает подробную статистику, уменьшая время поиска узких мест. На основе собранных данных можно запланировать перераспределение ресурсов, расширение аппаратных средств до того, как начнут появляться проблемы. Преимущество AtomMind Network Manager заключается в том, что мониторинг СУБД можно одинаково хорошо проводить в комплексе с мониторингом аппаратной части, мониторингом производительности операционных систем и каналов связи. К тому же у администраторов появляются широкие возможности по созданию собственных произвольных SQL запросов к PostgreSQL.

Настройка прав доступа для мониторинга PostgreSQL

Для мониторинга баз данных PostgreSQL необходимо дать права на чтение следующих системных таблиц:

- pg_stat_activity
- pg_stat_database
- pg_indexes
- pg_locks
- pg_stat_database
- pg_stat_all_tables
- pg_trigger

Метрики, доступные для мониторинга

Метрики мониторинга PostgreSQL включают:

- Активность пользователей
- Статистика по таблицам
- Статистика блокировок
- Статистика индексов
- Статистика триггеров
- Размер баз данных
- Статистика операций с базами данных
- Эффективность буфера
- И другие

19.1.13 Мониторинг файловой системы

AtomMind Network Manager предоставляет функциональность для мониторинга файловой системы на компьютере с запущенным AtomMind на AtomMind Server. Это включает проверку существующих файлов и папок, их содержимого, размера, атрибутов и пр.

См. подразделы Мониторинг локальной папки 1882 и Мониторинг локального файла 1881.

19.1.13.1 Мониторинг локального файла

Мониторинг локального файла выполняет Драйвер локального файла 576. Он позволяет отслеживать файлы, расположенные на сервере:

- проверяет наличие того или иного файла
- позволяет получить атрибуты файла:
  - *время последних изменений*
  - *размер*
- подсчитывает *контрольную сумму* файла (используя алгоритм Adler-32)
- загружает содержимое файла и делает его доступными для анализа и редактирования при помощи инструментария обработки данных AtomMind (например, отправки сообщения при обнаружении определенной конфигурации).

Мониторинг файла может оказаться полезным и эффективным инструментарием при выполнении ряда административных задач:

- Мониторинг событий на хосте - наличие/отсутствие отдельного файла.
- Мониторинг доступности сервиса или приложения путем отслеживания времени последнего изменения и/или других атрибутов отдельного файла.
- Мониторинг активности приложения путем анализа содержимого файла.

- Управление приложениями путем изменения содержимого их конфигурационных файлов.

19.1.13.2 Мониторинг локальной папки

Мониторинг локальной папки выполняет Драйвер локальной папки[577]. Он позволяет отслеживать папки, расположенные на сервере, и предлагает следующие опции:

- проверить наличие папки по заданному пути на сервере
- получить атрибуты папки, а именно:
  - *время последнего изменения*
  - *подсчет файлов*
  - *размер,* т.е. общий размер файлов в папке (выбранных фильтром)
- загружать и изучать содержимое папки (файлы и подпапки)

Содержимое папки можно анализировать рекурсивно (включая подпапки) или не рекурсивно. Кроме этого, обрабатываемые элементы можно отфильтровать, используя специальный символ или обычное выражение.

Мониторинг локальной папки может оказаться полезным и эффективным инструментарием при выполнении ряда сетевых административных задач:

- Отслеживание наличия/отсутствия той или иной папки.
- Мониторинг доступности сервиса или приложения путем отслеживания времени последнего изменения и/или других атрибутов определенной папки.
- Удаленный анализ вывода приложения или активности приложения путем изучения содержимого папки.
- Проверка общего размера файлов журнала приложения и предупреждение о необходимости выполнения их резервной копии.

19.1.13.3 Мониторинг файла журнала

Существует несколько методов мониторинга файлов журнала различных приложений для обнаружения ошибок и создания предупреждений:

- Журналы, расположенные на ПК AtomMind Server, можно отследить при помощи драйвера мониторинга локального файла[1881].
- Доступ к файлам журнала, расположенным на удаленных компьютерах, можно получить, используя драйвер мониторинга локального файла[1881], перенеся удаленные папки в локальную файловую систему средствами операционной системы (функция **Подключить сетевой диск** для Microsoft Windows; NFS или Samba для Linux).
- Удаленные журналы можно проверить на ошибки при помощи bash-сценария, который периодически выполняется по SSH[1903]. Пример такого скрипта `grep error_text /var/log/some_log_file.log | tail -n1.`

19.1.14 Консолидация и мониторинг событий

AtomMind Network Manager может собирать, обрабатывать, и хранить уведомления от различных источников о событиях в сети и ошибках.

Общие концепции корреляции событий и анализа первопричин описаны в разделе Управление событиями[761].

AtomMind Network Manager поддерживает следующие источники:

- сообщения Syslog[1883]
- SNMP-ловушки[1886]
- WMI-события[1887]
- Журнал событий Windows[1887].
- Любые другие события, предоставляемые различными драйверами устройств[518].

19.1.14.1 Маскировка событий и зависимости устройств

AtomMind Network Manager имеет особую реализацию Маскировки событий 762, решающую следующие задачи:

- Деактивация опроса хостов сети, которые недоступны на AtomMind Server из-за сбоев промежуточных устройств
- Блокировка инициации тревог устройствами, которые в данный момент недоступны из-за сбоев промежуточных устройств

Технология маскировки событий основана на отношениях устройство-родительское устройство, которые определены моделью 810 **Подключение устройств**, включенной в дистрибутив AtomMind Network Manager. Эта модель:

- Добавляет таблицу **Устройства-родители** и переменную **Выражение подключения** к каждому устройству хоста сети 593.
- Устанавливает свойство выражение зависимости 512 каждого заново создаваемого устройства хоста сети, чтобы проверка его подключения выполнялась в начале каждого цикла синхронизации 514. Синхронизация устройства (опрос) деактивируется, если проверка подключения не выполняется успешно.
- Настраивает функции проверки подключения, так что устройство считается недоступным из-за сбоев устройств-родителей, если происходит сбой пинга самого устройства и его родителей.

Конфигурация маскировки событий

Базовая конфигурация маскировки событий/тревог и блокировки очень простая:

- Кликните правой кнопкой мыши на устройстве Хост сети
- Выберите **Редактировать дополнительные свойства**
- Перейдите во вкладку **Подключение устройства**
- Откройте таблицу **Устройства-родители**
- Заполните таблицу устройствами сети, которые являются (или могли бы являться -- в случае динамической маршрутизации) следующим шагом в маршруте сети от конфигурируемого устройства до сервера AtomMind Network Manager

Модель подключения устройств

Эта модель 810 активирует маскировку событий и блокировку тревог, инициируемых устройствами, которые недоступны из-за сбоев некоторых промежуточных сетевых устройств.

19.1.14.2 Syslog

Syslog - это стандарт для передачи журнальных сообщений в сети IP. Этот протокол обычно используется для управления компьютерными системами и выполнения проверки безопасности путем интеграции данных журнала из систем различного типа в центральный репозиторий.

Syslog - это клиент серверный прокол: *отправитель Syslog* отправляет небольшое текстовое сообщение (*сообщение Syslog*) *получателю Syslog*. AtomMind Network Manager может выполнять и ту, и другую роль, предоставляя возможность отслеживать сообщения Syslog 1883, а также генерировать и отправлять сообщения Syslog 1885. Кроме того, осуществляется поддержка автоматического обнаружения источников сообщений Syslog 1885 .

19.1.14.2.1 Консолидация и мониторинг событий Syslog

Осуществляя **мониторинг событий Syslog,** плагин Syslog 1797 активизирует **получателя** *Syslog,* который получает *сообщения Syslog*, собирает их и конвертирует в события 73 AtomMind. События Syslog можно обрабатывать, хранить, отслеживать, отображать и отфильтровывать как любые другие события AtomMind. Подробное описание протокола Syslog можно найти в RFC 5424.

Syslog-мониторинг настраивается через параметры конфигурации плагина Syslog 1804. Когда получено Syslog-сообщение, оно анализируется, генерируется соответствующее событие AtomMind. Структура и правила преобразования данных представлены в следующей таблице:

Поле события	Имя поля события	Тип	Описание
Хост источника	source	Строка	Адрес источника сообщения, т.е. адрес, откуда было получено Syslog-сообщение.
Уровень	level	Целое число	Исходный *уровень*, заданный в Syslog-сообщении. За списком значений уровня обратитесь к RFC 5424.
Средство	facility	Целое число	Исходное *средство*, заданное в Syslog-сообщении.
IP-адрес или имя хоста	host	Строка	Хост, заданный в Syslog-сообщении (обычно генератор сообщения).
Сообщение	message	Строка	Текст Syslog-сообщения, сообщающий информацию о событии.
Отметка времени	timestamp	Данные	Отметка времени, заданная в Syslog-сообщении.

Событие AtomMind генерируется с уровнем, который преобразуется в исходной Syslog-критичности при помощи таблицы преобразования 1804, заданной в параметрах настройки плагина Syslog.

19.1.14.2.1.1 Syslog-тревоги

AtomMind Network Manager предоставляет ряд *предварительно настроенных* тревог 779 для сообщений Syslog, описанных ниже.

Триггеры Syslog-тревог содержат не более трех условий, связанных логическими операторами "И". Они осуществляют проверку, если сообщение:

- содержит определенную строку
- имеет свойство, равное определенному значению
- имеет уровень, равный определенному значению.

Ниже приведенная таблица описывает Syslog-тревоги и условиях их триггеров.

Имя	Описание	Состояни триггера		
		Сообщение содержит	Свойство	Уровень
syslogAlertFailedLogin	Авторизация неуспешна	*FAILED LOGIN*	Безопасность (4)	Уведомление (5)
syslogAlertKernelAlert	Тревожное сообщение от ядра		Ядро (0)	Тревога (1)
syslogAlertKernelEmergency	Авария на ядре		Ядро (0)	Авария (0)
syslogAlertMailCritical	Критическая ситуация с почтой		Mail (2)	Критическая ситуация (2)
syslogAlertMailEmergency	Аварийная ситуация с почтой		Mail (2)	Авария (0)
syslogAlertSecurityOrAuthorizationAlert	Тревожное сообщение о безопасности или авторизации		Безопасность (4)	Тревога (1)

syslogAlertSecurityOrAuthorizationEmergency	Аварийная ситуация с безопасностью или авторизацией		Безопасность (4)	Авария (0)
syslogAlertFtpLogOut	Выход из FTP	*FTP session closed*	FTP (11)	Информационное сообщение (6)
syslogAlertFtpLogIn	Вход в FTP	*FTP LOGIN FROM*	FTP (11)	Информационное сообщение (6)
syslogAlertSuperuserLoginSuccess	Успешная авторизация суперпользователя	*opened*	Безопасность (4)	Уведомление (5)
syslogAlertUserLoginSuccess	Успешная авторизация пользователя	*session opened*	Безопасность (4)	Информационное сообщение (6)
syslogAlertDaemonAlert	Тревожное сообщение от демона		Демоны (3)	Тревога (1)
syslogAlertDaemonEmergency	Аварийная ситуация на демоне		Демоны (3)	Авария (0)
syslogAlertUserLevelAlert	Тревожное сообщение уровня "пользователь"		Пользователь (1)	Тревога (1)
syslogAlertUserLevelEmergency	Аварийная ситуация на уровне "пользователь"		Пользователь (1)	Авария (0)

Обычные тревожные сообщения для Syslog-сообщений можно легко создать с учетом пользовательских требований, используя в качестве шаблона описанные тревоги.

19.1.14.2.1.2 Автоматическое обнаружение источника Syslog-сообщения

IP хосты, генерирующие Syslog-сообщения, можно считать потенциально управляемыми элементами. У Syslog-плагина AtomMind Network Manager есть опция активировать автоматическое обнаружение источников Syslog-сообщений[1805]. Если она включена, каждое создание Syslog-события может запускать процесс обнаружения для хоста, который отправил соответствующее Syslog-сообщение.

19.1.14.2.2 Отправка Syslog-сообщений

AtomMind Network Manager может генерировать и отправлять Syslog-сообщения. Эту операцию выполняет действие[87] Отправить **Syslog-**сообщение (`sendSyslogMessage`), которое находится в корневом контексте[1559].

Формат ввода

Действие можно настроить, используя параметры следующего формата:

Имя	Тип	Описание	Информация
protocol	Строка	Протокол	Транспортный протокол, который используется для доставки сообщения (`UDP` или `TCP`).
host	Целое число	Целевой IP-адрес или имя хоста	Адрес получателя (`127.0.0.1` по умолчанию).
port	Строка	Порт	Порт, на который будет отправлено сообщение (`514` по умолчанию).

message	Целое число	Сообщение	Текст сообщения.
facility	Целое число	Средство	Средство сообщения.
level	Целое число	Критичность	Уровень сообщения.
sendName	Логическое значение	Отправить локальное имя	Будет ли имя хоста источника включено в сообщение.
sendTimestamp	Логическое значение	Отправить локальную отметку времени	Будет ли отметка времени включена в сообщение.

19.1.14.3 SNMP-уведомления

AtomMind Network Manager получает и обрабатывает **SNMP-ловушки** (см. Получение SNMP-ловушек 1846). Полученная ловушка конвертируется во внутреннее `trap` событие 73 AtomMind. Ловушки можно использовать для мониторинга важных событий, которые происходят с управляемыми объектами. AtomMind Network Manager включает ряд предварительно настроенных тревог для некоторых важных событий SNMP-ловушек, см ниже подраздел Тревоги SNMP-ловушек 1886.

В то же время сам AtomMind Network Manager может генерировать и отправлять SNMP-ловушки, уведомляя другие хосты в сети о важных событиях. Дополнительную информацию см в разделе Отправка SNMP-ловушек 1848.

19.1.14.3.1 Тревожные сообщения SNMP

AtomMind Network Manager предоставляет ряд предварительно настроенных тревог для SNMP-уведомлений, как описано ниже.

Триггеры тревог могут содержать два состояния: для уведомлений SNMP версии 1, а также для версии 2c и выше. Следующая таблица описывает тревоги SNMP-ловушки и условия триггера.

Имя	**Описание**	**Условия триггера**			
		для SNMP-ловушек версии 2c	**SNMP-ловушки версии 1**		
		ID объекта ловушки	**Общий ID**	**Специальный ID**	**Производитель**
Ошибка авторизации	Авторизация завершена неуспешно. Адрес источника включен в тревожное сообщение.	*authenticationFailure*	*4*		
Настройка маршрутизатора изменена	Настройка маршрутизатора изменена.		*6*	*1*	*1.3.6.1.4.1.9* or *cisco*
Уведомления о вентиляторах Cisco	Один из вентиляторов не работает.	*1.3.6.1.4.1.9.9.13.3.0.4* or *ciscoEnvMonFanNotification*			
Остановка Cisco	Монитор среды определяет, что контрольная точка достигает критического состояния и собирается инициировать остановку.	*1.3.6.1.4.1.9.9.13.3.0.1* or *ciscoEnvMonShutdownNotification*			
Уведомление о температуре Cisco	Температура, измеряемая на данной контрольной точке, выходит за допустимую норму.	*1.3.6.1.4.1.9.9.13.3.0.3* or *ciscoEnvMonTemperatureNotification*			

Уведомление об электрическом напряжении Cisco	Электрическое напряжение, измеряемое на данной контрольной точке, выходит за допустимую норму.	*1.3.6.1.4.1.9.9.13.3.0.2* or *ciscoEnvMonVoltageNotification*			
Уведомление о резервном источнике питания для Cisco	Резервный источник питания отключен.	*1.3.6.1.4.1.9.9.13.3.0.5* or *ciscoEnvMonRedundantSupplyNotification*			
Тревога о состоянии принтера	Обнаружено критическое событие принтера.	*1.3.6.1.2.1.43.18.2.0.1* or *printerV2Alert*			
Соединение прервано	Интерфейс близок к состоянию отключения.	*linkDown*	*2*		
Соединение установлено	Интерфейс вышел из состояния "отключен".	*linkUp*	*3*		
"Холодный" старт	Холодный старт, т.е. перезапуск с возможными изменениями в настройке.	*coldStart*	*0*		
"Теплый" старт	Теплый старт, т.е. перезапуск с возможными изменениями в настройке.	*warmStart*	*1*		
Потеря соседа EGP	Сосед EGP, для которого устройство одного уровня утрачено, и между ними больше нет одноуровневой связи.		*5*		

При необходимости пользователи могут создать собственные тревоги для SNMP-ловушек, используя в качестве шаблона уже существующие.

19.1.14.4 События WMI

WMI использует средство подписки для доставки уведомлений о важных событиях в управляемой системе. AtomMind Network Manager использует сервисы, которые доставляет **драйвер устройства WMI** для подписки и получения WMI-событий. См. главу WMI 661.

19.1.14.5 События журнала Windows

Windows NT предлагает стандартный, централизованный способ записывать важные события программного и аппаратного оборудования в один пункт сбора информации, именуемый **Журналом событий**. AtomMind Network Manager можно использовать для мониторинга активности компьютеров, с установленной на них ОС Windows, настраивать журналы событий для трансляции Windows-событий в SNMP-ловушки и отправки их на сервер AtomMind Network Manager. Дополнительную информацию о настройке см. в разделе Получение сообщений журнала событий Windows 1810.

19.1.14.6 Преднастроенные фильтры событий

AtomMind Network Manager включает в себя несколько встроенных фильтров событий 762:

Все события

Фильтр **Все события** показывает все важные события системы и сети. Смотрите детали в соответствующей секции[773] главы Фильтры событий.

Устройство обнаружено

Показывает события, сформированные при обнаружении нового устройства.

Прерывание сервисов

Фильтр **Прерывание сервисов** показывает события прерывания[595], сформированные устройствами хоста сети. Следующие колонки видны для событий прерывания:

- **Время сервера,** когда случилось событие.
- **Контекст**, представляющий хост сети.
- **Уровень** события.
- **Сервис**, формирующий событие.
- **Время начала** прерывания.
- **Длительность** прерывания в секундах.
- **Причина** прерывания, предоставленная сервисом.

Ловушки SNMP

Фильтр **Ловушки SNMP** показывает события, сформированные при получении Ловушки SNMP[1846]. Представлены следующие колонки:

- **Время сервера** события.
- **IP-адрес агента или имя хоста**, ссылающийся на источник ловушки.
- **Тип объекта-источника** поля ловушки.
- **Идентификатор стандартного типа ловушки**.
- **Время беспрерывной работы** устройства, которое создало ловушку (в 1/100 секунды).
- **Привязки переменных**, ассоциируемых с ловушкой.

Syslog

Фильтр **События Syslog** показывает события, соответствующие сообщениям Syslog[1883], полученным сервером AtomMind Network Manager. Следующие колонки являются видимыми:

- **Время сервера** события.
- **Уровень** события.
- **Временная метка** сообщения Syslog.
- Текст **сообщения**.
- **IP-адрес или имя хоста**, предоставленные сообщением Syslog.
- **Категория**, определенная в сообщении Syslog.
- **Критичность** сообщения Syslog.

19.1.15 Мониторинг сервисов/приложений

AtomMind Network Manager предоставляет возможность производить мониторинг работоспособности[1798] различных сервисов и приложений по протоколам TCP и UDP.

Для того, чтобы запустить мониторинг отдельного приложения/сервиса, следует активировать соответствующий монитор в конфигурации сетевого устройства хоста. Это можно выполнить вручную, во время настройки отдельного устройства[2103] (а также нескольких устройств одновременно), или автоматически, как часть процесса обнаружения[1811].

Сводку состояния о доступности/работоспособности для всех устройств/сервисов можно отслеживать в Таблице статуса сервисов[1854], которая находится в Избранном[2186].

19.1.15.1 Мониторинг веб-сервера

AtomMind Network Manager предоставляет готовые опции для мониторинга веб-сервера:

- Мониторинг веб-страниц и проверка URL-контента 1889 путем выполнения HTTP-запросов. Этот мониторинг не зависит от типа веб-сервера (IIS, Apache, SunONE и др.). Это наиболее полезный мониторинг функционирования веб-сайта, но он также может использоваться для организации функционального тестирования веб-сервера -- см. раздел Мониторинг HTTP 1889 для получения более подробной информации
- Отслеживание времени загрузки 1863 веб-страницы и отсылка тревог при превышении порога
- Комплексный мониторинг серверов приложений и движков баз данных 1875
- Мониторинг нагрузки на процессор, памяти, использования пропускной способности и других ключевых показателей веб-сервера
- Сбор ошибок веб-сервера через Syslog 1883
- Анализ файлов журнала 1882 веб-сервера

Тревоги веб-сервера и мониторинга веб-страниц

Имя тревоги	Условие активации	Комментарии
Неправильный ответ HTTP	Код ответа HTTP, полученный у **веб-сайта**, отличается от обозначенного.	Для создания используйте группу **HTTP** в действии Настройка профайла мониторинга 1808.
Контент ответа HTTP изменился	Контент, полученный у **веб-сайта**, изменился.	Для создания используйте группу **HTTP** в действии Настройка профайла мониторинга 1808.
Контент ответа HTTP	Ответ HTTP, полученный у **веб-сайта**, содержит обозначенную строку.	Для создания используйте группу **HTTP** в действии Настройка профайла мониторинга 1808.

19.1.15.1.1 Мониторинг веб-страниц

Драйвер устройства HTTP 569 и Сервис мониторинга HTTP 600, включенный в Драйвер устройства хоста сети 593, позволяют пользователю выполнять произвольные HTTP-запросы и анализировать URL-контент (веб-страницу).

Для настройки мониторинга URL:

- Создайте **устройство HTTP** или активируйте **HTTP-сервис** в устройстве хоста сети
- Настройте порт HTTP/HTTPS, отслеживаемый URL, тип запроса (GET или POST) и параметры, опциональную информацию об авторизации, пользовательские заголовки HTTP, агенты и тайм-аут пользователя
- Откройте настройку устройства (выберите **Настроить** из контекстного меню) для просмотра статуса HTTP-сервиса и контента загружаемой веб-страницы
- Используйте любые категории системы (такие как тревоги 779, модели 810, скрипты 879 и т.д.) для подробного анализа, например, проверки кода ответа, заголовков HTTP, формы ответа и т.д.

19.1.15.1.2 Мониторинг веб-приложений

Драйвер устройства Веб-транзакция 659 предоставляет широкие возможности мониторинга веб-приложений. Тестирование и мониторинг веб-приложений выполняется скриптами. **Selenium** помогает в управлении браузерами и создании тестовых скриптов.

Selenium – это портативный фреймворк для тестирования ПО веб-приложений.

Инструмент мониторинга веб-приложений выполняется как драйвер Device, поэтому для начала, вам необходимо создать веб-транзакцию 659 Device. В процессе добавления Device вам нужно определить скрипт, который будет выполняться при каждой синхронизации.

Запуск тестового скрипта требует наличия установленного браузера **Mozilla Firefox** 42.0 или более ранней версии наряду с AtomMind Server.

Создание скрипта

Selenium IDE предоставляет возможность создания тестовых скриптов. Он реализуется как расширение **Mozilla Firefox**, которое позволяет записывать, редактировать и отлаживать скрипты.

Более детальную информацию о **Selenium IDE** вы можете найти на сайте http://www.seleniumhq.org/docs/.

Selenium IDE требует дополнительной конфигурации после установки. Включите экспериментальные возможности: **Options -> Options... -> Enable experimental features**.

Затем установите новый заголовок и нижний колонтитул для формата **Java / JUnit 4 / WebDriver**, перейдите к: **Options -> Options... -> Formats -> Java / JUnit 4 / WebDriver**.

Установите заголовок:

```
import org.openqa.selenium.support.events.EventFiringWebDriver;
import org.openqa.selenium.By;
import com.tibbo.linkserver.plugin.device.web.transaction.Screenshot;


public class WebTransactionClass {
  private String baseUrl = "${baseURL}";
  public void test (EventFiringWebDriver driver, Screenshot screenshot) throws
Exception {
```

И установите нижний колонтитул:

```
  }
}
```

Эти настройки позволяют преобразовывать записанные действия в формат, совместимый с AtomMind Server. Теперь вы можете записывать действия.

После записи действия, вы должны добавить источник скрипта в нужном формате, перейдите к: **Options -> Format -> Java / JUnit 4 / WebDriver**. Затем скопируйте источник скрипта из вкладки **Источник** Selenium IDE и сохраните его в свойстве **Скрипт** настроек Device.

Пример тестового скрипта:

```
import org.openqa.selenium.support.events.EventFiringWebDriver;
import org.openqa.selenium.By;
import com.tibbo.linkserver.plugin.device.web.transaction.Screenshot;


public class WebTransactionClass {
  private String baseUrl = "http://example.com/";
  public void test (EventFiringWebDriver driver, Screenshot screenshot) throws
Exception {
    driver.get(baseUrl + "/");
    driver.findElement(By.linkText("Technology")).click();
    screenshot.takeScreenshot(driver);
    driver.findElement(By.linkText("Industries")).click();
    driver.findElement(By.linkText("Solutions")).click();
  }
```

```
}
```

Результаты выполнения скрипта сохраняются в **Действиях** и **Скриншотах** настроек Device при каждой синхронизации 514.

Драйвер веб-транзакций педоставляет легкий способ получения скриншотов. Вы можете вставить данную строку в скрипт, чтобы сделать скриншот: `screenshot.takeScreenshot(driver);`

19.1.15.1.3 Мониторинг web-сервера Apache

Подробный мониторинг веб-сервера Apache позволяет видеть некоторые внутренние детали (configuration, status, performance, error statistics) of *Apache HTTP Server.* Those details are available via SNMP in the presence of модуль Mod-Apache-Snmp.

Для активации мониторинга Apache:

- Установите модуль Mod-Apache-Snmp на веб-сервере Apache, если он еще не установлен
- Убедитесь, что в директории MIB 644 имеется файл APACHE2-MIB (хотя он входит в стандартный пакет AtomMind Network Manager и загружается по умолчанию)
- Добавьте устройство Хост сети 593 через обнаружение 181 или вручную
- Убедитесь, что APACHE-MIB проверяется в активах 501 устройств (обычно он проверяется по умолчанию)
- Дождитесь синхронизации 514 устройства (переключения в режим **Онлайн, Синхронизировано**, обозначенный иконкой )

AtomMind Network Manager предлагает несколько способов мониторинга веб-сервера Apache "из коробки", как описано в подразделах ниже:

- Инструментальная панель Apache 1891
- Тревоги Apache 1891

19.1.15.1.3.1 Инструментальная панель Apache

Инструментальная панель Apache открывается при двойном щелчке на узле устройства Веб-сервер Apache. Ее также можно выбрать из контекстного меню узла.

Это инструментальная панель предоставляет следующую информацию:

- Имя сервера
- Версия сервера
- Прослушиваемые порты
- Время непрерывной работы
- Общий трафик
- Общее количество обслуживаемых страниц
- Графики работающих и неработающих процессов
- Диаграмма количества обслуживаемых в секунду страниц
- График трафика
- График статистики ошибок HTTP
- И некоторые другие метрики

19.1.15.1.3.2 Тревоги сервера Apache

Для того, чтобы создать сообщения-тревоги для определенного web-сервера Apache, следует выбрать его хост в Системном дереве, а затем выбрать элемент меню **Установить профиль мониторинга**. Проверьте элемент **Создать тревоги** для группы мониторинга **Веб-сервера Apache** в окне Мониторинг настройки профиля. Можно создать одну и более тревог (сообщений) путем добавления записей тревог в таблицу настроек, определив их тип, имя, описание и параметры.

Доступные тревоги:

Имя тревоги	Условие активации	Примечания
Перегрузка запросов Apache (SNMP)	Метрика **Запросы в секунду** веб-сервера Apache превышает заданный порог.	Используйте действие Установить профайл мониторинга, группу **Apache** для создания. Доступна только для устройств, совместимых с SNMP. Параметры тревоги включают: Порог (количество запросов в секунду), Период проверки, Задержка и Уровень тревоги.
Перегрузка трафика Apache (SNMP)	Метрика **Количество переданных килобайт в секунду** веб-сервера Apache превышает заданный порог.	Используйте действие Установить профайл мониторинга, группу **Apache** для создания. Доступна только для устройств, совместимых с SNMP. Тревога активируется, когда количество переданных сервером Apache килобайт в секунду (`serverKBytesPerSec`) превышает заданный порог.
Рабочие процессы Apache (SNMP)	Метрика **Общее количество рабочих процессов** веб-сервера Apache превышает заданный порог.	Используйте действие Установить профайл мониторинга, группу **Apache** для создания. Доступна только для устройств, совместимых с SNMP. Тревога активируется, когда общее количество рабочих процессов Apache (`busyWorkers`) превышает заданный порог.
Ошибка HTTP Apache (SNMP)	Новая **ошибка HTTP определенного типа** генерируется веб-сервером Apache. Тревога функционирует путем мониторинга значений SNMP переменных, представляющих количество ошибок HTTP (см. таблицу ниже) и возникает при изменении их количества. AtomMind Network Manager может мониторить следующие ошибки HTTP (см. раздел Определения кодов состояния в Hypertext Transfer Protocol -- HTTP/1.1 для подробного ознакомления с кодами состояния HTTP):	Используйте действие Установить профайл мониторинга, группу **Apache** для создания. Доступна только для устройств, совместимых с SNMP. Параметры тревоги включают: Ошибка HTTP, Период проверки, Задержка и Уровень тревоги.

Код состояния	Краткое описание	SNMP-переменная
400	НЕВЕРНЫЙ ЗАПРОС	`httpError400`
403	ЗАПРЕЩЕНО	`httpError403`
404	НЕ ОБНАРУЖЕНО	`httpError404`
405	МЕТОД НЕ РАЗРЕШЕН	`httpError405`
500	ОШИБКА ВНУТРИ СЕРВЕРА	`httpError500`

501	НЕ РЕАЛИЗОВАНО	httpError501
505	ВЕРСИЯ НЕ ПОДДЕРЖИВАЕТСЯ	httpError505

19.1.15.1.4 Мониторинг Microsoft IIS

AtomMind Network Manager имеет встроенные средства для мониторинга Microsoft Internet Information Server. Он своевременно предупреждает о существующих или возможных проблемах, а также подсказывает наиболее эффективные пути решения. Решение позволяет производить мониторинг всех компонентов, входящих в состав Microsoft Internet Information Server, таких как Web, FTP, NNTP и SMTP серверов, пулов приложений и приложений в отдельности.

Конфигурирование мониторинга MS IIS Server

Для настройки мониторинга приложения:

- Заведите аккаунт сервера с приложением в узле **Устройства** в системном дереве
- Щелкните правой кнопкой на созданном аккаунте
- Выберите пункт **Setup Monitoring Profile** в контекстном меню открывшейся таблице
- Введите **Create Queries** напротив названий необходимых приложений и нажмите OK
- Удалите при необходимости ненужные запросы из таблицы или оставьте значения по умолчанию и нажмите OK
- Откройте инструментальную панель двойным щелчком по аккаунту устройства.

Ключевые метрики Microsoft Internet Information Server:

- Текущие подключения и пользователи, попытки входа в систему
- Трафик
- Статистика ошибок
- Эффективность кеша
- Блокировки
- Запросы
- Длина очереди

19.1.15.2 Мониторинг сервера приложений

Современные серверы приложений являются передовой платформой для хостинга множества приложений. Из-за их сложности, большинство серверов приложений основаны на технологиях Java и .NET для того, чтобы упростить процесс разработки веб-приложений.

Серверы приложений дают сотни и даже тысячи метрик работоспособности и производительности, что позволяет отслеживать состояние как сервера, так и всех его приложений.

Методы и задачи мониторинга сервера приложений

AtomMind Network Manager онтролирует работоспособность и производительность серверов приложений различными способами. Это позволяет решить следующие задачи:

- Отслеживание общих показателей эффективности сервера приложений, таких как количество запросов в секунду
- Мониторинг состояния и производительности базы данных

- Мониторинг загрузки ЦПУ, памяти, производительности ввода/вывода, использования пропускной способности каналов и других метрик аппаратного обеспечения
- Комплексный мониторинг HTTP/HTTPS коннекторов сервера приложений
- Мониторинг транзакций отдельных приложений
- Мониторинг веб-сервисов через протокол SOAP
- Подробный мониторинг Java/.NET виртуальных машин серверов приложений, утилизация памяти, статистика пула потоков
- Централизованный сбор логов серверов приложений через Syslog и журнал событий Windows
- Наблюдение за статусами серверов приложений, входящих в состав отказоустойчивого кластера или кластера с балансировкой нагрузки
- Анализ лог файлов и уведомление при обнаружении заданных ошибок
- Пороги предупреждений и тревог "из коробки"
- Статистика времени работы по часам / дням / неделям / месяцам / годам
- Инструментальные панели, графики и диаграммы для визуализации состояния серверов приложений
- Аналитика "из коробки"

AtomMind Network Manager предлагает многочисленные средства анализа и визуализации данных для известных серверов приложений:

- JBoss 1894
- Tomcat 1896
- WebSphere
- GlassFish
- И другие

Конфигурирование мониторинга Java Пприложений

Для настройки мониторинга:

- Настройте удаленный доступ через JMX вашего приложения в соответствии с его документацией
- Заведите аккаунт JMX устройства в узле **Устройства** в Системном дереве 370
- Щелкните правой кнопкой на созданном аккаунте
- Выберите пункт **Настроить мониторинг** в контекстном меню
- Отметьте **Создать запросы** в открывшейся таблице и нажмите ОК
- Удалите при необходимости не нужные запросы из таблицы или оставьте значения по умолчанию и нажмите ОК
- Откройте инструментальную панель двойным щелчком по аккаунту устройства

19.1.15.2.1 Мониторинг JBoss

AtomMind Network Manager содержит коробочное решение для безагентного мониторинга **JBoss** (**WildFly**).

Настройка параметров доступа для мониторинга JBoss

Для мониторинга **JBoss** или **WildFly** выполните последовательность действий:

- скопируйте файл из папки JBoss **\bin\client\jboss-client.jar** в папку **lib** AtomMind Network Manager при остановленном AtomMind Network Manager
- Отредактируйте файл **\standalone\configuration\standalone.xml** заменив

```
<interfaces>
    <interface name="management">
        <any-address/>
          <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
    </interface>
    <interface name="public">
          <inet-address value="${jboss.bind.address:127.0.0.1}"/>
    </interface>
    <interface name="unsecure">
        <inet-address value="${jboss.bind.address.unsecure:127.0.0.1}"/>
    </interface>
</interfaces>
```

на

```
<interfaces>
    <interface name="management">
        <any-address/>
    </interface>
    <interface name="public">
        <any-address/>
    </interface>
    <interface name="unsecure">
        <inet-address value="${jboss.bind.address.unsecure:127.0.0.1}"/>
    </interface>
</interfaces>
```

или подставьте вместо `127.0.0.1` адрес сервера мониторинга и перезапустите сервер приложений

- В случае использования domain режима, отредактируйте файл `domain\configuration\domain.xml`

URL для подключения устройства

```
service:jmx:http-remoting-jmx://[IP Address]:[Port]
```

Порт удаленного управления **JBoss** по умолчанию `9990`.

Метрики, доступные для мониторинга

AtomMind Network Manager отслеживает следующие метрики **JBoss**:

- Полная статистика подключений
- Статистика запросов
- Статистика транзакций
- Статистика ошибок
- Трафик
- Состояние памяти Java машины

- Потоки
- Загруженные классы
- И другие

19.1.15.2.2 Мониторинг Tomcat

AtomMind Network Manager предоставляет детальный мониторинг серверов Apache Tomcat "из коробки"

Настройка удаленного JMX доступа к Apache Tomcat 8

Добавьте следующие параметры в скрипт startup.bat для **Microsoft Windows** или startup.sh для **Linux** вашего сервера **Tomcat**.

Это синтаксис для **Microsoft Windows**. Команды вводятся в одну строку. Если **Tomcat** запущен как **Windows** сервис, используйте эти параметры для настройки опций Java для сервиса. Для **Linux** уберите "`set `" в начале строки.

```
set CATALINA_OPTS=-Dcom.sun.management.jmxremote -
Dcom.sun.management.jmxremote.port=9001 -Dcom.sun.management.jmxremote.ssl=false -
Dcom.sun.management.jmxremote.authenticate=false
```

Где `9001` - порт для JMX подключения.

Если вы нуждаетесь в авторизации, то внесите следующие изменения:

```
-Dcom.sun.management.jmxremote.authenticate=true
```

```
-Dcom.sun.management.jmxremote.password.file=../conf/jmxremote.password
```

```
-Dcom.sun.management.jmxremote.access.file=../conf/jmxremote.access
```

отредактируйте файл параметров доступа `$CATALINA_BASE/conf/jmxremote.access:`

```
monitorRole readonly
```

```
controlRole readwrite
```

отредактируйте файл с паролями `$CATALINA_BASE/conf/jmxremote.password:`

```
monitorRole tomcat
```

```
controlRole tomcat
```

Файл с паролями должен быть с атрибутом "только для чтения" для пользователя операционной системы от имени которого запущен Tomcat.

URL для подключения устройства

```
service:jmx:rmi:///jndi/rmi://[IP Address]:[Port]/jmxrmi
```

Доступные метрики

AtomMind Network Manager позволяет отслеживать следующие метрики **Tomcat**:

- Статистика по трафику
- Детализация кеша
- Статистика подключений
- Статистика ошибок
- Статистика запросов
- и другие

19.1.15.2.3 Мониторинг GlassFish

AtomMind Network Manager способен поднять уровень мониторнга и устранения неисправностей **Oracle Glassfish** на гораздо более высокий уровень. Он способен круглосуточно и непрерывно выполнять мониторинг всех ваших серверов и выводить информацию на единой консоли. Для решения этой задачи используется комплексный подход. AtomMind Network Manager способен отслеживать как текущие данные, так и накапливать статистические, благодаря чему вы сможете заранее спланировать расширение ресурсов. Это дает возможность экономить на

модернизации оборудования и стоимости лицензий для программного обеспечения, так как вы будете точно знать, какие именно ресурсы нуждаются в замене или модернизации.

Настройка удаленного JMX доступа к серверу Oracle GlassFish

Для мониторинга **GlassFish** выполните ряд действий:

- Откройте интерфейс администратора вашего сервера **GlassFish**
- Расширьте меню `Configuration` и найдите элемент `server-config`
- Расширьте его и выберите элемент `Admin Service`
- Введите IP-адрес AtomMind Network Manager в поле **Адрес**
- Сохраните изменения

URL для создания аккаунта устройства

```
service:jmx:rmi:///jndi/rmi://[host]:[port]/jmxrmi
```

Порт удаленного мониторинга по умолчанию **GlassFish** – `8686`.

Доступные метрики

Вот лишь некоторые доступные метрики производительности Oracle GlassFish Server:

- Общая информация о сервере и приложениях
- Утилизация памяти
- Активные сессии
- Активные подключения
- Потоки
- Трафик сервера
- Подробная статистика HTTP ошибок и ошибок сервлетов
- Использование кэша
- Статистика запросов
- Статистика транзакций
- И другие

19.1.15.2.4 Мониторинг WebLogic

AtomMind Network Manager способен поднять уровень мониторинга и устранения неисправностей **Oracle WebLogic Server** на гораздо более высокий уровень. Он способен круглосуточно и непрерывно выполнять мониторинг всех ваших серверов и выводить информацию на единой консоли. AtomMind Network Manager способен отслеживать как текущие данные, так и накапливать статистические, благодаря чему вы сможете заранее спланировать расширение ресурсов. Это дает возможность экономить на модернизации оборудования и стоимости лицензий для программного обеспечения, так как вы будете точно знать, какие именно ресурсы нуждаются в замене или модернизации.

Настройка удаленного доступа JMX к Oracle WebLogic Server

Если вы используете аутентификацию JMX, выполните следующие действия:

- Найдите вкладку %JAVA_HOME%\jre\lib\management.
- Переименуйте jmxremote.password.templatefile на jmxremote.password. Отредактируйте jmxremote.passwordby перемещение/добавление его контента с помощью следующего:

пароль пользователя

Где пользователь является пользователем JVM, а пароль – пользовательским паролем.

Затем, измените данный файл:

```
%MIDDLEWARE_HOME%\user_projects\domains\<your_domain>\bin\startWebLogic.cmd
```

где %MIDDLEWARE_HOME% – это путь к вашей инсталляции WebLogic, и your_domain – это ваш домен WebLogic, добавьте строчки после call "%DOMAIN_HOME%\bin\setDomainEnv.cmd" %*.

Для Windows:

```
set "JAVA_OPTIONS=%JAVA_OPTIONS% -
Djavax.management.builder.initial=weblogic.management.jmx.mbeanserver.WLSMBeanServerBu
ilder"

set "JAVA_OPTIONS=%JAVA_OPTIONS% -Dcom.sun.management.jmxremote"

set "JAVA_OPTIONS=%JAVA_OPTIONS% -Dcom.sun.management.jmxremote.port=8686"

set "JAVA_OPTIONS=%JAVA_OPTIONS% -Dcom.sun.management.jmxremote.ssl=false"

set "JAVA_OPTIONS=%JAVA_OPTIONS% -Dcom.sun.management.jmxremote.authenticate=true"

set "JAVA_OPTIONS=%JAVA_OPTIONS% -Dcom.sun.management.jmxremote.password.file=" c:
\jmxremote\jmxremote.password""

set "JAVA_OPTIONS=%JAVA_OPTIONS% -Dcom.sun.management.jmxremote.access.file=" c:
\jmxremote\jmxremote.access""
```

Для Linux:

```
JAVA_OPTIONS="$JAVA_OPTIONS-
Djavax.mSanagement.builder.initial=weblogic.management.jmx.mbeanserver.WLSMBeanServerB
uilder"

JAVA_OPTIONS="$JAVA_OPTIONS-Dcom.sun.management.jmxremote"

JAVA_OPTIONS="$JAVA_OPTIONS-Dcom.sun.management.jmxremote.port=8686"

JAVA_OPTIONS="$JAVA_OPTIONS-Dcom.sun.management.jmxremote.ssl=false"

JAVA_OPTIONS="$JAVA_OPTIONS-Dcom.sun.management.jmxremote.authenticate=true"

JAVA_OPTIONS="$JAVA_OPTIONS-
Dcom.sun.management.jmxremote.password.file="/jmxremote/jmxremote.password""

JAVA_OPTIONS="$JAVA_OPTIONS-
Dcom.sun.management.jmxremote.access.file="/jmxremote/jmxremote.access""
```

Сохраните изменения и перезагрузите WebLogic Server.

Добавьте устройство JMX, не используя Пользовательский URL Сервиса

Доступные метрики

Менеджер обеспечивает мониторинг следующих метрик Oracle WebLogic Server:

- Общая информация о сервере и операционной системе
- Использование памяти
- Активные сессии
- Загрузка классов
- Threads
- Каналы
- Детальная статистика транзакций, JMS, SAF, WLS, безопасности, подсистем коннектора
- И другие

19.1.15.3 Мониторинг промежуточного ПО

AtomMind Network Manager упрощает и автоматизирует мониторинг и управление вашего промежуточного программного обеспечения:

- Идентификация промежуточного программного обеспечения и проблемных транзакций, которые влияют на качество ИТ услуг
- Устранением проблем, влияющих на производительность приложений и доставку бизнес-сервисов

- Мониторинг промежуточного программного обеспечения масштаба предприятия в единой консоли
- Планирование мощностей, система раннего предупреждения о тенденциях, которые могут повлиять на бизнес-приложения
- Просмотр производительности промежуточного программного обеспечения в реальном времени для оценки его состояния
- Графики и отчеты для обнаружения пиковых значений и построения долгосрочных трендов

Мониторинг сервера промежуточного программного обеспечения

AtomMind Network Manager предоставляет аналитику для распространенного промежуточного программного обеспечения "из коробки":

- Microsoft SharePoint 1899
- IBM WebSphere MQ 1900

Встроенная аналитика включает:

- Детальный мониторинг очереди сообщений (глубина, сообщение возрастов статистики и т.д.)
- Мониторинг активности каналов (трафик, количество сообщений, статистика буфера и т.д.)
- Мониторинг слушателей (текущий статус, статистика)
- И другое (процессы, куча, кэш, SQL запросы, потоки, сессии, и т.д.)

19.1.15.3.1 Мониторинг Microsoft SharePoint

AtomMind Network Manager озволяет быстро и точно оценить как общее состояние Microsoft SharePoint, так и любого ее компонента в отдельности. Это позволит вам сократить время поиска неполадок.

Конфигурация мониторинга MS Share PointServer

Для конфигурации мониторинга SharePoint:

- Создайте аккаунт WMI в узле **Устройства** Системного дерева 370
- Кликните правой кнопкой мыши по аккаунту
- Выберите **Профиль Мониторинга Настроек** в контекстном меню
- Введите **Создать запросы** в таблице и нажмите OK
- При необходимости удалите запросы и просто нажмите
- Откройте Панель инструментов двойным щелчком по аккаунту устройства

Доступные метрики

Менеджер может отслеживать состояние всех счетчиков производительности Microsoft SharePoint Server, а именно:

- Размер, эффективность, использование кэша
- Количество и размер объектов в кэше
- Количество и использование выделенной памяти, работу сборщика мусора и многие другие метрики MS Foundation
- Очередь выполняемых SQL запросов
- Очередь запросов к веб-страницам
- Детальная информация о службах Excel Calculation
- Производительность системы поиска, сессии, запрошенные документы
- Метрики производительности веб-приложений

19.1.15.3.2 Мониторинг WebSphere MQ

Мониторинг IBM WebSphere MQ производится через драйвер устройства WebSphere MQ 660. Он обеспечивает следующую функциональность:

- Мониторинг доступности и работоспособности сервера MQ
- Мониторинг каналов (состояние, время запуска, входящий/исходящий трафик)
- Мониторинг очередей (глубина, последнее время операций получения/ввода, максимальный и средний возраст сообщений, количество входов/выходов, количество незафиксированных сообщений)
- Мониторинг слушателей (время запуска, размер блока данных, счетчики команд/сессий)

19.1.15.4 Мониторинг Active Directory

AtomMind Network Manager предлагает средства комплексного мониторинга Active Directory "из коробки":

Конфигурация мониторинга Active Directory

Для конфигурирования мониторинга Сервера AD:

- Создайте аккаунт WMI в узле **Устройства** Системного дерева 370
- Кликните правой кнопкой мыши по аккаунту
- Выберите **Профиль Мониторинга Настроек** в контекстном меню
- Введите **Создать запросы** в таблице и нажмите OK
- При необходимости удалите запросы и просто нажмите
- Откройте Панель инструментов двойным щелчком по аккаунту устройства

Доступные метрики

- Отслеживание состояния всех сервисов и компонентов Active Directory
- Выявление проблем репликации данных с возможностью ее настройки в соответствии с возможностями сети и загрузкой
- Помощь системным администраторам в создании Active Directory
- Помощь в оценке необходимых ресурсов при масштабировании системы
- Возможность заранее выявить различные виды потенциальных проблем, предотвратить возможные пагубные изменения и минимизировать отказы в работе
- Сбор статистики за длительные периоды времени, позволяющие анализировать производительнось Active Directory
- Предоставление необходимых данных для выявления узких мест и планирования производительности
- Повышение общего уровня безопасности в компании

19.1.15.4.1 Мониторинг Active Directory (LDAP)

LDAP-мониторинг осуществляется драйвером устройства сетевого хоста 593. Он предоставляет возможность устанавливать связь с LDAP-сервером и выполнять там запросы поиска. См. раздел LDAP-мониторинг 611.

Тревоги мониторинга LDAP

Имя тревоги	Условие активации	Примечания
Результат запроса LDAP	Результат запроса **LDAP** содержит заданную строку.	Для создания используйте группу **LDAP** в действии Установить профиль мониторинга 1808.

19.1.15.5 Мониторинг сервера FTP

Мониторинг сервера FTP поддерживается драйвером устройства Хоста сети[593]. Он выполняет проверку работоспособности, получая информацию о состоянии сервера и атрибутов удаленного файла. Более того, FTP-мониторинг можно использовать для отслеживания удаленных файлов или директорий, для анализа деятельности удаленных приложений путем обработки отчетов об ошибках, журналов и пр. Дополнительную информацию о процедуре мониторинга, параметрах и возможных результатах можно найти в разделе Мониторинг FTP-сервера[605].

Тревоги мониторинга FTP

Имя тревоги	Условие активации	Примечания
Размер файла FTP	Размер файла на сервере **FTP** удовлетворяет *условию,* состоящему из порога (в байтах) и *операции сравнения*.	Для создания используйте группу **FTP** в действии Установить профиль мониторинга[1808].
Общий размер файлов FTP	Общий размер всех отслеживаемых файлов через **FTP** удовлетворяет *условию,* состоящему из порога (в байтах) и *операции сравнения*.	Для создания используйте группу **FTP** в действии Установить профиль мониторинга[1808].
Изменение файла FTP	Контент определенного файла, полученного от сервера **FTP**, изменился.	Для создания используйте группу **FTP** в действии Установить профиль мониторинга[1808].
Файл FTP слишком старый	Файл на сервере **FTP** не был изменен в заданный период времени.	Для создания используйте группу **FTP** в действии Установить профиль мониторинга[1808].

Графики мониторинга FTP

Имя графика	Описание	Примечания
Размер файла FTP	Размер диаграмм файла, отслеживаемого через FTP.	График можно создать для определенного устройства, работающего по FTP, используя действие Установить профиль мониторинга[1808].

19.1.15.6 Мониторинг E-mail сервера

Поддержка AtomMind Network Manager мониторинга e-mail сервера гарантирует то, что сервисы электронной почты работают должным образом. Методы мониторинга включают:

- Мониторинг общей работоспособности серверов POP3[1901], IMAP[1902] и SMTP[1902].
- Мониторинг доставки[1902].

19.1.15.6.1 POP3-мониторинг

Доступность и соответствующее функционирование почтового сервера **POP3** отслеживается путем подключения к почтовому ящику и получения информации о количестве сообщений и общем размере всех сообщений из папки `INBOX`. Это реализует драйвер устройства Хоста сети[593]. Описание функционирования сервиса, настроек и результатов см. в разделе Мониторинг сервера POP3[601].

Тревоги сервиса POP3

Имя тревоги	Условие активации	Примечания
Слишком много E-mail сообщений (POP3)	Количество сообщений (полученных через **POP3**) превышает заданный порог.	Для создания используйте группу **E-mail** в действии Установить профиль мониторинга[1808].

Большой размер сообщений (POP3)	Общий размер сообщений (полученных через **POP3**) превышает заданный порог.	Для создания используйте группу **E-mail** в действии Установить профиль мониторинга 1808.

19.1.15.6.2 IMAP-мониторинг

Доступность и функционирование почтового сервера **IMAP** реализуется драйвером устройства Хоста сети 593. Мониторинг осуществляется путем подключения к почтовому ящику и получения информации о количестве сообщений и общем размере всех сообщений из указанной папки. См раздел IMAP-мониторинг 602 и соответствующие подразделы.

Тревоги сервиса IMAP

Имя тревоги	Условие активации	Примечания
Слишком много E-mail сообщений (IMAP)	Количество сообщений (полученных через **IMAP**) превышает заданный порог.	Для создания используйте группу **E-mail** в действии Установить профиль мониторинга 1808.
Большой размер сообщений (POP3)	Общий размер сообщений (полученных через **IMAP**) превышает заданный порог.	Для создания используйте группу **E-mail** в действии Установить профиль мониторинга 1808.

19.1.15.6.3 SMTP-мониторинг

Доступность и функционирование **почтового сервера SMTP** отслеживается путем установления соединения с сервером и прохождения процедуры авторизации. Почтовые сообщения не отправляются. См. раздел Мониторинг сервера SMTP 603.

19.1.15.6.4 Мониторинг отправки письма с возвратом

Мониторинг отправки письма с возвратом гарантирует доставку электронных сообщений, он проверяет функционирование почтовой системы в целом, генерирует специально созданные тестовые сообщения, отправляет их, а затем проверяет, доставлены ли они получателю. Монитор использует SMTP для отправки сообщений, а также POP3 или IMAP для возврата и удаления их из отслеживаемой папки. Сервис реализуется драйвером устройства сетевого хоста 593. См. раздел Мониторинг отправки письма с возвратом 604 и его подразделы.

19.1.15.6.5 Мониторинг Postfix

Доступность и полноценное функционирование **Postfix** осуществляется драйвером устройства Хост Сети 593. Мониторинг выполняется SNMP.

19.1.15.6.6 Мониторинг Exchange

AtomMind Network Manager включает в себя мониторинг Microsoft Exchange Server, обеспечивающий своевременное прогнозирование существующих или возможных проблем, а также предлагает наилучший вариант решения проблем. Он предоставляет мониторинг для всех компонентов Exchange.

Exchange Server Monitoring Configuration

Для конфигурирования мониторинга MS IIS:

- Создайте аккаунт WMI в узле **Устройства** Системного дерева 370
- Кликните правой кнопкой мыши по аккаунту
- Выберите **Профиль Мониторинга Настроек** в контекстном меню
- Введите **Создать запросы** в таблице и нажмите OK
- При необходимости удалите запросы и просто нажмите OK

- Откройте Панель инструментов двойным щелчком по аккаунту устройства

Доступные метрики

- AB сессии клиента
- Дорожки ATQ
- Поиски
- Привязки
- Статистика DRA
- Статистика LDAP
- Задержка SAM
- Статистика DFS
- И другие

19.1.15.7 Мониторинг SSH-сервера

Мониторинг SSH-сервера осуществляется драйвером устройства сетевого хоста 593. Он предоставляет возможность подключаться к сетевому хосту по протоколу SSH, авторизоваться и выполнять скрипты на удаленном хосте. Помимо мониторинга доступности/работоспособности он позволяет выполнять различные административные задачи. См. раздел Мониторинг SSH-сервера 609.

Тревоги мониторинга SSH

Имя тревоги	Условие активации	Примечания
Результат скрипта SSH	Вывод скрипта **SSH** содержит заданную строку.	Для создания используйте группу **SSH** в действии Установить профиль мониторинга 1808.

19.1.15.8 Мониторинг Radius-сервера

Мониторинг RADIUS-сервера выполняется драйвером устройства сетевого хоста 593. Он позволяет отправлять запросы и обрабатывать ответы. См. раздел Мониторинг RADIUS-сервера 612.

19.1.15.9 Мониторинг DNS-сервера

Мониторинг DNS-сервера осуществляет драйвер устройства сетевого хоста 593. Он позволяет отправлять DNS-запросы серверу и предоставляет их результаты для анализа. Дополнительную информацию о его операции, настройке и результатах см. в разделе Мониторинг DNS-сервера 600.

Тревоги мониторинга DNS

Имя тревоги	Условие активации	Примечания
Результат запроса DNS	Результат данного запроса **DNS** равен (или отличается от) строки.	Для создания используйте группу **DNS** в действии Установить профиль мониторинга 1808.

19.1.15.10 Мониторинг DHCP-сервера

Мониторинг DHCP-сервера (протокол динамической настройки хостов) выполняет драйвер устройства сетевого хоста 593. Мониторинг осуществляется путем отправки *DISCOVER*-запроса и прослушивания соответствующего ответа от DHCP-сервера. См. раздел Мониторинг DHCP-сервера 611.

19.1.15.11 Общий мониторинг TCP-сервиса

Драйвер устройства сетевого хоста 593 осуществляет *мониторинг доступности/работоспособности* TCP-сервисов любого вида (поэтому в этом случае мы говорим об *общем* мониторинге, подключаясь к устройству по назначенному порту, отправляя сырые данные и получая ответы. Некоторые службы можно настроить так, чтобы они отслеживались на определенном IP-хосте. См. раздел Общий мониторинг TCP-сервиса 598.

Тревоги мониторинга порта TCP

Имя тревоги	Условие активации	Примечания
Результат мониторинга TCP/UDP	Результат заданного запроса **TCP или UDP** содержит строку.	Для создания используйте группу **Общие сервисы TCP и UDP** в действии Установить профиль мониторинга 1808.

19.1.15.12 Общий мониторинг UDP-сервиса

Поддержку мониторинга UDP-сервиса осуществляет драйвер устройства сетевого хоста 593. Некоторые UDP-сервисы можно отследить на разных портах IP-хоста. Процедура мониторинга включает отправку данных пользователя и получение ответа. См. раздел Общий мониторинг UDP-сервиса 599.

Тревоги мониторинга порта TCP

Имя тревоги	Условие активации	Примечания
Результат мониторинга TCP/UDP	Результат заданного запроса **TCP или UDP** содержит строку.	Для создания используйте группу **Общие сервисы TCP и UDP** в действии Установить профиль мониторинга 1808.

19.1.16 Мониторинг виртуальной инфраструктуры

AtomMind Network Manager обеспечивает консолидацию, мониторинг, автоматизацию, оптимизацию и планирование вашей виртуальной инфраструктуры.

Для более подробного рассмотрения мы подразделяем мониторинг виртуальной инфраструктуры на несколько уровней:

- Централизованная платформа для управления: VMware vCenter Server (уровень централизованной платформы)
- Гипервизоры: VMware ESX/ESXi; Microsoft Hyper-V (уровень сервера)
- Виртуальные машины (уровень виртуальной машины)
- Гостевые ОС (уровень операционной системы)

В контексте этой классификации AtomMind Network Manager предоставляет мониторинг на всех уровнях при помощи стандартных и специфических для производителя инструментов. Например, на уровне сервера, гипервизор ESX/ESXi может рассматриваться как стандартная машина Linux, и поэтому ее мониторинг может осуществляться с помощью стандартных инструментов (см. Мониторинг доступности 1853, Мониторинг производительности 1856, Мониторинг сервисов/приложений 1888). Это также работает для гипервизора Hyper-V на Windows. Похожим образом, на уровне *операционной системы,* работающая внутри Guest VM операционная система ведет себя как любая другая ОС, поэтому можно осуществлять ее мониторинг при помощи того же подхода.

Набор инструментов Мониторинга Виртуальной Инфраструктуры AtomMind Network Manager позволяет осуществлять мониторинг определенных *виртуальных машин, серверов* гипервизора или *централизованных платформ*, на которых они работают. Он включает панели инструментов с набором виджетов, моделей и фильтров событий. При необходимости, набор иструментов может быть индивидуализирован или расширен с помощью создания новых инструментов на основе шаблонов и/или примеров существующих инструментов.

AtomMind Network Manager осуществляет мониторинг виртуальных машин, работающих на гипервизорах VMware с помощью специальных плагинов VMware и протокола SNMP, используя `vmwVmTable` и другие переменные на основе MIB файлов VMware. Эти переменные конфигурируются автоматически для периодических опросов (для предоставления интерактивного обновления значений). Настройки опросов можно изменить для специальных задач, используя Параметры синхронизации настроек 502. Обратитесь к документации VMware для получения набора MIB файлов и более подробной информации об активации SNMP на гипервизорах VMware.

Мониторинг виртуальных машин, работающих на гипервизоре Microsoft Hyper-V осуществляется с помощью *WMI*. Смотрите Мониторинг и управление WMI 1850 для правильной настройки WMI.

Конфигурация мониторинга виртуальной инфраструктуры для VMware ESX/ESXi и VMware vCenter Server

Для конфигурации и мониторинга VMware с помощью **плагина VMware**:

- Создайте аккаунт **VMware** в узле **Устройства** Системного дерева 370
- Установите свойство **URI vSphere Service (чувствительно к регистру)** как https://your_host_ip/sdk
- Введите **Имя пользователя** и **Пароль**. В большинстве случаев **Имя пользователя –** *root*
- Если вы настраиваете vCenter Server, установленный как приложение Windows, сделайте активным свойство **Is SSO активен (только vCenter)**. В других случаях оставьте это свойство неактивным (по умолчанию)
- Сделайте неактивным свойство **Пропустить проверку SSL сертификатов**, если вы редактируете SSL механизм в вашем соединении. В большинстве случаев, если вам неизвестно о состоянии SSL, оставьте свойство неактивным (по умолчанию)
- Откройте инструментальную панель Обзор сервера виртуализации 1906 двойным щелчком по аккаунту устройства

Для конфигурации мониторинга VMware с помощью **SNMP**:

- Активируйте и настройте **SNMP** сервис на вашем хосте. Пожалуйста, обратитесь к документации VMware для получение дополнительной информации
- Добавьте MIB файлы Tibbo в папку **mib** folder, расположенную в директории инсталляции AtomMind Network Manager
- Создайте **Хост сети** или аккаунт **SNMP** в узле **Устройства** Системного дерева 370
- Проверьте и настройте параметры **SNMP сервиса,** если это необходимо. Они находятся в аккаунте **Сервисы** вкладки **SNMP** для типа контекста **Хост сети** или в аккаунте вкладки **Свойства подключения** для типа контекста **SNMP**
- Активируйте добавленные MIB файлы в аккаунт **Метаданные** вкладки **Активы** для типа контекста **Хост сети** или в аккаунте **Метаданные** вкладки **Активы** нажатием на **SNMP** ячейки **Вложенные активы** для типа контекста **SNMP**
- Откройте панель инструментов Обзор сервера виртуализации 1906 двойным щелчком по аккаунту устройства

Конфигурация мониторинга виртуальногй инфраструктуры для Microsoft Hyper-V

Для конфигурации мониторинга Hyper-V с помощью **WMI**:

- Активируйте и настройте **WMI** сервис на вашем хосте. Смотрите Мониторинг и управление WMI 1850 для правильной настройки WMI
- Создайте **Хост сети** или аккаунт **WMI** в узле **Устройства** Системного дерева 370
- Настройте параметры **WMI сервиса**, если это необходимо. Они находятся в аккаунте **Сервисы** вкладки **WMI** для типа контекса **Хост сети** или во вкладке **Свойства подключения** для типа контекса **WMI**. Убедитесь, что добавлено пространство имен **ROOT\VIRTUALIZATION** или активировано свойство **Все пространства имен**
- Активируйте основные классы виртуализации WMI: `ROOT\VIRTUALIZATION:Msvm_ComputerSystem`, `ROOT\VIRTUALIZATION:Msvm_KvpExchangeComponent`, `ROOT\VIRTUALIZATION:Msvm_VirtualSystemManagementService` в аккаунте **Метаданные** вкладки **Активы** для типа контекста **Хост сети** или в аккаунте **Метаданные** вкладки **Активы**, кликнув по **WMI** ячейки **Вложенные активы** для типа контекста **WMI**
- Откройте панель инструментов Обзор сервера виртуализации 1906 двойным щелчком по аккаунту устройства

19.1.16.1 Инструментальные панели виртуализации

AtomMind Network Manager предоставляет следующие готовые панели инструментов *Мониторинга виртуальной инфраструктуры*:

Имя панели инструментов	Описание панели инструментов

Обзор сервера виртуализации 1906	Панель инструментов **Обзор сервера виртуализации** предоставляет информацию о *виртуальных машинах*, которая располагается в специальном *гипервизоре*, и ее основные метрики являются дополнительной информацией.
Обзор Виртуализации 1907	Панель инструментов **Обзор систем виртуализации** отображает суммарную информацию о вашей виртуальной машине.
Топ 10 систем виртуализации 1908	Панель инструментов **Топ 10 систем виртуализации** отображает основные показатели (использование ЦП, использование памяти) всех *виртуальных машин* и *гипервизоров*, сортируемых по загрузке.

Панели инструментов создаются автоматически в группе **Виртуализация**.

Полнота информации (показатели из списка *виртуальных машин* или *гипервизоров*) зависит от доступности данных и переменных на данном хосте.

Например: если *гипервизор* VMware ESXi подключен с помощью плагина VMware, он предоставляет больше информации о *виртуальных машинах*, чем когда он подключен с помощью SNMP.

19.1.16.1.1 Обзор сервера виртуализации

Панель инструментов **Обзор сервера виртуализации** предоставляет информацию о виртуальных машинах, которая хранится в специальном *гипервизоре,* и его основные метрики являются дополнительной информацией.

Данная панель является относительной и применяется ко всем *гипервизорам* или *серверам централизованной платформы*. Она открывается двойным нажатием на определенный контекст в системном дереве.

Внешний вид инструментальной панели зависит от роли хоста.

Панель инструментов Обзор сервера виртуализации для VMware ESX/ESXi и Microsoft Hyper-V

Данная панель инструментов содержит несколько элементов:

- Виджет **Hypervisor Main Counters** показывает использование ЦП *гипервизора* и использование памяти (отображаются как радиальный индикатор), использование дискового пространства *гипервизора* и сетевую загрузку, если такие данные доступны
- Таблица данных **Information** показывает всю дополнительную информацию о *гипервизоре*: описание, URL/IP адрес, имя хоста, имя вендора, время последней синхронизации, статус подключения, статус синхронизации, детали синхронизации
- Журнал событий **Host Events** отображает все события относящиеся к *гипервизору*
- Виджет **VMs Counter** отображает количество *виртуальных машин* по статусу: все, работающие; и их контрольные сигналы: неизвестный, ok, предупреждение, критический
- Таблица данных **VMs List** показывает список *виртуальных машин*, находящийся на данном *гипервизоре*. Этот список содержит полную информацию о *виртуальных машинах*: имя, состояние, контрольный сигнал, имя гостевой ОС, использование ЦП, использование памяти, использование диска, сетевая загрузка, имя хоста, IP-адрес, время непрерывной работы
- Таблица данных **Active Alerts** показывает активные тревоги *гипервизора*
- Виджет **Cpu Usage Hypervisor VMs** представляет собой диаграмму с областями, которая показывает использование ЦП *гипервизора виртуальными машинами*
- Виджет **Memory Usage Hypervisor VMs** представляет собой диаграмму с областями, которая показывает использование памяти *гипервизора виртуальными машинами*
- Виджет **Disk Usage Hypervisor VMs** представляет собой диаграмму с областями, которая показывает использование дискового пространства *гипервизора виртуальными машинами*
- Виджет **Network Usage Hypervisor VMs** представляет собой диаграмму с областями, которая показывает сетевую загрузку *гипервизора виртуальных машин*

Панель инструментов Обзор сервера виртуализации для VMware vCenter Server

Данная панель инструментов содержит несколько элементов:

- Таблица данных **Hypervisors List** показывает список всех *гипервизоров*, контролируемых с помощью *серверов централизованной платформы* и их показателей: имя датацентра, имя кластера, ссылка на определенную

панель инструментов *гипервизора* Обзор сервера виртуализации1906, состояние, IP-адрес, имя продукта, версия продукта, ядра всех процессоров, частота всех процессоров, размер памяти, использование ЦП, использование памяти, использование диска, сетевая загрузка, дата последней загрузки

- Таблица данных **Information** показывает информацию о сервере централизованной платформы: описание, URL, имя хоста, имя вендора, время последне синхронизации, статус подключения, статус синхронизации, детали синхронизации
- Журнал событий **Host Events** отображает все соответствующие события сервера *централизованной платформы*
- Виджет **VMs Counter** представляет собой количество *виртуальных машин* по статусу: все, работающие; и их контрольные сигналы: неизвестный, ok, предупреждение, критический
- Таблица данных **VMs List** показывает весь список *виртуальных машин*, расположенный на *гипервизорах*, которые контролируются при помощи *сервера централизованной платформы*. Данный список содержит полную информацию о *виртуальных машинах*: имя, состояние, пульс, имя гостевой ОС, использование ЦП, использование памяти, использование диска, сетевая загрузка, имя хоста, IP-адрес, время непрерывной работы
- Таблица данных **Active Alerts** показывает активные тревоги *централизованной платформы*
- Виджет **Hypervisors Cpu Usage** представляет собой диаграмму, которая показывает использование ЦП для всех *гипервизоров*, контролируемых с помощью *сервера централизованной платформы*
- Виджет **Hypervisors Memory Usage** представляет собой диаграмму, которая показывает использование памяти для всех *гипервизоров*, контролируемых с помощью *сервера централизованной платформы*
- Виджет **Hypervisors Disk Usage** представляет собой диаграмму, которая показывает использование дискового пространства для всех *гипервизоров*, контролируемых с помощью *сервера централизованной платформы*
- Виджет **Hypervisors Network Usage** представляет собой диаграмму, которая показывает загрузку сети для всех *гипервизоров*, контролируемых с помощью *сервера централизованной платформы*
- Виджет **Cpu Usage Hypervisors VMs** представляет собой диаграмму, которая показывает использование ЦП *гипервизоров виртуальными машинами,* контролируемыми с помощью *сервера централизованной платформы*
- Виджет **Memory Usage Hypervisors VMs** представляет собой диаграмму, которая показывает использование памяти *гипервизоров виртуальными машинами,* контролируемыми с помощью *сервера централизованной платформы*
- Виджет **Disk Usage Hypervisors VMs widget** представляет собой диаграмму, которая показывает использование дискового пространства *гипервизоров виртуальными машинами,* контролируемыми с помощью *сервера централизованной платформы*
- Виджет **Network Usage Hypervisors VMs** представляет собой диаграмму, которая показывает загрузку сети *гипервизоров виртуальными машинами,* контролируемыми с помощью *сервера централизованной платформы*

19.1.16.1.2 Обзор виртуализации

Панель инструментов **Обзор виртуализации** предоставляет суммарную информацию о вашей виртуальной инфраструктуре.

Эта инструментальная панель абсолютна и может быть открыта двойным щелчком по элементу панели в системном дереве.

Данная панель содержит несколько элементов:

- Журнал событий **События виртуализации** отображает все события виртуальной инфраструктуры в пределах фильтра События виртуализации1910
- Таблица данных **Hypervisors List** показывает весь список *гипервизоров, серверов централизованной платформы*, и их показатели: имя датацентра, имя кластера, ссылка на определенную инструментальную панель *гипервизора* Обзор сервера визуализации1906, состояние, IP-адрес, имя продукта, версия продукта, ядра всех процессоров, частота всех процессоров, объем памяти, использование ЦП, использование памяти, использование дискового пространства, загрузка сети, дата последней загрузки
- Виджет **VMs Counter** отображает количество *виртуальных машин* по статусу: все, работающие; и их контрольные сигналы: неизвестный, ok, предупреждение, критический
- Таблица данных **VMs List** показывает весь список *виртуальных машин,* расположенный на гипервизорах или контролируемый с помощью *сервера централизованной платформы*. Этот список содержит полную информацию о *виртуальных машинах*: имя, состояние, контрольный сигнал, имя гостевой системы, использование ЦП, использование памяти, использование дискового пространства, загрузка сети, имя хоста, IP-адрес, время непрерывной работы

19.1.16.1.3 Топ 10 систем виртуализации

Инструментальная панель **Virtualization TOP 10** показывает главные индикаторы (использование ЦП, использование памяти) всех *виртуальных машин* и *гипервизоров*, сортируемых по нагрузке.

Эта инструментальная панель абсолютна и может быть открыта двойным щелчком по элементу панели в системном дереве.

Данная панель содержит несколько элементов:

- Таблица данных **Hypervisors Cpu Utilization TOP 10** отображает топ 10 *гипервизоров* и *серверов централизированной платформы*, сортированных по использованию процессора. Этот список содержит некую информацию о *гипервизорах*: ссылка на определенную инструментальную панель *гипервизора* Обзор сервера визуализации 1908, имя датацентра, имя кластера, имя продукта, версия продукта, использование ЦП, использование памяти, IP-адрес, дата последней загрузки
- Таблица данных **Hypervisors Memory Utilization TOP 10** отображает топ 10 *гипервизоров* и *серверов централизированной платформы, сортированных по использованию памяти*. Этот список содержит некую информацию о *гипервизорах*: ссылка на определенную инструментальную панель *гипервизора* Обзор сервера визуализации 1908, имя датацентра, имя кластера, имя продукта, версия продукта, использование ЦП, использование памяти, IP-адрес, дата последней загрузки
- Таблица данных **VMs Cpu Utilization TOP 10** отображает список топ 10 *виртуальных машин*, расположенных на *гипервизорах* или контролируемых с помощью *сервера централизованной платформы*, сортированных по использованию ЦП. Этот список содержит некую информацию о *виртуальных машинах*: имя, имя гостевой системы, использование ЦП, использование памяти, IP-адрес, время непрерывной работы
- Таблица данных **VMs Memory Utilization TOP 10** список топ 10 *виртуальных машин*, расположенных на *гипервизорах* или контролируемых с помощью *сервера централизованной платформы*, сортированных по использованию памяти. Этот список содержит некую информацию о *виртуальных машинах*: имя, имя гостевой системы, использование ЦП, использование памяти, IP-адрес, время непрерывной работы

19.1.16.2 Модели виртуализации

AtomMind Network Manager предоставляет следующие готовые модели *Мониторинга виртуальной инфраструктуры*:

Имя модели	Описание модели
Мониторинг визуализации 1908	Модель **Мониторинг виртуализации** используется для сбора даных хоста и построения списка Виртуальных Машин для определенного *гипервизора* или *сервера централизованной платформы*. Эти данные используются в панели инструментов Обзор сервера виртуализации 1906.
Обзор визуализации 1909	Модель **Обзор виртуализации** используется для построения таблиц из списка всех *виртуальных машин* и *гипервизоров*. Данные таблицы используются в инструментальных панелях Обзор визуализации 1907 и Топ 10 систем виртуализации 1908.

Модели создаются автоматически в группе **Визуализация**.

19.1.16.2.1 Модель "Мониторинг виртуализации"

Модель **Мониторинг визуализации** используется для сбора данных хоста и построения списка Виртуальных машин для определенного *гипервизора* или *сервера централизованной платформы*. Эти данные используются в панели инструментов Обзор сервера виртуализации 1906.

Эта модель является относительной, и применяется ко всем *гипервизорам* или *серверам централизованной платформы*. Созданные с помощью модели переменные могут быть открыты кликом правой кнопкой мыши по элементу **Редактировать дополнительные свойства** в особом контексте системного дерева.

Модель предоставляет несколько переменных:

- Переменная **Information** (`hostInfo`) собирает данные из различных переменных хоста для создания краткой таблицы с информацией о данном хосте, включая метрики использования ЦП и памяти
- Переменная **VMs List** (`vmsList`) получает доступные данные о *виртуальных машинах*, расположенных на данном хосте и представляет данные в форме таблицы. Таблица содержит большое количество такой информации как имя, состояние, контрольный сигнал, имя гостевой системы, использование ЦП, использование памяти, использование дискового пространства, загрузка сети, имя хоста, IP-адрес, время непрерывной работы
- Переменная **Hypervisors List** (`hvsList`)собирает метрики *гипервизора*: имя датацентра, имя кластера, ссылка на определенную панель *гипервизора* Обзор сервера виртуализации 1906, состояние, IP-адрес, имя продукта, версия продукта, ядра всех процессоров, частота всех процессоров, размер памяти, использование ЦП, использование памяти, использование дискового пространства, загрузка сети, дата последней загрузки. Обычно

переменная содержит только один ряд, так как данные описывают текущий хост, но если модель применяется к *серверу централизованной платформы*, переменная будет содержать информацию о контролируемых *гипервизорах*.

Модель вычисляет переменные один раз в минуту и во время первой загрузки по умолчанию. Она может быть изменена при помощи редактирования параметров привязок модели.

19.1.16.2.2 Модель "Обзор виртуализации"

Модель **Обзор виртуализации** используется для создания списков *Виртуальных машин* и *гипервизоров*. Такие таблицы используются в панелях инструментов Обзор визуализации 1907 и Топ 10 систем виртуализации 1908.

Эта модель абсолютная. Созданные с помощью модели переменные открываются нажатием правой кнопкой мыши по элементу **Редактировать дополнительные свойства** в особом контексте системного дерева.

Модель предоставляет несколько переменных:

- Переменная **VMs List** (`vmsList`) объединяет в себе информацию о в*иртуальных машинах,* предоставленную моделью Мониторинг виртуализации 1905 от всех *гипервизоров* и *серверов централизованной платформы*. Таблица содержит такую информацию как имя, состояние, контрольный сигнал, имя гостевой системы, использование ЦП, использование памяти, использование дискового пространства, загрузка сети, имя хоста, IP-адрес, время непрерывной работы
- Переменная **Hypervisors List** (`hvsList`) объединяет в себе информацию о *гипервизорах* и *серверах централизованной платформы,* предоставленную моделью Мониторинг виртуализации 1905. Таблица содержит такую информацию как имя датацентра, имя кластера, ссылка на определенную панель *гипервизора* Обзор сервера виртуализации 1906, состояние, IP-адрес, имя продукта, версия продукта, ядра всех процессоров, частота всех процессоров, размер памяти, использование ЦП, использование памяти, использование дискового пространства, загрузка сети, дата последней загрузки.

Модель вычисляет переменные один раз в минуту и во время первой загрузки по умолчанию. Она может быть изменена при помощи редактирования параметров привязок модели.

19.1.16.3 Виджеты виртуализации

AtomMind Network Manager предоставляет следующие готовые виджеты *Мониторинга виртуальной инфраструктуры*:

Имя виджета	Описание виджета
VMs Counter	Виджет **VMs Counter** отображает количество виртуальных машин по статусам и контрольным сигналам.
Hypervisor Main Counters	Виджет **Hypervisor Main Counters** отображает основные метрики гипервизора как радиальный индикатор.
Hypervisors Cpu Usage	Виджет **Hypervisors Cpu Usage** представляет собой диаграмму, отображающую использование ЦП для всех *гипервизоров* и *серверов централизованной платформы*. Доступен только для *сервера централизованной платформы*.
Hypervisors Memory Usage	Виджет **Hypervisors Memory Usage** представляет собой диаграмму, отображающую использование памяти для всех *гипервизоров,* контролируемых *сервером централизованной платформы*. Доступен только для *сервера централизованной платформы*.
Hypervisors Disk Usage	Виджет **Hypervisors Disk Usage** представляет собой диаграмму, отображающую использование дискового пространства для всех *гипервизоров,* контролируемых *сервером централизованной платформы*. Доступен только для *сервера централизованной платформы*.
Hypervisors Network Usage	Виджет **Hypervisors Network Usage** представляет собой диаграмму, отображающую загрузку сети для всех *гипервизоров,* контролируемых *сервером централизованной платформы*. Доступен только для *сервера централизованной платформы*.
Cpu Usage Hypervisor VMs	Виджет **Cpu Usage Hypervisor VMs** представляет собой диаграмму с областями, отображающую использование ЦП *гипервизора виртуальными машинами*. Доступен только для *гипервизоров.*

Cpu Usage Hypervisors VMs	Виджет **Cpu Usage Hypervisors VMs** представляет собой диаграмму, отображающую использование ЦП *гипервизоров виртуальными машинами*, контролируемыми *сервером централизованной платформы*. Доступен только для *сервера централизованной платформы*.
Memory Usage Hypervisor VMs	Виджет **Memory Usage Hypervisor VMs** представляет собой диаграмму с областями, отображающую использование памяти *гипервизора виртуальными машинами*. Доступен только для *гипервизоров.*
Memory Usage Hypervisors VMs	Виджет **Memory Usage Hypervisors VMs** представляет собой диаграмму, отображающую использование памяти *гипервизоров виртуальными машинами*, контролируемыми *сервером централизованной платформы*. Доступен только для *сервера централизованной платформы*.
Disk Usage Hypervisor VMs	Виджет **Disk Usage Hypervisor VMs** представляет собой диаграмму, отображающую использование дискового пространства *гипервизора виртуальными машинами*. Доступен только для *гипервизоров.*
Disk Usage Hypervisors VMs	Виджет **Disk Usage Hypervisors VMs** представляет собой диаграмму, отображающую использование дискового пространства *гипервизора виртуальными машинами,* контролируемыми *сервером централизованной платформы*. Доступен только для *сервера централизованной платформы*.
Network Usage Hypervisor VMs	Виджет **Network Usage Hypervisor VMs** представляет собой диаграмму с областями, отображающую загрузку сети *гипервизора виртуальными машинами*. Доступен только для *гипервизоров.*
Network Usage Hypervisors VMs	Виджет **Network Usage Hypervisors VMs** представляет собой диаграмму, отображающую загрузку сети *гипервизоров виртуальными машинами,* контролируемыми *сервером централизованной платформы*. Доступен только для *сервера централизованной платформы*.

Виджеты создаются автоматически в группе **Визуализация**..

Этот виджет относительный и применяется ко всем *гипервизорам* или *серверам централизованной платформы*. Он открывается щелчком правой кнопкой мыши по определенному контексту системного дерева.

19.1.16.4 События виртуализации

Фильтр **События виртуализации** позволяет отображать только события, относящиеся к *гипервизорам* или *серверам централизованной платформы.* Только контексты, имеющие переменные `hvsList` и `vmsList`, подготовленные моделью Мониторинг виртуализации[1908], принимаются как подходящие.

19.1.16.5 Мониторинг VMware

Как один из лидирующих провайдеров технологий виртуализации, VMware предоставляет набор продуктов, включая решения для ПК и серверов. AtomMind Network Manager предоставляет управление и мониторинг для серверов VMware ESX/ESXi и Гостевых Виртуальных Машин (Guest VMs).

Для большей наглядности, мы разделяем мониторинг виртуальных систем на несколько уровней: уровень *сервера*, уровень *машины*, уровень *виртуальной машины* и уровень *операционной системы*. В контексте данной классификации, AtomMind Network Manager предоставляет мониторинг на всех уровнях при помощи как универсальных, так и специфических для VMware инструментов. То есть, на уровне *сервера*, сервер VMware может рассматриваться как стандартная машина Linux, поэтому мониторинг может осуществляться с помощью "универсальных" инструментов мониторинга (смотрите Мониторинг доступности[1853], Мониторинг производительности[1856], Мониторинг сервисов/приложений[1888]). Похожим образом, на уровне *операционной системы*, операционная система, работающая внутри Guest VM, действует как любая другая ОС, поэтому может отслеживаться с помощью того же подхода.

Специфические для VMware инструменты нужны в первую очередь для мониторинга на уровне определенных *виртуальных машин,* так как именно здесь происходят все "магические манипуляции". Н*абор инструментов мониторинга виртуальной машины VMware* AtomMind Network Manager позволяет осуществлять мониторинг определенных виртуальных машин, работающих на определенном сервере VMware. Он включает в себя

Информационный виджет VMware 1916 вместе с набором тревог 1911 и диаграмм 1913. Набор инструментов можно при необходимости индивидуализировать или расширить при помощи создания новых инструментов, используя старые инструменты в качестве шаблонов и/или примеров.

AtomMind Network Manager осуществляет мониторинг виртуальных машин, работающих на серверах VMware через *SNMP*, используя переменные `vmTable`, `cpuTable`, `memTable`, `vmwHBATable` и `vmwNetTable`. Эти переменные конфигурируются автоматически для периодических опросов (для предоставления обновлений значений в режиме реального времени) и хранения исторических значений (для статистического анализа). Настройки опросов могут быть изменены для выполнения определенных задач, используя Параметры синхронизации настроек 502. Обратитесь к документации VMware для получения более подробной информации об активации SNMP на сервере VMware.

19.1.16.5.1 Тревоги VMware

AtomMind Network Manager предоставляет следующие *тревоги VMware* "из коробки":

Имя тревоги	Условие активации	Примечания
Состояние виртуальной машины 1912	**Состояние виртуальной машины** удовлетворяет *условию,* состоящему из *состояния виртуальной машины* (включена, выключена или неактивна) и *операции сравнения*.	Для создания используйте действие Установить профиль мониторинга 1808, группу **VMware**. Доступно только для устройств, совместимых с SNMP.
Загрузка процессора виртуальной машины 1912	**Процент утилизации процессора виртуальной машины** удовлетворяет *условию,* состоящему из *порога* и *операции сравнения*.	Для создания используйте действие Установить профиль мониторинга 1808, группу **VMware**. Доступно только для устройств, совместимых с SNMP.
Утилизация абсолютной памяти виртуальной машины 1912	**Объем утилизации памяти виртуальной машины** удовлетворяет *условию,* состоящему из *порога* и *операции сравнения*.	Для создания используйте действие Установить профиль мониторинга 1808, группу **VMware**. Доступно только для устройств, совместимых с SNMP.
Утилизация относительной памяти виртуальной машины 1912	**Процент утилизации памяти виртуальной машины** удовлетворяет *условию,* состоящему из *порога* и *операции сравнения*.	Для создания используйте действие Установить профиль мониторинга 1808, группу **VMware**. Доступно только для устройств, совместимых с SNMP.
Скорость чтения виртуальной машины с жестких дисков 1912	**Скорость чтения виртуальной машины с жестких дисков** удовлетворяет *условию,* заданному в параметрах тревоги и состоящему из *порога* и *операции сравнения*.	Для создания используйте действие Установить профиль мониторинга 1808, группу **VMware**. Доступно только для устройств, совместимых с SNMP.
Скорость записи виртуальной машины на жесткие диски 1913	**Скорость записи виртуальной машины на жесткие диски** удовлетворяет *условию,* определенному в параметрах тревоги и состоящему из *порога* и *операции сравнения*.	Для создания используйте действие Установить профиль мониторинга 1808, группу **VMware**. Доступно только для устройств, совместимых с SNMP.

Дополнительную информацию см. в соответствующем разделе.

Чтобы создать и настроить тревоги VMware, следует использовать действие Настроить профиль мониторинга 1808. Эти тревоги являются членами *группы тревог VMware*. Таким образом, чтобы настроить тревоги VMware, следует:

- Выбрать устройство сервер VMware;
- Выбрать в контекстном меню элемент Установить профиль мониторинга 1808;
- Проверить элемент `VMware` в колонке `Create Alert`;
- Когда появится окно `Configure VMware Alerts`, добавьте одну или несколько тревог и задайте для них параметры: имя, описание, тип, экземпляр виртуальной машины, для которой должна быть применена тревога, а также параметры установки (доп. информацию см. в соответствующем разделе о тревогах).

Обратите внимание, что присваивая имя, аналогичное имени уже существующей тревоги, Вы не создаете новую тревогу, а скорее добавляете триггер 780 к существующей. Если необходимо создать новую тревогу, убедитесь, что Вы присваиваете ей уникальное имя.

Управлять тревогами и их триггерами можно, используя общий инструментарий AtomMind; описано в разделе Тревоги 779.

19.1.16.5.1.1 Тревога состояния виртуальной машины

Тревога **Состояния виртуальной машины** отслеживает состояние выбранной виртуальной машины. Тревога запускается, если текущее состояние VM удовлетворяет состоянию, заданному в параметрах тревоги. Состояние состоит из *состояния VM* (Включено, Отключено, Приостановлено) и *операции сравнения* (Равно, Не равно).

Пример: если пользователь определил "не равно" для операции сравнения и "включено" - для значения состояния, тревога будет запущена, когда виртуальная машина окажется отключенной или приостановленной.

19.1.16.5.1.2 Тревога загрузки процессора виртуальной машины

Тревога **Загрузка процессора** отслеживает *процент использования CPU* виртуальной машиной. Тревога запускается, если текущее значение удовлетворяет *состоянию,* заданному в параметрах тревоги. Состояние состоит из *предельной величины* (в процентах) и *операции сравнения* (Равно, Не равно, Больше, Меньше, Меньше или равно, Больше или равно).

Пример: если пользователь задал 50 для предельной величины, и "больше" для операции сравнения, тревога будет активирована, когда загрузка CPU превышает 50%.

19.1.16.5.1.3 Тревога утилизации абсолютной памяти виртуальной машины

Тревога **Утилизация абсолютной памяти** отслеживает *объем памяти* (измеряется в килобайтах), используемый определенной виртуальной машиной. Она активируется, когда текущее значение удовлетворяет состоянию, определенному в параметрах тревоги. Условия состоят из *предельной величины* (в килобайтах) и *операции сравнения* (Равно, Не равно, Больше, Меньше, Меньше или равно, Больше или равно).

Пример: если пользователь задал 175000 для предельной величины, и "Не равно" - для операции сравнения, тревога будет активирована, когда память, используемая виртуальной машиной не составляет 175000 килобайтов (т.е. строго больше или строго меньше заданного) .

19.1.16.5.1.4 Тревога утилизации относительной памяти виртуальной машины

Тревога **Утилизация относительной памяти** отслеживает *процент памяти*, использованной определенной виртуальной машиной**.** Она активируется, когда текущее значение удовлетворяет *условию*, заданному в параметрах тревоги. Состояние состоит из *порога* (в процентах) и *операции сравнения* (Равно, Не равно, Больше, Меньше, Меньше или равно, Больше или равно).

Пример: если пользователь задал 50 для предельного значения и "больше" для операции сравнения, тревога будет активирована, когда память использована более, чем на 50% от все доступной для памяти виртуальной машины.

19.1.16.5.1.5 Тревога скорости чтения с жестких дисков виртуальной машины

Тревога **Скорость чтения с жестких дисков** отображает *скорость чтения диска* определенной виртуальной машины. Тревога активируется, если текущее значение удовлетворяет *условию*, заданному в параметрах тревоги, и состоит из *порога* (килобайтов в секунду) и *операции сравнения* (Равно, Не равно, Больше, Меньше, Меньше или равно, Больше или равно).

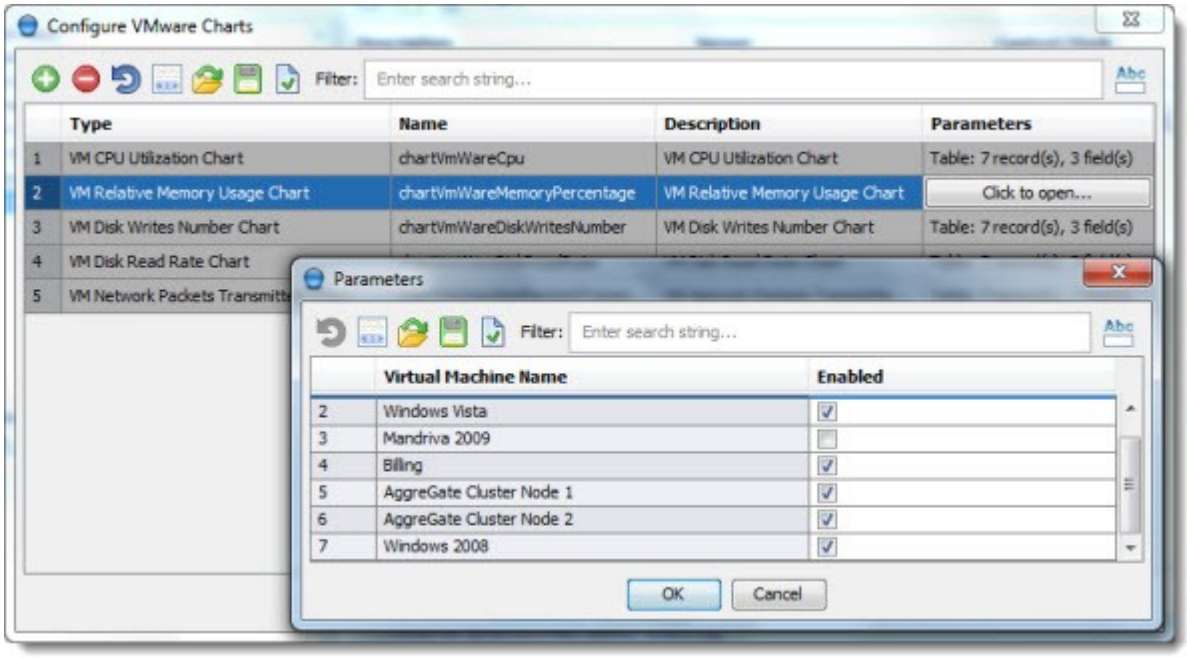
Пример: если пользователь задал 25000 для предельного значения, и "меньше или равно" для операции сравнения, тревога активируется, когда скорость чтения диска не превышает 25000 Kbytes/s.

19.1.16.5.1.6 Тревога скорости записи на жесткие диски виртуальной машины

Тревога **Скорость записи на жесткие диски** аналогична Скорости чтения с жестких дисков 1912, однако она отслеживает *скорость записи на диск* выбранной виртуальной машины.

19.1.16.5.2 Диаграммы VMware

AtomMind Network Manager включает группу графиков, отображающих данные для *виртуальной машины*.



Диаграммы можно создавать и настраивать через группу *VMware* действия Настройка профиля мониторинга 1808:

- диаграмма *использование процессора* 1914
- четыре диаграммы *производительности диска* 1915: Запись, *Чтение, Скорость записи* и Скорость чтения
- два графика *использования памяти* 1915 (*абсолютного и относительного*)
- четыре графика *производительности сети* 1915: *переданные по сети пакеты, полученные по сети пакеты, скорость передачи по сети, и скорость получения по сети*

Группа *VMware* доступна для контекстов устройства с SNMP-переменной `vmTable`.

Чтобы настроить один или несколько диаграмм VMware, используя действие настроить профиль мониторинга 1808, следует:

- Выбрать контекст устройства сервера VMware
- Выбрать в контекстном меню элемент Настроить профиль мониторинга 1808
- Проверить элемент `VMware` в колонке `Create Chart`
- В окне `Configure VMware Charts` , добавьте столько диаграмм, сколько необходимо, и настройте их, определив тип (один из упомянутых ранее), имена, описания (как описано в подразделе Настройка диаграмм 1808 в Настройке профиля мониторинга 1808), и *параметры*. Все типы диаграмм VMware имеют одинаковые параметры: просто проверьте виртуальные машины, метрические данные которых должны быть описаны.
- Будет создан Виджет 943 диаграммы с заданным именем и описанием. Вы можете найти его под узлом Виджеты в Системном дереве 370, или активировать из контекстного меню устройства (имя элемента меню будет таким же как и описание диаграммы).

Диаграммы статуса VMware

Имя диаграммы	Описание	Примечания
Загруженность процессора виртуальной машины 1914	Указывает количество емкости процессора, используемой виртуальной машиной в настоящее время.	Диаграмму можно создать для определенного SNMP-контекста, используя действие Установить профиль мониторинга 1808.
Количество записей на диск виртуальной машины 1915	Отображает использование диска, измеряемое как *количество операций записи на диск*, выполненных с момента последней синхронизации.	Диаграмму можно создать для определенного SNMP-контекста, используя действие Установить профиль мониторинга 1808.
Количество операций чтения с диска виртуальной машины 1915	Отображает использование диска, измеряемое как *количество операций чтения с диска*, выполненных с момента последней синхронизации.	Диаграмму можно создать для определенного SNMP-контекста, используя действие Установить профиль мониторинга 1808.
Скорость записи на диск виртуальной машины 1915	Отображает скорость записи на диск в килобайтах в секунду.	Диаграмму можно создать для определенного SNMP-контекста, используя действие Установить профиль мониторинга 1808.
Скорость операций чтения с диска виртуальной машины 1915	Отображает скорость чтения с диска в килобайтах в секунду.	Диаграмму можно создать для определенного SNMP-контекста, используя действие Установить профиль мониторинга 1808.
Объем памяти виртуальной машины 1915	Указывает текущее использование памяти в килобайтах.	Диаграмму можно создать для определенного SNMP-контекста, используя действие Установить профиль мониторинга 1808.
Процент памяти виртуальной машины 1915	Указывает процент текущего использования памяти, т.е. памяти, используемой относительно количества памяти виртуальной машины, с которой она была установлена.	Диаграмму можно создать для определенного SNMP-контекста, используя действие Установить профиль мониторинга 1808.
Переданные сетевые пакеты виртуальной машины 1915	Отображает количество переданных пакетов.	Диаграмму можно создать для определенного SNMP-контекста, используя действие Установить профиль мониторинга 1808.
Полученные сетевые пакеты виртуальной машины 1915	Отображает количество полученных пакетов.	Диаграмму можно создать для определенного SNMP-контекста, используя действие Установить профиль мониторинга 1808.
Скорость передачи сети виртуальной машины 1915	Отображает текущую скорость передачи.	Диаграмму можно создать для определенного SNMP-контекста, используя действие Установить профиль мониторинга 1808.
Скорость получения сети виртуальной машины 1915	Отображает текущую скорость получения.	Диаграмму можно создать для определенного SNMP-контекста, используя действие Установить профиль мониторинга 1808.

19.1.16.5.2.1 Диаграмма использования процессора виртуальной машиной

Диаграмма **Использование процессора** указывает, сколько от общей мощности процессора использует в настоящий момент виртуальная машина (-ы). Значение в диаграмме представляет собой производную по времени 1801 от общего времени, сколько виртуальная машина использует CPU (отображено SNMP-переменной `cpuTable`).

При настройке опций синхронизации 502, следует убедиться, что период синхронизации `cpuTable` не слишком маленький - см также раздел Обработка данных 1800.

19.1.16.5.2.2 Диаграммы использования памяти виртуальной машины

Существуют две диаграммы, отображающие использование памяти виртуальной машиной (-ами):

- **Использование абсолютной памяти**
- **Использование относительной памяти**

Обе диаграммы используют значение memUtil в SNMP-переменной memTable для выбора текущей памяти, используемой определенной виртуальной машиной. Диаграмма **Использование абсолютной памяти** отображает memUtil как абсолютное значение, *отображающее использование памяти в килобайтах*, в то время как диаграмма **Использование относительной памяти** отображает это значение относительно объема памяти, на которое виртуальная машина была настроена, отображая *использования памяти в процентном выражении*.

19.1.16.5.2.3 Диаграммы производительности диска виртуальной машины

Эти диаграммы отображают текущее использование диска виртуальной машиной(-ами).

Диаграммы **Количество чтения/записей на диск** изображают графически использование диска, измеряемое как количество операций чтения/записи на диск, выполненных с момента последней синхронизации. Используемые значение - это в действительности дифференциация 1799 значений numReads (numWrites) , предоставленных SNMP-переменной vmwHBATable.

Диаграммы **Скорость чтения/записи на диск** отображают графически скорость чтения/записи на диске, подсчитанную как производные по времени значений kbRead / kbWritten (итоговое количество килобайт, прочитанных/записанных на диск за секунду) и выраженную в SNMP-переменной vmwHBATable.

Обратите внимание на моменты, касающиеся *слишком короткого периода синхронизации*, которые упомянуты в разделе Обработка данных 1800, при настройке опции синхронизации 502 vmwHBATable.

19.1.16.5.2.4 Диаграммы производительности сети виртуальной машины

Диаграммы описывают *активность сети* виртуальной машины, измеряемую пакетами данных и/или полученными/переданными килобайтами и выраженную SNMP-переменной vmwNetTable:

Диаграмма **Переданные пакеты по сети** отображает количество пакетов, переданных с момента последней синхронизации, как дифференциацию общего количества переданных пакетов (pktsTx).

Диаграмма **Полученные пакеты по сети** отображает количество пакетов, полученных с момента последней синхронизации, как дифференциацию общего количества полученных пакетов(pktsRx).

Диаграмма **Скорость передачи по сети** отображает текущую скорость передачи, как производное от общего количества отправленных Кб (kbTx).

Диаграмма **Скорость получения по сети** отображает текущую скорость получения по сети, как производное от общего количества полученных Кб (kbRx).

Обратите внимание на моменты, связанные со слишком коротким периодом синхронизации (см раздел Обработка данных) 1800, при настройке опций синхронизации 502 vmwNetTable.

19.1.16.5.3 Отчеты VMware

Отчеты и запросы для виртуальных машин VMware создаются автоматически в группе **VMware**.

Краткие сведения о VMware

Отчет **Краткие сведения о VMware** отображает следующую информацию для всех виртуальных машин, доступных на зарегистрированных устройствах:

- Описание устройства хоста
- Отображаемое имя виртуальной машины
- Текущеее состояние виртуальной машины
- Имя гостевой ОС, связанной с виртуальной машиной

- Состояние гостевой ОС.

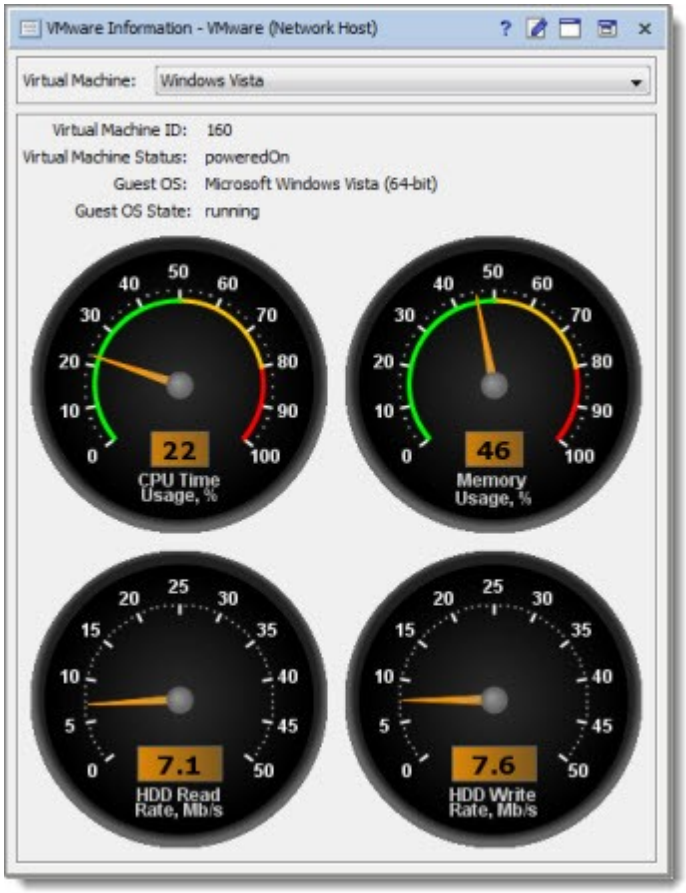
Отчет ссылается на запрос **Краткие сведения о VMware**.

Использование памяти VMware

Отчет об использовании памяти VMware отображает *настроенный объем памяти, использованный объем памяти и процентное соотношение* для всех виртуальных машин, доступных на зарегистрированных устройствах. Отчет ссылается на запрос об **Использовании памяти VMware**.

19.1.16.5.4 Информационный виджет VMware

Информационный виджет VMware отображает информацию о виртуальных машинах, зарегистрированных на определенном сервере VMware.



Чтобы запустить виджет для сервера VMware, следует выбрать соответствующий контекст в Системном дереве, а затем выбрать элемент меню `VMware Information`.

Виджет позволяет выбрать одну из виртуальных машин, зарегистрированную на сервере и отображающую следующую информацию:

- *ID виртуальной машины*
- *текущий статус* выбранной виртуальной машины
- *операционная система*, запущенная на виртуальной машине ("*гостевая ОС*")
- *текущее состояние* гостевой ОС
- *Ресурсы*, используемые виртуальной машиной:
  - Использование *времени CPU*
  - Процент использования *памяти*
  - *Скорость чтения HDD*
  - *Скорость записи HDD*

Виджет можно настроить в редакторе виджетов 423 (см Виджеты 943).

Виджет и соответствующий элемент меню доступны, если SNMP-переменная `vmTable` для устройства сервиса VMware имеет как минимум одну запись.

19.1.17 Мониторинг принтеров

Инструментарий **мониторинга принтеров** AtomMind Network Manager включает виджет Информация о принтере 1919 и ряд тревог 779: Состояние принтера 1918, Маркер принтера 1918, Статус крышки принтера 1918 и SNMP-ловушка о критическом событии для принтера 1887. Эти инструментальные средства используют протокол SNMP для того, чтобы получить данные от принтеров и поддерживают выполнение таких обычных задач мониторинга принтера как:

- отобразить описание принтера, модель, серийный номер, описать возможности, текущее состояние и предоставить другую информацию о принтере.
- зафиксировать и получить уведомление о проблемах, таких как замятие бумаги, отсутствие бумаги, переполненность лотка и пр.
- получить уведомление (заранее) об осутствии ресурсов: тонера, фотобарабана, проявителя и пр.

В то же время не является сложностью расширение этих основных возможностей, путем создания пользовательских инструментальных средств для мониторинга принтера особого типа или модели. Этот инструментарий может использовать SNMP-опрос или обрабатывать события принтера (прослушивая SNMP-ловушки или Syslog-сообщения, отправляемые принтерами).

За информацией об управлении и мониторинге принтера обратитесь к RFC 3805.

Настройка

Настройка мониторинга принтера 1917 включает спецификацию среды (статусы принтера, опции синхронизации) и ряд специфичных инструментальных средств (информационный виджет принтера и несколько типов тревог), включая их настройки.

СОСТОЯНИЯ ПРИНТЕРА

Состояние принтера - это важный параметр для мониторинга принтера, который используют тревога состояния принтера 1918 и информационный виджет принтера 1919. AtomMind Network Manager определяет состояние принтера, зависящего от значений трех SNMP-переменных: `Device Status`, `Printer Status` и `Printer Detected Error State`. Их значения включаются в одно состояние принтера при помощи переменной таблицы **Состояния принтера**, которая доступна в свойствах устройства 1494 принтера. Метод объединения этих значений можно изменить для определенного устройства принтера, чтобы отображать специфические свойства.

Следующие состояния принтера определяются по умолчанию:

Некритические состояния принтера	**Критические состояния принтера**
Обычный	Замятие бумаги
В печати	Крышка не закрыта
В режиме офлайн	Отсутствует подающий лоток
В состоянии готовности	Подающий лоток пуст
Первоначальное включение	Выходной лоток отсутствует
Разогрев/подготовка к работе	Выходной лоток заполнен
Подающий лоток ниже нормы	Отсутствует краска
Выходной лоток почти полный	Запас краски исчерпан
Мало краски	
Отсутствует подающий лоток	
Подающий лоток пустой	
Отсутствует выходной лоток	
Выходной лоток заполнен	

При *критическом* состоянии принтер перестает работать, и печать не может быть продолжена до тех пор, пока проблема, вызвавшая это состояние, не будет решена.

Автоматически создается таблица состояний принтера по умолчанию для устройств, предоставляющих SNMP-переменную `prtGeneralTable` . Она аналогична описанной в *Приложении E - Сводной таблице статусов принтера* RFC 3805.

Тревоги мониторинга принтеров

Имя тревоги	Условие активации	Примечания
Состояние принтера (SNMP) 1918	**Состояние принтера** равно или отличается от выбранного значения.	Используйте действие Установить профиль мониторинга 1808, группу **Принтер** для создания. Доступно только для устройств, совместимых с SNMP.
Краска принтера (SNMP) 1918	**Уровень краски принтера** определенного типа упал ниже заданного порога.	Используйте действие Установить профиль мониторинга 1808, группу **Принтер** для создания. Доступно только для устройств, совместимых с SNMP.
Крышка принтера (SNMP) 1918	Любая крышка **принтера** имеет определенное состояние.	Используйте действие Установить профиль мониторинга 1808, группу **Принтер** для создания. Доступно только для устройств, совместимых с SNMP.

19.1.17.1 Тревожное сообщение о состоянии принтера

Тревоги 779 **состояния принтера** помогают отслеживать текущее состояния принтера, как описано в RFC 3805. Тревоги такого типа можно создавать при помощи действия Установить профиль мониторинга 1808 и настраивать, когда статус принтера соответствует или отличается от выбранного значения состояния принтера 1917.

19.1.17.2 Тревога о состоянии краски в принтере

Тревога 779 об **отсутствии краски в принтере** позволяет ощуствлять контроль за *наличием краски в принтере* (см. RFC 3805 и, в частности, раздел *2.2.6. Маркеры*). Тревога настраивается через параметры `Marker Supplies Type` и `Threshold`. Она активизируется, когда уровень определенного маркера оказывается ниже предельной величины. Значения, доступные в поле `Marker Supplies Type`, зависят от типа принтера (см. **PrtMarkerSuppliesTypeTC** в RFC 3805).

Например, у принтера *Sharp MX-2300N*: *Черный тонер, Отработанный тонер, Черный фотобарабан, Черный проявитель* и *Узел закрепления*. В свою очередь, элементы предельной величины зависят от выбранного типа маркера (см PrtMarkerSuppliesSupplyUnitTC в RFC 3805). Например, элемент Черный тонер для принтера *Sharp MX-2300N* выражен в процентах.

AtomMind Network Manager использует SNMP-переменную `prtMarkerSuppliesTable` , чтобы отслеживать текущие значения запаса маркеров.

19.1.17.3 Тревога о статусе крышки принтера

Тревога 779 о **статусе крышки принтера** позволяет отслеживать состояние крышки принтера. Эту тревогу настраивает действие Установить профиль мониторинга 1808. У тревог такого типа лишь один специфический параметр: *Переключение состояния крышки*. Тревога будет активироваться, если любая из крышек согласно отчету оказывается в указанном состоянии.

19.1.17.4 Информационный виджет принтера

Информационный виджет[943] **принтера** доступен для всех устройств, предоставляя значения `hrDeviceTable` и `hrPrinterTable`. Его можно активизировать, выбрав соответствующее действие из контекстного меню для устройства принтера. Этот виджет создается и настраивается автоматически, в большинстве случаев не требуется дополнительная адаптация. В то же время не сложно изменить настройки[423] виджета с тем, чтобы он отражал состояние для новых, с улучшенными свойствами моделей принтеров.

Виджет отображает следующие данные:

- Общая информация о принтере
  - *Имя*
  - *Описание* (обычно производитель/модель принтера)
  - *Серийный номер*
  - *Физический адрес (MAC)*
- *Состояние принтера* (как описано в случае состояние принтера[1917] в разделе *Настройка мониторинга принтера*)
- Счетчик колеров принтера
- Запас тонера (см также раздел тревога о количестве тонера в принтере[1918])

Значения обновляются согласно значениям `Synchronization Period` для `hrDeviceTable`, `hrPrinterTable` и `prtMarkerSuppliesTable` (см раздел Опции синхронизации)[502].

19.1.18 Мониторинг беспроводных устройств

AtomMind Network Manager может отслеживать различные **беспроводные устройства** по SNMP. Необходимые MIB-файл (IEEE802dot11-MIB) включен в дистрибутивный пакет.

Инструментальная панель обзора сетевых устройств отображает следующие метрики "из коробки":

- SSID устройства
- Тип BSS
- Алгоритм аутентификации
- Текущий режим CCA
- Описание интерфейса
- Код страны
- Тип интерфейса
- Настройка MTU
- Состояние беспроводного интерфейса
- Рабочее состояние беспроводного интерфейса
- Адрес MAC
- MAC наиболее передаваемых фреймов устранения связи
- MAC наиболее передаваемых фреймов деаутентификации
- MAC неуспешно завершенных фреймов аутентификации

Также возможно отслеживать специфичную для производителя информацию о беспроводном устройстве. Например, беспроводные устройства Cisco предоставляют следующую подробную информацию через SNMP:

- Количество клиентов, мостов и повторителей, доступных через каждый беспроводной интерфейс
- Подробная информация о каждом беспроводном клиенте (имя, состояние, адрес, роль, класс устройства, RSSI, объем трафика и другое)

19.1.19 Мониторинг модуля распределения питания

AtomMind Network Manager предлагает коробочный мониторинг модулей распределения питания по протоколу SNMP.

Инструментальная панель **Мониторинг модуля распределения питания** включает в себя виджет **Информация о модуле распределения питания**, который показывает:

- Полная мощность
- Сила тока
- Активная мощность
- Напряжение

19.1.20 Мониторинг систем бесперебойного электроснабжени

AtomMind Network Manager предлагает "коробочный" мониторинг сетевых устройств бесперебойного электроснабжения по SNMP, поддерживающих PowerNet-MIB.

Инструментальная панель Информация об устройстве бесперебойного электроснабжения включает виджет **Информация об устройстве бесперебойного электроснабжения**, отображающий:

- Датчики напряжения входной и выходной линии
- Датчики частоты входного и выходного напряжения
- Датчик выходной нагрузки
- Датчик емкости батареи
- Модель устройства
- Время работы батареи
- Состояние, температура и индикатор замены батареи

19.1.21 Мониторинг и управление Java

Функциональность AtomMind Network Manager по мониторингу и управлению Java позволяет контролировать и отслеживать:

- Приложения и сервисы на базе Java
- Серверы приложения (Tomcat, JBoss, WebLogic, WebSphere, SunOne, Jetty, сервер приложений Oracle и др.)

Управление приложениями Java позволяет:

- Иметь доступ к стандартным и специфичным для приложений MBeans.
- Просматривать, записывать в журнал и изменять атрибуты MBean.
- Выполнять операции MBean по требованию, по расписанию или в ответ на тревоги.
- Хранить и реагировать на уведомления о событиях MBean.
- Использовать полный объем инструментальных средств интегрированных данных (тревоги 779, отчеты 928, датчики 2181, виджеты 943 и пр.) для доступа к данным JMX.

Функциональность JMX-мониторинга реализует драйвер устройства JMX 571.

Встроенные стредства мониторинга виртуальной машины и приложений Java

Дистрибутив AtomMind Network Manager включает инструментальные панели 912, помогающие отслеживать виртуальные машины Java, распространенные приложения Java и сервера приложений:

ОБЗОРНАЯ ИНСТРУМЕНТАЛЬНАЯ ПАНЕЛЬ JAVA

Эта инструментальная панель предоставляет общую информацию о виртуальной машине Java:

- Свойства среды виртуальной машины
- Статистика сбора мусора
- Статистика классов

- Статистика потоков
- График нагрузки на процессор, вызванной виртуальной машиной

Для входа в эту инструментальную панель дважды кликните на учетной записи любого устройства Java/JMX.

ИНСТРУМЕНТАЛЬНАЯ ПАНЕЛЬ ПАМЯТИ JAVA

Эта инструментальная панель предлагает подробную статистику памяти виртуальной машины Java:

- Графики динамической и нединамической памяти
- Подробная информация о пулах памяти

Для входа в эту инструментальную панель дважды кликните на учетной записи любого устройства Java/JMX.

19.1.22 Мониторинг .NET

Задачу мониторинга платформы .NET можно разделить на две категории:

- Мониторинг состояния приложений необходим для постоянного отслеживания рабочих характеристик приложений, чтобы поддерживать их штатную работоспособность. Такой мониторинг позволяет выявлять возможные причины ухудшения производительности или сбоев еще на ранних этапах, до того, как произойдет инцидент
- Мониторинг производительности, как правило, служит средством выявления причин уже проявившихся проблем, а также для оптимизации приложений и устранения узких мест

AtomMind Network Manager является прекрасным инструментом для решения общих задач мониторинга .NET. Система предоставляет комплексную информацию по вашим .NET-приложениям и помогает независимо анализировать функционирование различных подсистем, таких как подсистема ввода-вывода, нагрузка на центральный процессор, эффективность использования памяти, определение наличия утечек памяти. Также можно определить требования к производительности при масштабировании приложения.

Конфигурация мониторинга .NET

Для конфигурации мониторинга .NET:

- Создайте аккаунт WMI в узле **Устройства** Системного дерева 370
- Кликните правой кнопкой мыши по аккаунту
- Выберите **Настроить профиль мониторинга** в контекстном меню
- Напечатайте **Создать запросы** в таблице и нажмите OK
- Удалите некоторые запросы из таблицы при необходимости и просто нажмите OK
- Откройте Инструментальную панель двойным щелчком по аккаунту устройства

Ключевые метрики .NET:

Метрики мониторинга .NET включают:

- Физические и логические потоки
- Длина очереди
- Блокировки синхронизации
- Контексты
- Исключения
- Блокировки
- Использование и распределение памяти
- Эффективность и работа сборщика мусора
- Домены приложений
- Сборки
- Загруженные классы

- Вызываемые COM оболочки
- Заглушки
- Проверки выполнения кода
- JIT компиляции
- И другие

19.1.23 Управление соответствием стандартам и конфигураци

Некоторые версии AtomMind Network Manager включают модуль, называемый *Управление соответствием стандартам и конфигурациями*. Он помогает сетевым администраторам:

- Резервировать и отслеживать конфигурации сетевых устройств
- Выполнять централизованное пакетное изменение конфигураций устройств и операций управления
- Заранее реагировать на различные нарушения политики конфигураций
- Организовать отчеты о соответствии стандартам

19.1.23.1 Управление конфигурациями

Каждая установка AtomMind Network Manager в большой сети имеет огромную базу данных подключенных устройств, включая маршрутизаторы, коммутаторы, беспроводные точки доступа, фаерволы и другие. Эта база данных заполняется, преимущественно используя сетевое обнаружение[1811].

Конфигурации всех этих устройств могут периодически извлекаться сервером и сохраняться на сервере базы данных. В большинстве случаев это не требует дополнительной конфигурации, исключая определение учетных данных устройств.

Системные операторы будут немедленно оповещены, если будет обнаружено изменение конфигурации периодическим опросом или событие изменения конфигурации будет получено от устройства (т.е. в виде Syslog сообщения или SNMP-ловушки). Это позволяет в режиме реального времени в масштабах всей сети производить мониторинг конфигураций и быстрой изоляции инцидентов, связанных с их ошибочным изменением.

Вся история конфигураций сохраняется на сервере в базе данных пока они не устареют в соответствии с предварительно настроенными условиями. Эти условия, а также расписания резервного копирования могут быть изменены системным администратором.

Администраторы могут просматривать прошлые конфигурации в базе данных в любое время и легко сравнить несколько конфигураций в визуальном просмотрщике. Поддерживаемые типы сравнений:

- Сравнение между историей версий одной конфигурации
- Сравнение конфигураций различных устройств
- Сравнение разных типов конфигураций, например, startup с runtime

Любая конфигурация может быть повторно загружена на устройство в несколько кликов мыши. Функция базовой конфигурации позволяет выделить одно, несколько или все конфигурации, которые "известно, что работают" и быстро к ним вернуться в случае возникновения проблем.

Операции с устройствами любых производителей

Конфигурации устройств могут быть получены из широкого диапазона сетевых устройств разных производителей. Поддерживается несколько способов получения конфигураций для каждого из устройств различных производителей. Например, устройство Cisco может управляться через SSH, Telnet или SNMP, а актуальная конфигурация может быть записана или прочитана с использованием FTP, TFTP или SCP.

Тем не менее, все конфигурации различных производителей хранятся в общей системе, что позволяет использовать единый подход для их проверки и сравнения.

19.1.23.1.1 Установка управления конфигурациями

Для установки централизованного управления конфигурациями необходимо:

- Настроить или скорректировать скрипты резервного копирования конфигураций в глобальных настройках модуля Управление конфигурациями и изменениями

- Включить и настроить управление конфигурациями для выбранных учетных записей устройств 497

Глобальные настройки управления конфигурациями и соответствием стандартам

Общие настройки конфигураций и соответствия стандартам доступны в глобальной конфигурации плагина 207 CCM. Для доступа к конфигурации:

- Разверните узел **Драйвера/Плагины** ( ) в системном дереве 370.
- Дважды кликните на плагине **Управление конфигурациями и соответствием стандартам** ( ).

ТАБЛИЦА КОНФИГУРАЦИЙ

Конфигурация плагина производится через таблицу **Конфигурации**, имеющую следующие поля:

- **Имя**. Описание типа/семейства/производителя устройства, которому соответствует данная запись, например `Cisco` или `D-Link DFL-800`.
- **Последовательность резервного копирования**. Набор правил 1924, используемых для настройки резервного копирования из устройств этого типа или семейства.
- **Формат параметров резервного копирования**. Формат 50 настроек резервного копирования конфигураций по устройствам. Эти настройки отдельно настраиваются в каждой учетной записи CCM-устройства и затем используются правилами **Последовательности резервного копирования**.
- **Последовательность восстановления**. Набор правил 1924, используемых для восстановления конфигураций устройств этого типа или семейства.
- **Формат параметров восстановления**. Формат 50 настроек восстановления конфигураций по устройствам. Эти настройки отдельно настраиваются в каждой учетной записи CCM-устройства и затем используются правилами **Последовательности восстановления**.

Настройка резервного копирования конфигураций и восстановления для устройства

Управление конфигурациями устройства контролируется через свойство **Управление конфигурациями и изменениями**, на которое можно выйти через диалоговое окно **Редактировать свойства устройства**. Его поля:

- **Включение**. Контролирует, активно ли управление конфигурациями для данного устройства.
- **Выражение резервного копирования конфигураций**. Это выражение AtomMind 112, которое должно вернуть текст конфигурации. Ее вывод будет конвертирован в строку. Выражение резервного копирования конфигурации по умолчанию предназначено для вызова последовательности резервного копирования конфигурации, определенного полем **Последовательность резервного копирования** и наполнения его параметрами, определенными полем **Параметры резервного копирования** (см. ниже).

<table>
<tr><td colspan="2">Среда вычисления 114 выражения резервного копирования конфигурации:</td></tr>
<tr><td>Контекст по умолчанию 119</td><td>Контекст устройства, чья конфигурация восстанавливается.</td></tr>
<tr><td>Таблица данных по умолчанию 120</td><td>Таблица с одной строкой следующего формата 50:
<table>
<tr><th>Имя поля</th><th>Тип поля</th><th>Примечания</th></tr>
<tr><td>configurationType</td><td>Строка</td><td>Тип конфигурации для резервного копирования, например, <code>startup</code> или <code>runtime</code>.</td></tr>
</table></td></tr>
<tr><td>Строка по умолчанию 119</td><td>0</td></tr>
<tr><td>Переменные среды 123</td><td>Стандартные 123 переменные.</td></tr>
</table>

- **Выражение восстановления конфигурации**. Это выражение AtomMind 112, которое должно записывать конфигурацию в устройство (предоставляемую через переменную среды 123 `configuration`). Результат выражения будет игнорироваться. Выражение восстановления конфигурации по умолчанию предназначено для вызова последовательности восстановления конфигурации, определяемой полем **Восстановить последовательность**, и заполнения ее параметрами, определенными полем **Восстановить параметры** (см. ниже).

Среда вычисления 114 выражения восстановления конфигурации:	
Контекст по умолчанию 119	Контекст устройства, чья конфигурация резервируется.
Таблица данных по умолчанию 120	Таблица с одной строкой следующего формата 50: **Имя поля** / **Тип поля** / **Примечания** configurationType / Строка / Тип конфигурации для восстановления, например `startup` или `runtime`.
Строка по умолчанию 119	0
Переменные среды 123	Стандартные 123 переменные. Дополнительные переменные: **Имя переменной** / **Тип значения** / **Описание** configuration / Строка / Текст конфигурации устройства для восстановления.

- **Максимальное количество сохраненных конфигураций**. Определяет, сколько исторических конфигураций будет сохраняться для устройства. Как только новые процессы резервного копирования наталкиваются на изменения конфигурации, а список конфигураций обновляется, старые конфигурации очищаются для поддержания определенного размера списка.
- **Последовательность резервного копирования**. Позволяет указывать последовательность резервного копирования 1924.
- **Параметры резервного копирования**. Позволяет определять параметры для резервного копирования конфигурации данного устройства (такие как имя пользователя Telnet, пароль и т.д.). Список параметров определяется выбранной **Последовательностью резервного копирования**.
- **Последовательность восстановления**. Позволяет указывать последовательность восстановления 1924.
- **Параметры восстановления**. Позволяет определять параметры для восстановления конфигурации данного устройства (такие как имя пользователя Telnet, пароль и т.д.). Список параметров определяется выбранной **Последовательностью восстановления**.
- **Фильтр сравнения**. Регулярное выражение для исключения какого-либо текста из сравнения конфигураций.

19.1.23.1.2 Последовательности резервного копирования и восстановления

Последовательности резервного копирования и *восстановления* - это наборы правил, используемые для резервного копирования и восстановления конфигураций устройств.

Каждая последовательность (набор правил) используется для чтения и записи конфигураций из/на устройства определенного производителя, семейства, модели или марки. Сфера применения набора правил варьируется в зависимости от того, как она построена и насколько схожи резервное копирование/восстановление конфигураций, выполняемые с различными устройствами определенного производителя.

Все последовательности (наборы правил) полностью независимы друг от друга. Каждый набор включает одно или более правил, работающих в порядке последовательности. Каждый набор правил может иметь определенные параметры, и его обработка всегда имеет результатом один объект (любого типа, например, число, строка или дата).

Компоненты правил

Каждое правило в наборе состоит из цели, выражения и условия. Правило либо возвращает результат всего набора правил, либо (пере-)определяет одну переменную среды, действительную до конца обработки набора правил.

Также возможно добавлять комментарии к отдельным правилам.

Каждое правило выполняет одно из следующих действий:

- Выполняет некоторое действие, например, выполняет скрипт Telnet/SSH
- Рассчитывает определенные временные значения
- Извлекает и возвращает текст конфигурации устройства

ЦЕЛЬ ПРАВИЛ

Цель правила определяет, как обрабатывать результат этого правила. Если цель правила - **Результат финального набора правил**, это правило прекращает обработку последовательности. Если правило является частью Последовательности резервного копирования, оно должно вернуть текст конфигурации устройства (значение Строка). Если это часть Последовательности восстановления, результат набора правил игнорируется.

В других случаях цель определяет имя переменной среды, получаемое, как только заканчивается обработка правила. На эту переменную можно ссылаться из выражений и условий других правил.

ВЫРАЖЕНИЕ ПРАВИЛ

Выражение правил - это выражение AtomMind 112, возвращающее результат любого типа. Этот результат:

- Возвращается как результат всего набора правил, если Цель правила - это **Результат финального набора правил**.
- Хранится как переменная внутренней среды набора правил, переписывающая значение этой переменной, ели оно уже было определено другими правилами. Переменная среды действительна во время текущего цикла обработки набора правил.

Выражения правил могут ссылаться на результаты других правил из этого набора через ссылки {env:*rule_target_variable_name*}.

Среда вычисления 114 выражения правил:	
Контекст по умолчанию 119	Контекст устройства, чья конфигурация резервируется или восстанавливается.
Таблица данных по умолчанию 120	Резервирует или восстанавливает таблицу параметров. Формат таблицы определяет **Формат параметров резервного копирования** или **Формат параметров восстановления**, определенные в той же строке Таблицы конфигураций 1923, имеющей текущую последовательность резервного копирования/восстановления.
Строка по умолчанию 119	0
Переменные среды 123	Стандартные 123 переменные. Переменные среды, определенные прежде выполняемыми правилами той же последовательности.

УСЛОВИЕ ПРАВИЛ

Условие правил - это выражение AtomMind 112, которое должно выводить булевое значение. Если это значение false, обработка правил опускается и обрабатывается следующее правило.

Условия правил могут относиться к результатам других правил из этого набора через ссылки {env:*rule_target_variable_name*}.

Условие правил опционально.

Среда вычисления 114 условия правила:	
Контекст по умолчанию 119	Контекст устройства, чья конфигурация резервируется или восстанавливается.
Таблица данных по умолчанию 120	Резервирует или восстанавливает таблицу параметров. Формат таблицы определяется **Форматом параметров резервного копирования** или **Форматом параметров восстановления**, указанными в той же записи Таблицы конфигураций 1923, имеющейся в текущей последовательности резервного копирования/восстановления.
Строка по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

	Переменные среды, определенные ранее выполненными правилами той же последовательности.

Пример последовательности резервного копирования

Этот пример описывает набор правил, который извлекает конфигурацию из радио модема. Он выполняет следующие действия:

- Входит в модем, используя протокол Telnet, и выполняет серии команд, выводящих текущую конфигурацию в его внутренний флеш-диск
- Входит в модем, используя протокол FTP, и загружает файл конфигурации

Цель	Выражение	Условие	Примечания
Пустая команда	`{executeExpectScript("Save Configuration", "Telnet")}`		
Результат финального набора правил	`{ftpDownload('{.:connectionProperties$address}', '{.:ftpSettings$port}', '{ftpDirectory}', '{ftpFileName}', '{.:ftpSettings$userName}', '{.:ftpSettings$password}')}`		

Первое правило выполняет Ожидание скрипта 613 путем вызова функции 70 `executeExpectScript()`. Скрипт (не являющийся частью данного примера) посылает устройству серии команд для вызова конфигурации сохранения на внутреннем диске, который становится доступен для загрузки через протокол FTP. Результат правила не учитывается (он переносится в переменную под названием `Dummy`, которая не будет использоваться последующими правилами).

Второе правило загружает правило конфигурации из устройства путем выполнения функции `ftpDownload()`. Функция принимает адрес, порт, логин, пароль, путь удаленной директории и имя файла устройства для загрузки в качестве параметров. Некоторые параметры берутся из контекста учетной записи устройства (через ссылки 118, такие как `{.:connectionProperties$address}` или `{.:ftpSettings$userName}`), при этом другие берутся из таблицы по умолчанию 120, содержащей свойства, описанные в **Формате параметров резервного копирования** (например, функция `{ftpDirectory}` или `{ftpFileName}`). Функция `ftpDownload()` возвращает контент загруженного файла конфигурации устройства, который, как результат, возвращается из последовательности резервного копирования.

19.1.23.1.3 Резервное копировние и восстановление конфигурации

В нормальном режиме работы AtomMind Network Manager выполняет периодическое резервное копирование конфигураций для всех устройств, которые были правильно настроены для управления конфигурациями. Восстановление конфигурации - это проводимая вручную операция, которая происходит, если определенное устройство испытывает трудности.

Как только от устройства поступает новая конфигурация, она добавляется к истории конфигураций сервера, только если она имеет различия с предыдущим резервным копированием.

Типы конфигураций

Каждое устройство, управляемое AtomMind Network Manager, может иметь несколько *типов* конфигураций. Каждый тип конфигурации резервируется, его тип сохраняется и хранится вместе с текстом конфигурации.

Пример: Большинство устройств Cisco имеет два типа конфигураций: конфигурация *startup*, сохраняемая во флеш-памяти, и конфигурация *running*, включающая изменения, сделанные с момента запуска устройства.

Типы конфигурации:

- Доступны в **Выражении чтения конфигурации**, **Выражении записи конфигурации**, **Последовательности резервного копирования** и **Последовательности восстановления**. Это позволяет инструментам определять свое поведение типом конфигурации.
- Сохраняются в конфигурационной базе данных и отображаются в списке исторических конфигураций.

Ручное резервное копирование конфигураций

Чтобы вручную запустить немедленное резервное копирование конфигурации устройства, запустите действие 87 **Резервное копирование конфигураций** и выберите тип конфигурации для резервного копирования. Действие находится в группе **Управление конфигурациями**.

Ручное восстановление конфигураций

Для восстановления конфигураций устройства до версии, которая резервировалась ранее, запустите действие 87 **Восстановить конфигурацию**. Это действие находится в группе **Управление конфигурациями**.

Это действие сначала предложит выбрать конфигурацию из списка доступных операций резервного копирования.

19.1.23.1.3.1 Планирование резервного копирования конфигураций

Модуль управления конфигурациями включает задачу планировщика 823 **Резервное копирование конфигураций**. Эта задача изначально сконфигурирована для ежедневного резервного копирования конфигураций всех CCM-устройств. Возможно дополнить настройки, чтобы:

- Изменить период резервного копирования
- Уменьшить количество устройств до одного устройства или группы устройств

Создайте несколько копий задачи **Резервное копирование конфигураций**, если Вам нужно:

- Резервировать несколько групп устройств с различными периодами, или
- Резервировать различные типы конфигураций с различными периодами

19.1.23.1.3.2 Автоматическое резервное копирование при изменениях

Конфигурации устройств обычно резервируются по одному в день в соответствии с расписанием 1927. Однако бывает полезно вызывать автоматическое резервное копирование конфигураций сразу после некоторых изменений в них. В этом случае AtomMind Network Manager предоставит поробную историю всех изменений конфигураций.

Это технически возможно для устройств, которые могут отправлять SNMP-ловушки, Syslog-сообщения или другие уведомления для сообщения об изменениях конфигураций. Тревога 779 сервера должна настраиваться для прослушивания уведомлений об изменении конфигураций и выполнения автоматического корректирующего действия 800 **Резервное копирование конфигурации** при получении уведомления.

Тревога триггера резервного копирования "из коробки"

Модуль AtomMind Network Manager Конфигурации и соответствие стандартам включает тревогу **Конфигурация роутера изменена**. Эта тревога реагирует на оповещения об изменении конфигурации, отправленные устройствами Cisco через SNMP-ловушки. Как только мы получаем данное уведомление, система предпринимает попытку резервировать конфигурацию устройства-источника.

Тревогу **Конфигурация роутера изменена** можно легко клонировать и адаптировать к другим типам устройств, поддерживающих асинхронные уведомления об изменении конфигурации.

19.1.23.2 Настройки базовой конфигурации

В определенный момент работы сети команда центра управления сетью может решить, что конфигурация определенного устройства, группы устройств или всех устройств в сети/зоне отлично работает довольно продолжительное время и удовлетворяет всем политикам соответствия. Эти конфигурации обычно резервируются и помечаются как *настройки базовой конфигурации* для указания на то, что они стабильны и безошибочны. Если последующие изменения в конфигурации устройств вводят сеть в нестабильное состояние, сетевые администраторы могут захотеть восстановить конфигурации одного или более сетевых устройств до версий их базовой конфигурации, обходя автоматическое резервирование, которое происходило позже.

Отметка базовой конфигурации и снятие этой отметки

Отдельные конфигурации можно отмечать как базовые конфигурации или снимать с них эти отметки при работе со списком исторических конфигураций. Для получения более подробной информации см. Просмотр и управление конфигурациями 1928.

Групповая базовая конфигурация

Для получения доступа к списку операций базовой конфигурации нескольких устройств:

- Разверните узел **Драйвера/Плагины** () в системном дереве 370.
- Щелкните правой кнопкой на плагине **Управление конфигурациями и соответствием стандартам** ().

Групповые операции базовой конфигурации, доступные в меню, включают следующие операции:

- **Настроить базовую конфигурацию всей сети**. Это действие отмечает самые последние операции резервного копирования конфиграций всех устройств как версий базовой конфигурации.
- **Очистить все базовые конфигурации**. Это действие очищает все отметки базовых конфигураций для исторических конфигураций всех устройств.

19.1.23.3 Управление историей конфигураций

Существует две основных операции управления историей конфигураций:

- Просмотр конфигураций
- Сравнение конфигураций

Просмотр и управление конфигурациями

Для просмотра ранее сохраненных конфигураций устройства запустите действие 87 **Управление конфигурациями**. Это действие расположено в группе **Управление конфигурациями**.

Действие открывает список историй конфигураций, отображающий:

- Дата/время резервирования
- Тип 1926 конфигурации
- Метка конфигурации, т.е. доступное для понимания описание определенного резервирования
- Текст конфигурации
- Флажок, указывающий, что текущая конфигурация является базовой конфигурацией 1927
- Время последнего запроса

Список конфигураций позволяет:

- Удалять в истории выбранные конфигурации
- Редактировать метки конфигураций
- Изменять тексты конфигураций
- Помечать конфигурации как базовые

Сравнение конфигураций

Для сравнения двух ранее по порядку сохраненных конфигураций запустите действие 87 **Сравнить конфигурации**. Действие находится в группе **Управление конфигурациями**.

Сначала действие попросит оператора выбрать конфигурации для сравнения. После этого обе конфигурации откроются в просмотрщике визуальных различий.

19.1.23.4 Отслеживание и оповещение об изменении конфигураций

Каждый раз, когда AtomMind Network Manager определяет, что конфигурация устройства изменилась с момента последнего резервирования, он активирует событие **Конфигурация изменилась**. Это событие сохраняется в базе данных сервера 692.

Отслеживание изменений конфигураций

Для отслеживания истории изменений конфигураций включите фильтр событий 762 **Изменения конфигураций**. Этот фильтр показывает события изменения конфигураций и их основные поля:

- Устройство, чья конфигурация изменилась
- Тип 1926 конфигурации

- Метка конфигурации
- Различие между текущей конфигурацией и последним резервированием

Оповещения об изменениях конфигураций

Пакет AtomMind Network Manager включает тревогу **Конфигурация изменилась**, которая активируется каждый раз при обнаружении изменений конфигураций. Рекомендуем настроить эту тревогу так, чтобы:

- Предупреждать об изменениях конфиграций только главных устройств
- Получать уведомления по e-mail или SMS для изменений конфигураций только главных устройств

19.1.23.5 Управление соответствием стандартам

AtomMind Network Manager может протестировать конфигурации устройств на предмет их соответствия федеральным постановлениям (таким как PCI, HIPAA или SOX) или стандартам корпоративной безопасности. Все нарушения сразу же определяются, а системные операторы получают предупреждения об этом.

Политики соответствия стандартам основаны на правилах. Дистрибутив AtomMind Network Manager включает некоторое количество стандартных правил. Однако существующие правила можно перенастроить и по запросу добавить новые.

Измененные и вновь созданные правила можно легко протестировать на примере конфигурации, хотя нормальное функционирование системы предполагает, что соответствие автоматически проверяется каждый раз при обнаружении изменения конфигурации. Таким образом, статистика нарушений доступна в любое время, равно как и отчеты по расписанию.

Типичные проблемы с конфигурацией можно автоматически устранить путем запуска скрипта конфигурации при получении тревоги Нарушение политики соответствия.

Политики соответствия

Политика соответствия - это набор правил, по которым определяется конфигурация устройства. Если конфигурация соответствует всем правилам в политике, она называется *соответствующая стандартам конфигурация*. Если одно или более правил не выполняются, конфигурация *нарушает политику*.

Администрирование политик соответствия

Этот раздел объясняет, как управлять политиками соответствия. AtomMind Network Manager представляет каждую политику как отдельный контекст 41, которым можно управлять, как любыми другими контекстами (например, тревогами 779 или отчетами 928).

Для администрирования политик соответствия используются два контекста: один из них - это общий контекст Политики соответствия 1959, который служит в качестве контейнера. Другой - это контекст Политка соответствия 1960, который содержит правила одной политики.

Применение политик

После первоначальной конфигурации Политик соответствия необходимо связать выбранные политики с определенными устройствами, чтобы заставить сервер проверять каждое новое резервирование конфигурации на предмет возможных нарушений.

Управление соответствием устройства контролируется при помощи таблицы **Политики соответствия**, на которую можно выйти через диалоговое окно **Редактировать свойства устройства**. Таблица позволяет выбирать множество контекстов Политика соответствия 1960, по которым проверяется устройство.

Проверка соответствия

После привязки нескольких политик соответствия к устройству, соответствие его конфигурации этим политикам проверяется каждый раз при появлении нового резервирования конфигурации 1926.

Для запуска проверки конфигураций устройства в истории на предмет соответствия текущим политикам запустите действие **Проверить соответствие** для любого устройства, чьи конфигурации резервировались хотя бы раз.

Отчет о нарушении политик

Каждое устройство, которое было привязано к политикам соответствия, отправляет отчеты о нарушении политик в его статусе. Для просмотра списка нарушений:

- Щелкните правой кнопкой мыши на устройстве
- Запустите дейтсвие **Просмотр статуса**
- Перейдите во вкладку **Нарушение политик соответствия**

Список нарушений описывает следующее:

- Дата/время проверки нарушений политик
- Дата/время резервного копирования конфигурации
- Тип 1926 конфигурации
- Метка конфигурации
- Описание нарушенной политики
- Описание нарушенного правила политики
- Уровень критичности правила
- Выражение нарушенного правила
- Линейный номер (для правил, работающих в **линейном** режиме)

Список нарушенных политик постоянно сохраняется в базе данных 692 и, таким образом, доступен сразу же после запуска сервера.

Оповещения о нарушении политик

Каждый раз при обнаружении нарушения новой политики AtomMind Network Manager активирует событие Нарушение политики 1934 в контексте Администрирование 1450. Легко настроить тревогу 779, которая будет реагировать на новые события нарушений и, например, отправлять оповещения по e-mail или SMS специалистам по безопасности.

19.1.23.5.1 Конфигурирование политик соответствия

Этот раздел описывает свойства конфигурации политик соответствия.

Все свойства, описанные здесь, доступны через действие Конфигурировать 105 контекста Политики соответствия 1960.

19.1.23.5.1.1 Свойства

Это настройка контролирует основные свойства политики:

Описание поля	Имя поля
Имя. Имя контекста политики. Оно должно удовлетворять контексту соглашений о присвоении имен 42. Имя необходимо для ссылки на эту политику из других частей системы.	name
Описание. Текстовое описание 43 контекста Политика соответствия.	description
Критичность. Уровень критичности 75 по умолчанию для событий о нарушениях политик.	severity

19.1.23.5.1.2 Правила

Эта таблица описывает правила политики. Каждое правило определяется количеством полей:

Описание поля	Имя поля

Правило. Описание правила, например, цель правила.	rule
Критичность. Уровень критичности 75 сообщений о нарушении правил.	severity
Режим. Существует два режима правил политики: • **Файл**. В этом режиме правило **Выражение** применяется ко всему файлу конфигурации. Это режим по умолчанию, подходящий к большинству случаев. • **Строка**. В этом режиме правило **Выражение** применяется к каждой строке файла конфигурации по отдельности.	mode
Выражение. Выражение AtomMind 112, которое определяет, соответствуют ли правилу файл конфигурации и его отдельная строка. Выражение должно возвращать булевое значение: результат `true` означает, что файл конфигурации соответствует правилу, результат `false` означает противоположное. Используйте ссылки 123 `{env/configuration}` для доступа к содержимому конфигурации устройства или его текущей строки (в зависимости от **Режима**).	expression

Среда вычисления 114 выражения правил политики:

Контекст по умолчанию 119	Контекст устройства, чья конфигурация проверяется на предмет нарушений политики.
Таблица данных по умолчанию 120	Отсутствует.
Строка по умолчанию 119	0
Переменные среды 123	Стандартные 123 переменные. Дополнительные переменные:

Имя переменной	Тип переменной	Описание
configuration	Строка	Текст проверяемой конфигурации или строки конфигурации (в зависимости от **Режима** правила).

19.1.23.6 Командные и конфигурационные скрипты

AtomMind Network Manager может выполнять автоматическое реконфигурирование сетевых устройств путем выполнения так называемых конфигурационных скриптов. Эти скрипты решают типичные задачи по изменению конфигураций, такие как:

- Изменение ACL устройства
- Регистрация новой VLAN
- Отключение или перезагрузка устройства
- Изменение таблицы ARP
- Активация IP SLA или NetFlow

Конфигурационные скрипты обычно выполняются сервером через сессии SSH или Telnet. Скрипты не статичны, они могут ссылаться на любые данные устройств или сервера с тем, чтобы каждое устройство получило собственный набор команд.

Движок скриптов поддерживает два различных варианта синтаксиса для написания простых построчных скриптов и реализации пользовательской логики на основе ответов устройств.

Все конфигурационные скрипты можно применить к множеству устройств одновременно. Выполнение может начаться по запросу, расписанию или активироваться тревогой.

Типы скриптов

Существует два типа конфигурационных и командных скриптов:

- **Скрипты ожидания**. Эти скрипты отправляют данные устройству и ожидают определенного результата, то есть, имени. См. скрипты ожидания 613 для более подробного ознакомления.
- **Параметризованные скрипты**. Эти скрипты просят оператора ввести некоторый текст и динамически создают команду(ы), которая отправляется устройству. Обработка результата условного устройства недоступна. См. движок параметризации 1444 для получения более подробной информации о том, как настроить формат ввода и команды устройства.

Управление скриптами

Конфигурационные и командные скрипты управляются через таблицу Конфигурационные скрипты 594, доступную в глобальной конфигурации драйвера устройства Хост сети 593.

Выполнение скриптов

Чтобы выполнить конфигурационный или командный скрипт для определенного устройства, используйте действие 87 **Запустить конфигурационный скрипт**, доступный в группе **Управление конфигурациями**.

Подобно любым другим действиям AtomMind, действие **Запустить конфигурационный скрипт** поддерживает некоторые полезные опции:

- Автоматическое выполнение по тревоге или расписанию
- Пакетное выполнение для многих устройств одновременно

ПАРАМЕТРЫ СКРИПТОВ

До запуска скрипта AtomMind Server сканирует таблицу скриптов и находит все ссылки на поля 118, т.е. ссылки формы `{field}`. Диалоговое окно "Параметры скрипта", которое появляется до начала выполнения скрипта, просит операторов указать значения этих ссылок. Например, если скрипт включает ссылки `{MAC}` и `{IP}`, система требует указать адреса MAC и IP, которые будут использоваться при выполнении.

Если в таблице устройства Параметры резервного копирования 1923 имеются параметры (поля) с совпадающими именами, эти параметры заранее заполняются из этой таблицы, при этом позволяя оператору настроить значения "на ходу" до начала выполнения скрипта.

Имя действия runConfigurationScript

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
ccmScriptName	String	Имя скрипта.
ccmScriptParameters	DataTable	Таблица данных параметров скрипта. См. выше для подробной информации.
controlProtocol	String	Протокол управления: 'Telnet' или 'SSH'. По умолчанию: 'Telnet'.
telnetPort	Integer	Порт Telnet. По умолчанию: 23.
SSHPort	Integer	Порт SSH. По умолчанию: 22.
connectionTimeout	Integer	Таймаут соединения, мс. По умолчанию: 30000.
readTimeout	Integer	Таймаут чтения данных, мс. По умолчанию: 5000.
eolDelimiter	String	Разделитель строки: '\r\n' или '\n'. По умолчанию: '\r\n'. Используйте разделитель '\n' при выборе протокола управления SSH.
sshUsername	String	Имя пользователя для SSH аутентификации.

sshPassword	String	Пароль для SSH аутентификации.

19.1.23.7 Командные и конфигурационные события

Модуль управления конфигурациями и соответствием стандартам регистрирует несколько новых типов событий[73] в контексте Администрирование[1450]:

КОНФИГУРАЦИЯ ИЗМЕНИЛАСЬ

Активируется каждый раз, когда обнаруживаются изменения между новым и предыдущим резервным копированием конфигурации.

Имя события: configurationChanged

Права доступа: Доступны на уровне[486] прав доступа *Администратор*

Период действия: Непостоянный

Записи: 1

Формат[50] записи:

Имя поля	Тип поля	Примечания
device	Строка	Путь контекста устройства[1494], чья конфигурация изменилась.
configurationType	Строка	Тип[1926] измененной конфигурации.
configurationLabel	Строка	Метка конфигурации.
diff	Строка	Разница между новой и старой конфигурацией.

НЕВЫПОЛНЕНИЕ РЕЗЕРВНОГО КОПИРОВАНИЯ КОНФИГУРАЦИИ

Активируется каждый раз, когда резервное копирование конфигурации по какой-то причине не выполняется.

Имя события: configurationBackupFailed

Права доступа: Доступны на уровне[486] прав доступа *Администратор*

Период действия: Непостоянный

Записи: 1

Формат[50] записи:

Имя поля	Тип поля	Примечания
device	Строка	Путь контекста устройства[1494], чья конфигурация изменилась.
configurationType	Строка	Тип[1926] измененной конфигурации.
configurationLastBackupTimestamp	Дата	Дата/время последнего удачного резервного копирования.
error	Строка	Сообщение об ошибке.

НАРУШЕНИЕ ПОЛИТИКИ

Активируется каждый раз, когда обнаруживается новое нарушение политики соответствия.

Имя события: policyViolation

Права доступа: Доступны на уровне 486 прав доступа *Администратор*

Период действия: Непостоянный

Записи: 1

Формат 50 записи:

Имя поля	Тип поля	Примечания
device	Строка	Путь контекста устройства 1494, чья конфигурация изменилась.
configurationType	Строка	Тип 1926 измененной конфигурации.
violations	Таблица данных	Список нарушений политики. См. Отчет о нарушениях политики 1929 дл получения более подробной информации.

19.1.24 Мониторинг IP SLA

AtomMind Network Manager предлагает масштабируемый и легкий в использовании модуль мониторинга сети Cisco IP SLA, который легко интегрируется с другими компонентами по управлению ИТ-инфраструктурой.

Договора о сервисном обслуживании Cisco IOS IP дают клиентам возможность обеспечивать нормальный режим функционирования новых критически важных IP-приложений и IP-сервисов, использующих данные, голос и видео в IP-сети. Cisco имеет традиционный расширенный уровень сервиса мониторинга и продвинутую IP-инфраструктуру, учитывающую особенности используемых IP-приложений при помощи измерения как комплексных приложений, так и тех, что находятся на IP-уровне.

С договорами о сервисном обслуживании Cisco IOS IP SLA пользователи имеют подтверждение гарантии сервиса, увеличивается надежность сети путем подтверждения производительности сети, заранее определяются проблемы в сети и увеличивается возврат инвестиций путем упрощения внедрения новых IP-сервисов. Cisco IOS IP SLA используют активный мониторинг для генерирования трафика в непрерывном, надежном и предсказуемом темпе, таким образом измеряя производительность и состояние сети.

Функции мониторинга IP SLA

- Импорт существующих IP SLA тестов из устройств Cisco
- Создание пакетов новых тестов IP SLA внутри устройств Cisco через центральный сервер мониторинга
- Обновление и удаление тестов через центральный сервер
- Поддержка всех типов тестов IP SLA
- Поддержка полного набора тестовых параметров
- Поддержка мониторинга всех значительных тестовых метрик
- "Коробочные" инструментальные панели инфраструктуры IP SLA
- Заранее настроенные тревоги нарушений IP SLA

19.1.24.1 Концепции мониторинга IP SLA

Функциональность модуля IP SLA включает в себя две необходимые операции:

- Первоначальная настройка 1937 тестов IP SLA на стороне устройств Cisco и на стороне сервера AtomMind Network Manager
- Извлечение, обработка, хранение и визуализация 1952 статуса теста IP SLA и метрик производительности

Модуль IP SLA позволяет управлять и контролировать сотни тестов IP SLA без ручного входа в устройства Cisco, которые непосредственно проводили эти тесты.

Этот модуль использует протокол SNMP для импортирования существующих тестов из устройств Cisco IOS, загружать в них новые тесты и обновлять/удалять тесты. SNMP также используется для чтения статуса IP SLA и метрик производительности.

19.1.24.2 Тестовые типы IP SLA

Модуль IP SLA поддерживает следующие типы тестов:

- **ICMP Echo**. Измеряет время передачи между узлами вашей сети.
- **ICMP Path Echo**. Измеряет время на передачу всем хопам, ведущим к узлу в вашей сети.
- **UDP Echo**. Измеряет время передачи между узлами вашей сети путем отправки данных по протоколу User Datagram Protocol.
- **TCP Connect**. Измеряет качество WAN путем тестирования времени связи между двумя устройствами, используя специальный порт.
- **UDP Jitter**. Измеряет величину задержки между пакетами (джиттер) путем отправки UDP пакетов.
- **ICMP Path Jitter**. Измеряет джиттер от хопа к хопу, потерю пакетов и задержку в измерении статистики в IP-сети.
- **VoIP Call**. Измеряет время ответа вашей сети для настройки звонка VoIP.
- **RTP**. Измеряет джиттер, потерю фрейма, ожидаемую среднюю скорость передачи данных для качества общения (MOS-CQ) и ожидаемую среднюю скорость передачи для качества прослушивания (MOS-LQ), используя DSP голосового шлюза.
- **DNS**. Измеряет разницу времени с момента отправки DNS-запроса и получения ответа.
- **DHCP**. Измеряет время ответа, взятое для обнаружения сервера DHCP, а затем получает от него IP-адрес на временное пользование.
- **FTP**. Измеряет время ответа между устройством Cisco и сервером FTP для извлечения файла.
- **HTTP**. Измеряет различные аспекты процесса загрузки веб-страницы.
- **Custom**. Позволяет определять все настройки тестов IP SLA вручную.

19.1.24.3 Понимание технологии IP SLA

Технология IP SLA способствует тому, что устройства Cisco генерируют и анализируют трафик в узлах вашей сети. Устройство Cisco отправляет моделируемые данные другому устройству Cisco ("Респондер *IP SLA*") или узлу сети (такому как сервер DNS) и анализирует, сколько времени необходимо на получение ответа, что он содержит и т.д.

Ключевая концепция архитектуры IP SLA - это тестирование *IP SLA*. Тестирование можно настроить и задать расписание на выполнение путем программирования устройства Cisco из консоли (что вручную делается системными администраторами) или через протокол SNMP (что делает AtomMind Network Manager). Само же тестирование регулярно проводится устройством Cisco. Результаты теста (т.е. значения измеряемых метрик) выводятся через протокол SNMP. Эти результаты собираются AtomMind Network Managerом, обрабатываются, сохраняются в базе данных и визуализируются по запросу.

AtomMind Network Manager может отслеживать и контролировать тысячи тестирований, выполняемых множеством устройств Cisco.

IP SLA Definitions

Модули мониторинга IP SLA используют следующие определения:

ИСТОЧНИК

Устройство, которое создает и включает пакеты IP SLA в сеть. Источник - это то, откуда инициируется тестирование функционирования IP SLA.

ЦЕЛЬ

Финальное описание пакетов, созданных и отправленных источником.

ТИП

Тип тестирования, производимого в сети.

19.1.24.3.1 Задержка

Задержка VoIP - это метрика разницы во времени между тем, когда начинает говорить один звонящий и его начинает слышать другой. Избыточная задержка в сети может вызывать заметные "пробелы" и потерю синхронизации транслируемого общения, особенно когда VoIP используется с другими типами данных, как в видеоконференции. Если эти "пробелы" становятся довольно большими, звонящие замечают, что они начинают непроизвольно перебивать друг друга.

Тестирование IP SLA измеряет задержку, последовательно применяя четыре различых временых метки к одному пакету тестирования, как здесь:

1. Временная метка T1 применяется к пакету тестирования, когда он покидает исходный маршрутизатор.

2. Временная метка T2 применяется, когда пакет тестирования достигает целевого маршрутизатора.

3. Временная метка T3 применяется, когда пакет тестирования покидает целевой маршрутизатор, чтобы вернутся в исходный.

4. Временная метка T4 применяется, когда пакет тестирования возвращается в исходный маршрутизатор.

Тесты IP SLA затем производят четыре отдельных замера задержек путем вычисления различий между четырьмя метками, как здесь:

Задержки	**Расчет**
Общее время прохождения	T4 – T1
Задержка от исходного к целевому	T2 – T1
Задержка обработки на целевом	T3 – T2
Задержка от целевого к исходному	T4 – T3

19.1.24.3.2 Джиттер

Джиттер является мерой изменчивости задержек в сети, что приводит к потере синхронизации по времени. Во время VoIP звонков пользователи испытывают джиттер, т.е. отвлекающий шум, треск и другое. Для обеспечения приемлемого качества обслуживания джиттер сети должен быть обнаружен, изолирован и устранен.

19.1.24.3.3 Потеря пакетов

Потеря пакетов - это количественная метрика потери информации в данном сетевом соединении. И хотя потеря пакетов неизбежна в любой сетевой среде, целью всегда является определение того, где при передаче теряются пакеты, чтобы можно было минимизировать потерю информации и поддержать высокое качество ваших услуг.

19.1.24.3.4 Средняя оценка качества передачи (MOS)

Средняя оценка качества передачи - это стандартная мера качества звонка, выраженная по шкале возрастающего качества передачи от 1 до 5. Средняя оценка качества передачи вычисляется в соответствии со стандартным алгоритмом Международного Телекоммуникационного Объединения, включающего кодек для вашей сети VoIP и значения задержки, джиттера, потери пакетов и фактора средней оценки качества передачи. Джиттер, задержка и потеря пакетов - это количество переменных, которые измеряются в реальном времени. В общем средняя оценка качества передачи отражает качество звонка, как показано в следующей таблице:

Качество звонка	**Средняя оценка качества передачи**
Очень удовлетворительное	4.3-5.0

Удовлетворительное	4.0-4.3
Часть пользователей удовлетворены	3.6-4.0
Многие пользователи не удовлетворены	3.1-3.6
Почти все пользователи не удовлетворены	2.6-3.1
Не рекомендуемое	1.0-2.6

19.1.24.3.5 Тестовая топология

Тестовая топология - это структура тестов, выбранных во время процесса создания тестирования пакетов 1938. Существует три возможных тестовых топологии:

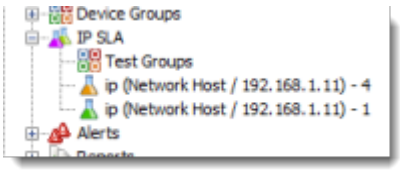
- **Один-к-одному**. Создается один тест. Необходимо выбрать одно исходное устройство Cisco и одно описание (устройство Cisco или необработанный IP/имя хоста).
- **Один-ко-многим**. Создается множество тестов от одного исходного. Необходимо выбрать одно исходное устройство Cisco и множество адресатов (устройства Cisco или необработанные IP/имена хостов).
- **Многие-ко-многим**. Выбирается множество устройств Cisco. Каждое из этих устройств производит тестирование всех других устройств.

19.1.24.4 Конфигурирование тестов IP SLA

Это раздел объясняет, как управлять тестами IP SLA. AtomMind Network Manager представляет каждый тест как отдельный контекст 41, которым можно управлять, как любыми другими контекстами (например, тревогами 779 или отчетами 928). Однако в отличие от других объектов, у каждого теста есть свой "аппаратный пир", т.е. конфигурация устройства. Эти конфигурации должны в общем оставаться синхронизированными 1939 друг с другом.

Администрирование тестов IP SLA

Два контекста используются для администрирования тестов IP SLA: один - это общий контекст Тестов IP SLA 1963, который выступает в качестве контейнера. Другой - это контекст Теста IP SLA 1965, который содержит информацию для одного теста.



19.1.24.4.1 Подготовка к мониторингу IP SLA

Необходимо завершить следующие шаги до создания или импортирования тестов IP SLA с устройства Cisco:

- Нужно создать учетную запись устройства 497, используя драйвер устройства IP Host 593 для устройства Cisco. Это можно сделать либо вручную, либо при помощи обнаружения сети 1811.
- Нужно активировать **CISCO-RTTMON-MIB** в активах 501 SNMP учетной записи этого устройства. Этот MIB-файл обычно описывает OID, относящиеся к управлению и мониторингу тестов IP SLA. Этот MIB обычно активирован по умолчанию.
- Учетная запись устройства должна оставаться в состоянии **Онлайн, Синхронизовано**, обозначенном иконкой . Это нормальное состояние учетной записи устройства.

19.1.24.4.2 Импортирование существующих тестов

Импортирование тестов полезно, когда ваша компания уже проводит IP SLA тесты, которые были раньше сконфигурированы из консоли или с помощью любого другого средства управления сетью. Эта операция загружает определения, свойства и расписание тестов в AtomMind Server и создает учетные записи для этих тестов.

Для импортирования существующих тестов из устройства Cisco:

1. Убедитесь, что учетная запись устройства Cisco, из которого тесты будут импортироваться, полностью готова для операций IP SLA 1937.
2. Щелкните правой кнопкой на иконке узла IP SLA ( ) и выберите **Импорт/Обновление тестов**.
3. Выберите учетную запись устройства Cisco, из которого импортируются тесты.
4. Подождите, пока определения тестов загрузятся из устройства. Отобразится таблица, содержащая источники тестов, цели, числа и типы.
5. Проверьте тесты для импорта и нажмите OK. Заметьте, что если тест с определенным номером уже существует на стороне сервера, он не будет изначально отмечен галочкой. Вы можете выбрать его с целью обновления его серверных свойств.
6. Подождите, пока тесты закончат импортироваться и учетные записи тестов создадутся на сервере.
7. Когда процесс импорта закончится, отобразится статус отчета. Этот отчет будет содержать список импортированных/обновленных тестов и ошибки импорта/обновления тестов.

19.1.24.4.3 Обновление тестов

Процесс обновления пакетов тестов полезен, когда конфигурация более одного теста изменилась в устройстве Cisco, но AtomMind Network Manager уже имеет учетные записи дял этих тестов. Эта операция извлекает все настройки теста из устройства и обновляет конфигурацию учетной записи теста на стороне сервера.

Для обновления одного теста используйте операцию синхронизация теста 1939.

Для обновления серверной конфигурации тестов следуйте по шагам, описанным в статье импортирование существующих тестов 1937. Все тесты, которые уже имееют учетную запись пира, изначально не будут выбраны в диалоговом окне выбора тестов. Выберите их для обновления свойств учетной записи сервера.

19.1.24.4.4 Создание новых тестов

AtomMind SCADA/HMI может создавать тесты IP SLA напрямую в устройстве Cisco. Системным администраторам не нужно входить в устройство из консоли и выполнять любые дополнительные настройки вручную.

Для создания новых тестов в устройстве Cisco:

1. Убедитесь, что учетная запись устройства Cisco, из которого будут импортироваться тесты, полностью готова для проведения операций IP SLA 1937.
2. Щелкните правой кнопкой мыши на иконке узла IP SLA ( ) и выберите **Создать тесты**.
3. Выберите типы тестов 1935 для создания.
4. Выберите топологию тестов 1937.
5. Выберите одну или больше исходных устройств Cisco в соответствии с выбранной топологией тестов.
6. Выберите одну или более целей в соответствии с выбранной топологией. Целью теста может быть **существующая учетная запись устройства Cisco** или **произвольный IP-адрес/имя хоста**.
7. Установите свойства тестов.
8. Установите расписание только что созданных тестов.
9. Подождите, пока сконфигурируются новые тесты на выбранных исходных устройствах и создадутся учетные записи устройств со стороны сервера.
10. После окончания процесса создания будет отображаться отчет о состоянии. Этот отчет будет содержать список созданных тестов и ошибки ввода/обновления тестов.

19.1.24.4.5 Удаление тестов

Для удаления тестов IP SLA:

- Щелкните правой кнопкой мыши на иконке теста ( ) и выберите **Устройство**
- Подтвердите удаление учетной записи теста с сервера
- Отдельно подтвердите или отклоните удаление теста из устройства Cisco

19.1.24.4.6 Синхронизирование тестов

Синхронизация теста - это:

- Процесс написания изменений в устройстве Cisco, сделанных в конфигурации учетной записи теста на стороне сервера, или
- Процесс прочтения изменений, сделанных в устройстве Cisco, и их применение к учетной записи теста на стороне сервера

Для синхронизации свойств теста между сервером AtomMind Network Manager и устройством Cisco:

- Щелкните правой кнопкой мыши по иконке узла теста и () выберите **Синхронизировать**.
- Выберите направление синхронизации (**Из устройства** или **В Устройство**).
- Дождитесь окончания процесса синхронизации.
- Как только процесс синхронизации будет завершен, появится отчет о состоянии. Этот отчет будет содержать список синхронизированных тестов и ошибок синхронизации.

19.1.24.4.7 Тестовые свойства IP SLA

Этот раздел описывает свойства теста IP SLA. Более подробно об этих свойствах можно узнать в ссылках Cisco IP SLA и файле CISCO-RTTMON-MIB.

Основные свойства

Свойство	Описание
Владелец	Строка, определяющее лицо, создавшее этот ряд в таблице.
Тег	Строка, используемая управляющим приложением для определения цели RTT. Эта строка вставляется в уведомления о ловушках, но иного значения для агента не имеет.
Тип	Тип операции IP SLA, подлежащей исполнению.
Порог	Административный лимит порога. Если время выполнения RTT превышает этот лимит и если выполняются условия, указанные в **Типе порога** (таблица **Реакции**) или **Фильтре** (таблица **История**), генерится событие о нарушении порога.
Частота	Определяет временной период между началом каждой операции RTT. Этот объект не может быть настроен на значение с меньшей продолжительностью, чем **Время ожидания**. Когда **Тип** теста - это Path Echo, предложенное значение для этого объекта больше, чем **Время ожидания**, умноженное на максимальное число ожидаемых хопов до цели. Когда **Тип** теста - это DHCP, минимальное разрешенное значение для этого объекта - это 10 секунд.
Время ожидания	Определяет время ожидания завершения операции RTT. Значение этого объекта нельзя установить на значение, которое определило бы продолжительность превышающей частоты. Для протоколов, ориентированных на подключение, оно может вызвать закрытие соединения зондом. После закрытия предполагается, что будет выполняться восстановление подключения. Для предотвращения нежелательного закрытия подключения установите реальное время ожидания для этого значения.
Протокол	Определяет протокол, используемый для выполнения операции RTT. Когда этот протокол не поддерживает данный тип, возвращается ошибка 'badValue'.
Целевой адрес	Строка, которая определяет целевой адрес.
Исходный адрес	Строка, которая определяет исходный IP-адрес. Этот объект применим ко всем зондам, кроме dns, dlsw и sna.

Расширенные свойства

Свойство	Описание
Проверка данных	Когда установлено на true, данные результата в каждой операции RTT сравнивается с ожидаемыми данными, что включает информацию заголовка (по возможности) и точный размер пакета. Любое несовпадение записывается в объект `rttMonStatsCollectVerifyErrors`. Некоторые типы тестов могут не поддерживать эту опцию. Когда тип не поддерживает эту опцию, агент переведет это свойство в false. Проверка этого перевода находится в зоне ответственности приложений по управлению. Это свойство применимо только для протоколов SNA.
Энергозависимая память	Когда установлено на true, эта запись будет отображаться в команде 'показать текущие' и может сохраняться в энергозависимой памяти.
Размер запроса	Этот объект представляет количество октетов, помещаемых в ARR Data сообщения-запроса при использовании протокола SNA. Для протоколов с RTT запросами/ответами без ARR, это значение представляет собственный размер информационного наполнения. Сверх того, Заголовок ARR не включается в это значение. Для зондов echo общий размер пакета = (IP-заголовок(20) + ICMP-заголовок(8) + 8 (внутренних именных меток) + размер запроса). Для запросов по умолчанию echo и pathEcho размер составляет 28. Для зонда udp размер запроса по умолчанию равен 16, а для зонда jitter - 32. Для зондов dlsw размер запросов по умолчанию равен 0. Минимальный размер запроса для echo и pathEcho - 28 байт, для udp - 4, а для jitter 16. Для зондов udp и jitter максимальный размер запроса - 1500.
Размер ответа	Этот объект представляет количество октетов, помещаемых в ARR Data сообщения-ответа. Значение передается на Сервер RTT Echo через поле Заголовок ARR. Для RTT запросов/ответов без ARR (например, ipIcmpecho) это значение устанавливается агентом, чтобы соответствовать объекту Размер запроса, когда поддерживается родное информационное наполнение. Сверх того, Заголовок ARR не включается в это значение. Данный объект поддерживается только протоколом SNA.
Имя VPN	Это поле необходимо для определения имени VPN, в котором будет использоваться операция RTT. Для стандартных операций RTT это поле не конфигурируется. Агент будет использовать это поле для определения Таблицы маршрутизации VPN для этой операции.
Значение типа синхронизации часов	Этот объект идентифицирует, определено ли значение в односторонней NTP синхронизации как абсолютное значение или процент.

Свойства тестов Echo, Path Echo

Свойство	Описание
Исходный порт	Объект представляет номер порта источника. Если этот объект не определен, приложение получит порт, заданный системой. Объект применим ко всем зондам, кроме dns, dlsw и sna.
Тип IP сервиса	Объект представялет тип октета сервиса в IP заголовке. Этот объект не применим к dhcp и dns.
Активированный LSR	При активации этго объекта приложение вычисляет время ответа для определенного пути. Этот объект применим только для зонда echo.
Тип LSP FEC	Тип цели FEC для RTT 'echo' и операции 'pathEcho' на базе **Протокола** 'mplsLspPingAppl'.

Селектор LSP	Строка, определяющая валидный адрес 127/8. Этот адрес имеет форму 127.x.y.z. Этот адрес не используется для маршрутизации пакета MPLS echo в место назначения, но используется для баланса загрузки в случаях, когда адрес назначения полезной информации IP используется для баланса загрузки.
Режим ответа LSP	Этот объект определяет режим ответа для запросов LSP Echo.
LSP TTL	Этот объект предоставляет настройку TTL для пакетов запросов MPLS echo. Для операции пинга он представляет значение TTL, которое должно быть установлено в пакет запросов echo. Для операции трассировки он представляет максимальное значение ttl, которое можно установить в пакетах запросов echo, начиная с TTL=1. Для 'echo' на базе mplsLspPingAppl TTL по умолчанию будет установлено 255, а для 'pathEcho' на базе mplsLspPingAppl значение по умолчанию будет равно 30. Примечание: этот объект не может быть установлен на значение равное 0. Значение по умолчанию 0 означает, что значения по умолчанию TTL должны использоваться для 'echo' и 'pathEcho' на базе 'mplsLspPingAppl'.
Значение LSP EXP	Этот объект представляет значение EXP, которое нужно установить как приоритетное в заголовке IP запроса MPLS echo.
Значение ответа LSP DSCP	Этот объект определяет значение DSCP, которое нужно установить в заголовке IP ответного пакета LSP. Значение этого объекта составит где-то между 0 и 63 для значений кодовых точек DiffServ. Примечание: Этот объект нельзя установить на значение 255. Это значение по умолчанию определяет, что DSCP не установлен для этой записи.
Нулевая метка LSP	Этот объект определяет, нужно ли добавлять метку explicit-null к запросам LSP echo, которые отправляются при выполнении операции RTT.

Свойства теста UDP Echo

Свойство	Описание
Целевой порт	Этот объект представляет номер целевого порта. Данный объект применим к udpEcho, tcpConnect и зондам джиттера.
Исходный порт	Этот объект представляет номер порта источника. Если этот объект не определен, приложение получит порт, определенный системой. Данный объект применим ко всем зондам, кроме dns, dlsw и sna.
Контрольное сообщение	Если этот объект активирован, тогда приложение RTR отправит контрольные сообщения респондеру, находящемуся на целевом маршрутизаторе, чтобы тот ответил на пакеты запроса данных, отправляемых маршрутизатором-источником. Данный объект не применим к зондам echo, pathEcho, dns и http.
Тип IP сервиса	Этот объект представляет тип октета сервиса в заголовке IP. Данный объект не применим к dhcp и dns.

Свойства теста TCP Connect

Свойство	Описание
Целевой порт	Этот объект предсталяет номер целевого порта. Данный объект применим к зондам udpEcho, tcpConnect и jitter.

Исходный порт	Этот объект предсталяет номер исходного порта. Если этот объект не определен, у приложения появится порт, определенный системой. Данный объект применим ко всем зондам, кроме dns, dlsw и sna.
Контрольное сообщение	Если этот объект включен, тогда приложение RTR отправит контрольное сообщение респондеру, находящемуся на целевом маршрутизаторе, ответить на пакеты запроса данных, отправляемых исходным маршрутизатором. Данный объект не применим к зондам echo, pathEcho, dns и http.
Тип IP сервиса	Этот объект представляет тип октета сервиса в IP-заголовке. Данный объект не применим к dhcp и dns.

Свойства теста HTTP

Свойство	Описание
Исходный порт	Этот объект предсталяет номер исходного порта. Если этот объект не определен, у приложения появится порт, определенный системой. Данный объект применим ко всем зондам, кроме dns, dlsw и sna.
Тип IP сервиса	Этот объект представляет тип октета сервиса в IP-заголовке. Данный объект не применим к dhcp и dns.
Тип операции	Код, который представляет определенный тип операций HTTP или FTP. Этот объект применим только к зондам http и ftp.
Версия HTTP Version	Строка, которая определяет номер версии HTTP Server. Синтаксис версии строки <major number>.<minor number> Примером будет 1.0, 1.1 и т.д. Этот объект применим только к зонду http.
URL	Строка, представляющая URL, к которой должен подключаться зонд HTTP. Этот объект применим только к зонду http.
Кэш	Если этот объект - false, значит HTTP-запрос не должен загружать кэшированные страницы. Это значит, что запрос нужно направить исходному серверу. Данный объект применим только к зонду http.
Прокси	Эта строка представляет информацию о прокси-сервере. Этот объект применим только к зонду http.
Строка запроса 1	Эта строка хранит контент необработанного запроса HTTP. Если запрос не соответствует Строке 1, тогда он должен быть разделен и вставлен в Строку 1 через 5. Эта строка необработанных опциональных данных DHCP. Необработанные опциональные данные DHCP должны быть в HEX. Если определено нечетное количество символов, то 0 будет добавлен к концу строки. Разрешена только DHCP опция 82 (десятичное). Вот пример допустимой строки: 5208010610005A6F1234 Только Строка запроса 1 используется для dhcp, Строки 1 через 5 не используются. Данный объект применяется только для зондов типа http и dhcp.
Строка запроса 2	Эта строка хранит содержимое необработанного запроса HTTP. Строки запроса 1-5 соединяются, формируя необработанный запрос HTTP, используемый в работе RTT. Этот объект применяется только для зонда http.
Строка запроса 3	Эта строка хранит содержимое необработанного запроса HTTP. Строки запроса 1-5 соединяются, формируя необработанный запрос HTTP, используемый в работе RTT. Этот объект применяется только для зонда http.
Строка запроса 4	Эта строка хранит содержимое необработанного запроса HTTP. Строки запроса 1-5 соединяются, формируя необработанный запрос HTTP, используемый в работе RTT. Этот объект применяется только для зонда http.

Строка запроса 5	Эта строка хранит содержимое необработанного запроса HTTP. Строки запроса 1-5 соединяются, формируя необработанный запрос HTTP, используемый в работе RTT. Этот объект применяется только для зонда http.

Свойства теста FTP

Свойство	Описание
Исходный порт	Этот объект предсталяет номер исходного порта. Если этот объект не определен, у приложения появится порт, определенный системой. Данный объект применим ко всем зондам, кроме dns, dlsw и sna.
Контрольное сообщение	Если активируется этот объект, тогда приложение RTR отправляет контрольные сообщения респондеру, находящемуся на целевом маршрутизаторе, для ответа на пакеты запроса данных, отправляемых исходным маршрутизатором. Этот объект не применим к зондам echo, pathEcho, dns и http.
Тип IP сервиса (TOS)	Этот объект представляет тип октета сервиса в IP-заголовке. Данный объект не применим к dhcp и dns.
Тип операции	Код, представляющий определенный тип операции HTTP или FTP. Этот объект применим только к зондам http и ftp.
Режим	Код, представляющий особый тип операции RTT. Этот объект применим только к зонду ftp.

Свойства теста DNS

Свойство	Описание
Строка целевого адреса	Строка, определяющая адрес цели. Эта строка может быть в формате IP-адреса или имени хоста. Этот объект применим только к зонду dns.
Сервер имен	Строка, определяющая ip-адрес сервера имен. Этот объект применим только к зонду dns.

Свойства теста джиттера

Свойство	Описание
Целевой порт	Этот объект предсталяет номер целевого порта. Данный объект применим к зондам udpEcho, tcpConnect и jitter.
Исходный порт	Этот объект предсталяет номер исходного порта. Если этот объект не определен, у приложения появится порт, определенный системой. Данный объект применим ко всем зондам, кроме dns, dlsw и sna.
Контрольное сообщение	Если активируется этот объект, тогда приложение RTR отправляет контрольные сообщения респондеру, находящемуся на целевом маршрутизаторе, для ответа на пакеты запроса данных, отправляемых исходным маршрутизатором. Этот объект не применим к зондам echo, pathEcho, dns и http.
Тип IP сервиса (TOS)	Этот объект представляет тип октета сервиса в IP-заголовке. Данный объект не применим к dhcp и dns.
Интервал	Значение представляет задержку между пакетами и измеряется в миллисекундах. Это значение в данный момент используется зондом jitter. Данный объект применим только для зонда jitter.

Количество пакетов	Значение представляет количество пакетов для передачи. Это значение в данный момент используется зондом jitter. Данный объект применим только к зонду jitter.
Тип кодека	Определяет тип кодека для использования с зондом jitter. Это применимо только для зонда jitter. Если конфигурируется тип "кодек", нельзя настроить следующие параметры: • Размер запроса • Интервал • Количество пакетов
Интервал кодека	Это поле представляет задержку между пакетами в миллисекундах. Данный объект применим только к зонду jitter, использующему тип кодек.
Полезные данные кодека	Объект представляет количество октетов, необходимых для помещения в порцию данных. Это значение используется только для зонда jitter с типом кодек.
Количество пакетов кодека	Значение представляет количество пакетов, необходимых для передачи. Это значение используется только для зонда jitter с типом кодек.
Фактор ICPIF	Преимущественный фактор зависит от типа доступа и того, как используется сервис. • Стандартный проводной доступ - 0 • Мобильный доступ в пределах здания - 5 • Мобильный доступ в пределах географической площади - 10 • Доступ в труднодоступных местах - 20 Это будет применяться при подсчете значений ICPIF. Действительно только для jitter при подсчете значения ICPIF.
Точность	Этот объект определяет точность статистики, которую нужно подсчитать в миллисекундах - точность статистики будет в микросекундах. Значение можно установить только для операции jitter.
Приоритет пакетов зонда	Этот объект определяет приоритетность, присваиваемую пакетам зонда. Это значение можно установить только для операции jitter.
Общее количество ошибок синхронизации часов, микросекунды	Этот объект определяет точное количество ошибок синхронизации часов на источнитке и респондере, которые считаются приемлемыми для одностороннего измерения, когда NTP используется в качестве механизма синхронизации часов. Общее количество ошибок синхронизации часов - это сумма начальных номеров NTP на исходнике и респондере. Значение определяется в микросекундах. Это значение может быть установлено только для операции jitter с точностью до микросекунд.
Общее количество ошибок синхронизации часов, %	Этот объект определяет точное количество ошибок синхронизации часов на источнитке и респондере, которые считаются приемлемыми для одностороннего измерения, когда NTP используется в качестве механизма синхронизации часов. Общее количество ошибок синхронизации часов - это сумма начальных номеров NTP на исходнике и респондере. Это значение выражается в процентах актуального измеряемого одностороннего времени задержки. Это значение может быть установлено только для операции jitter с точностью до микросекунд.

Свойства теста ICMP jitter

Свойство	Описание
Целевой порт	Этот объект предсталяет номер целевого порта. Данный объект применим к зондам udpEcho, tcpConnect и jitter.

Исходный порт	Этот объект предсталяет номер исходного порта. Если этот объект не определен, у приложения появится порт, определенный системой. Данный объект применим ко всем зондам, кроме dns, dlsw и sna.
Контрольное сообщение	Если активируется этот объект, тогда приложение RTR отправляет контрольные сообщения респондеру, находящемуся на целевом маршрутизаторе, для ответа на пакеты запроса данных, отправляемых исходным маршрутизатором. Этот объект не применим к зондам echo, pathEcho, dns и http.
Тип IP сервиса (TOS)	Этот объект представляет тип октета сервиса в IP-заголовке. Данный объект не применим к dhcp и dns.
Интервал	Значение представляет задержку между пакетами и измеряется в миллисекундах. Это значение в данный момент используется зондом jitter. Данный объект применим только для зонда jitter.
Количество пакетов	Значение представляет количество пакетов для передачи. Это значение в данный момент используется зондом jitter. Данный объект применим только к зонду jitter.
Тип кодека	Определяет тип кодека для использования с зондом jitter. Это применимо только для зонда jitter. Если конфигурируется тип "кодек", нельзя настроить следующие параметры: • Размер запроса • Интервал • Количество пакетов
Интервал кодека	Это поле представляет задержку между пакетами в миллисекундах. Этот объект применим только к зонду jitter, использующему тип кодек.
Полезные данные кодека	Объект представляет количество октетов, необходимых для помещения в порцию Данных. Это значение используется только для зонда jitter с типом кодек.
Количество пакетов кодека	Это значение представляет количество пакетов, необходимых для передачи. Данное значение применимо только для зонда jitter с типом кодек.
Фактор ICPIF	Преимущественный фактор зависит от типа доступа и того, как используется сервис. • Стандартный проводной доступ - 0 • Мобильный доступ в пределах здания - 5 • Мобильный доступ в пределах географической площади - 10 • Доступ в труднодоступных местах - 20 Это будет применяться при подсчете значений ICPIF. Действительно только для jitter при подсчете значения ICPIF.
Точность	Этот объект определяет точность статистики, которую нужно подсчитать в миллисекундах - точность статистики будет в микросекундах. Значение можно установить только для операции jitter.
Приоритет пакетов зонда	Этот объект определяет приоритетность, присваиваемую пакетам зонда. Это значение можно установить только для операции jitter.
Общее количество ошибок синхронизации часов, микросекунды	Этот объект определяет точное количество ошибок синхронизации часов на источнитке и респондере, которые считаются приемлемыми для одностороннего измерения, когда NTP используется в качестве механизма синхронизации часов. Общее количество ошибок синхронизации часов - это сумма начальных номеров NTP на исходнике и респондере. Значение определяется в микросекундах. Это значение может быть установлено только для операции jitter с точностью до микросекунд.
Общее количество ошибок	Этот объект определяет точное количество ошибок синхронизации часов на источнитке и респондере, которые считаются приемлемыми для одностороннего измерения, когда NTP используется в качестве механизма синхронизации часов. Общее количество

синхронизации часов, %	ошибок синхронизации часов - это сумма начальных номеров NTP на исходнике и респондере. Это значение выражается в процентах актуального измеряемого одностороннего времени задержки. Это значение может быть установлено только для операции jitter с точностью до микросекунд.

Свойства теста DLSW

Свойство	Описание
Контрольное сообщение	Если активируется этот объект, тогда приложение RTR отправляет контрольные сообщения респондеру, находящемуся на целевом маршрутизаторе, для ответа на пакеты запроса данных, отправляемых исходным маршрутизатором. Этот объект не применим к зондам echo, pathEcho, dns и http.
Тип IP сервиса	Этот объект представляет тип октета сервиса в IP-заголовке. Данный объект не применим к dhcp и dns.

Свойства теста DHCP

Свойство	Описание
Исходный порт	Этот объект предсталяет номер исходного порта. Если этот объект не определен, у приложения появится порт, определенный системой. Данный объект применим ко всем зондам, кроме dns, dlsw и sna.
Контрольное сообщение	Если активируется этот объект, тогда приложение RTR отправляет контрольные сообщения респондеру, находящемуся на целевом маршрутизаторе, для ответа на пакеты запроса данных, отправляемых исходным маршрутизатором. Этот объект не применим к зондам echo, pathEcho, dns и http.
Строка запроса 1	Эта строка хранит контент необработанного запроса HTTP. Если запрос не соответствует Строке 1, тогда он должен быть разделен и вставлен в Строку 1 через 5. Эта строка необработанных опциональных данных DHCP. Необработанные опциональные данные DHCP должны быть в HEX. Если определено нечетное количество символов, то 0 будет добавлен к концу строки. Разрешена только DHCP опция 82 (десятичное). Вот пример допустимой строки: 5208010610005A6F1234 Только Строка запроса 1 используется для dhcp, Строки 1 через 5 не используются. Данный объект применяется только для зондов типа http и dhcp.

Свойства теста VoIP звонков

Свойство	Описание
Исходный порт	Этот объект предсталяет номер исходного порта. Если этот объект не определен, у приложения появится порт, определенный системой. Данный объект применим ко всем зондам, кроме dns, dlsw и sna.
Контрольное сообщение	Если активируется этот объект, тогда приложение RTR отправляет контрольные сообщения респондеру, находящемуся на целевом маршрутизаторе, для ответа на пакеты запроса данных, отправляемых исходным маршрутизатором. Этот объект не применим к зондам echo, pathEcho, dns и http.

Тип IP сервиса	Этот объект представляет тип октета сервиса в IP-заголовке. Данный объект не применим к dhcp и dns.
Набранный номер	Эта строка хранит задержку набранного номера после его набора. Данный объект применим только к зонду задержки после набора voip. Номер будет таким же, какой набрал пользователь. В нем будет номер с кодом страны, плюс номер E.164. Максимальная длина - 24 цифр. Возможны только цифры 0-9.
Точка определения	Код, представляющий точку обнаружения задержки после набора номера. Этот объект применим только к зонду задержки после набора номера SAA.
Задержка регистрации VoIP GK	Булевое значение, представляющее задержку регистрации VoIP GK. Этот объект применим только к зонду задержки регистрации SAA GK.

Свойства теста RTP

Свойство	Описание
Исходный порт	Этот объект предсталяет номер исходного порта. Если этот объект не определен, у приложения появится порт, определенный системой. Данный объект применим ко всем зондам, кроме dns, dlsw и sna.
Контрольное сообщение	Если активируется этот объект, тогда приложение RTR отправляет контрольные сообщения респондеру, находящемуся на целевом маршрутизаторе, для ответа на пакеты запроса данных, отправляемых исходным маршрутизатором. Этот объект не применим к зондам echo, pathEcho, dns и http.
Тип IP сервиса	Этот объект представляет тип октета сервиса в IP-заголовке. Данный объект не применим к dhcp и dns.
Исходный голосовой порт	Строка определяет голосовой порт на исходном шлюзе. Объект применим только для порта RTP.
Продолжительность звонка	Длительность RTP сессии. Объект применим только для порта RTP.

19.1.24.4.8 Расписание тестирования IP SLA

Этот раздел описывает свойства расписания тестов IP SLA. Более подробно о свойствах можно почитать в ссылках Cisco IP SLA и файле CISCO-RTTMON-MIB.

Эта таблица содержит расширенные свойства IP SLA. В большинстве случаев они не должны редактироваться вручную.

Свойства

Свойство	Описание
Время жизни, секунды `(rttMonScheduleAdminRttLife)`	Определяет время жизни нового теста. Значение **2147483647** имеет особый смысл. Когда этот объект установлен на **2147483647**, объект **Время жизни** не уменьшается. Таким образом, время жизни никогда не заканчивается.
Время начала, секунды	Определяет время активации нового теста.

(rttMonScheduleAdminRttStartTime)	Когда агент имеет возможность определить дату и время, он должен сохранять этот объект в виде DateAndTime. Это позволяет агенту полностью переустановиться (перезапуститься) и при этом активировать концептуальные ряды RTT в определенное время. Если агент не может сохранить дату и время, он переустанавливается, а все записи берут одно из специальных значений, определенных ниже.
Истечение срока концептуального ряда (rttMonScheduleAdminConceptRowAgeout)	Определяет, когда автоматически удаляется неактивный тест. Когда данный концептуальный контрольный ряд RTT входит в 'активную' стадию, таймер будет переустановлен и переведен в состояние ожидания. Когда данный концептуальный контрольный ряд RTT входит в любую стадию, кроме 'активной', таймер перезапускается. Когда данное значение установлено на ноль, эта запись никогда не устаревает.
Рекурсия (rttMonScheduleAdminRttRecurring)	Когда установлено на true, данное значение будет запланировано на автозапуск с определенной продолжительностью, равной сконфигурированному времени жизни, и в то же самое время ежедневно. Это значение нельзя установить на true: (a) если объект **Время жизни** имеет значение большее или равное 86400 секундам. (b) если сумма значений **Время жизни** и **Истечение срока концептуального ряда** меньше или равно **86400** секундам.

19.1.24.4.9 Тестовые реакции IP SLA

Этот раздел описывает свойства реакций тестов IP SLA. Более подробно о свойствах можно почитать в ссылках Cisco IP SLA и файле CISCO-RTTMON-MIB.

Эта таблица содержит расширенные свойства IP SLA. В большинстве случаев они не должны редактироваться вручную.

Свойства

Свойство	Описание
Соединение активировано (rttMonReactAdminConnectionEnable)	Если true, генерируется реакция, когда операция RTT в **Целевом адресе** (типа эхо) заставляет тест ввести состояние **Соединение потеряно**. Таким образом, соединения с промежуточными хопами не поменяют это значение.
Истечение срока ожидания активировано (rttMonReactAdminTimeoutEnable)	Если true, генерируется реакция, когда операция RTT заставляет тест ввести состояние **Истечение срока ожидания**. Когда истекает срок ожидания **Типа** теста 'pathEcho' на промежуточном хопе, это не вызовет состояние **Истечение срока ожидания**.
Тип порога (rttMonReactAdminThresholdType)	Этот объект определяет условия, при которых тест вводит состояние нарушения порога (например, **Предупреждение**, **Критичное** или **Сбой)**: ПРИМЕЧАНИЕ: Когда **Тип** теста - 'pathEcho', значение этого объекта и все значения ассоциированных объектов действительны только когда RTT операции 'echo' находятся в адресе объекта rttMonEchoAdminTargetAddress. Таким образом, операции 'pathEcho' в промежуточных хопах не приведут к изменению данного объекта. • **никогда** - нарушение порога никогда не происходит • **немедленно** - нарушение порога происходит, когда время завершения операции превышает значение **Порога**; с другой стороны, нарушение порога деактивируется, когда время завершения операции ниже **Лимита порога** • **последовательно** - нарушение порога происходит, когда время завершения операции превышает **Порог** в **Количестве порогов, X** последовательных операций RTT; с другой стороны, нарушение порога деактивируется, когда время завершения

	операции ниже **Лимита порогов** для того же количества последовательных операций • **xOfy** - нарушение порога происходит, когда x (как определено **Количеством порогов, X**) из последнего времени завершения операций y (как определено **Количеством порогов, Y**) превышает **Порог**; с другой стороны, оно устанавливается на false, когда x из последнего времени завершения операций y находится ниже **Лимита порога** ПРИМЕЧАНИЕ: Когда x > y, зонд никогда не сгенерирует реакцию. • **среднее** - нарушение порога происходит, когда среднее значение предыдущего времени завершения операций **Количество порогов** превышает **Порог**; с другой стороны, оно устанавливается на false, когда действующее среднее значение ниже **Лимита порога** Если это значение меняется управляющей станцией, нарушение порога переустанавливается, но реакция не генерируется, если было активно нарушение порога.
Лимит порога (rttMonReactAdminThresholdFalling)	Этот объект определяет лимит порога. Если время операции RTT ниже данного лимита и если выполняются условия, определенные в **Типе порога**, генерируется порог.
Количество порогов, X (rttMonReactAdminThresholdCount)	Данный объект определяет значение 'x' условия xOfy, определенного в **Типе порога**.
Количество порогов, Y (rttMonReactAdminThresholdCount2)	Данный объект определяет значение 'y' условия xOfy, определенного в **Типе порога**.
Тип действия (rttMonReactAdminActionType)	Определяет, какой(ие) необходимо генерировать тип(ы) реакции(ий) (при их наличии), если операция нарушает одно из наблюдаемых условий: • **none** - реакция не генерируется • **trapOnly** - генерируется ловушка • **nmvtOnly** - генерируется SNA NMVT • **triggerOnly** - инициируются все действия триггера, определенные дял этой записи • **trapAndNmvt** - генерируются как ловушка, так и SNA NMVT • **trapAndTrigger** - инициируются как ловушка, так и действия триггера • **nmvtAndTrigger** - инициирутся как NMVT, так и все действия триггера • **trapNmvtAndTrigger** - инициирутся NMVT, ловушка и все действия триггера
Ошибка активирована (rttMonReactAdminVerifyErrorEnable)	Если true, генерируется реакция, когда происходит ошибка операции RTT.

19.1.24.4.10 Тестовая статистика IP SLA

Этот раздел описывает свойства статистики тестов IP SLA. Более подробно о свойствах можно почитать в ссылках Cisco IP SLA и файле CISCO-RTTMON-MIB.

Эта таблица содержит расширенные свойства IP SLA. В большинстве случаев они не должны редактироваться вручную.

Свойства

Свойство	Описание
Количество групп (rttMonStatisticsAdminNumHourGroups)	Максимальное количество групп путей для записи. В частности, это количество групп по часам для сохранения до наведения. Это значение не рекомендуется, потому что группа закроется и будет сразу же удалена до того момента, как станция управления сетью получит возможность извлечь статистику. Значение, используемое в таблице статистики (rttMonStatsCaptureTable) для уникального определения этой группы, является таблицей статистики, использующей индекс (rttMonStatsCaptureStartTimeIndex). Зонды HTTP и Jitter сохраняют только два часа данных. Когда данный объект установлен на значение ноль, весь сбор табличных данных статистики будет остановен.
Количество путей (rttMonStatisticsAdminNumPaths)	Когда **Тип** теста - это 'pathEcho', он представляет собой максимальное количество путей статистики для записи на каждую часовую группу. Это значение напрямую представляет путь к цели. Для всех других **Типов** это значение будет переведено агентом на один. ПРИМЕЧАНИЕ: Для 'pathEcho' источник к целевому пути будет создан для удержания всех ошибок, случающихся, когда определенный путь или связь не были найдены/установлены. Таким образом, рекомендуется установить это значение на больше одного. Поскольку этот индекс не наводится, сохраняться будет только **Количество путей**.
Количество хопов (rttMonStatisticsAdminNumHops)	Когда **Тип** теста - это 'pathEcho', он представляет собой максимальное количество хопов статистики для записи на каждую группу путей. Это значение напрямую представляет количество хопов на пути к цели, таким образом, мы можем поддержать только 30 хопов. Для всех других **Типов** это значение будет переведено агентом на один. Поскольку этот индекс не наводится, сохраняться будет только **Количество хопов**. Данный объект применим только к зондам 'pathEcho'.
Количество бакетов распределения (rttMonStatisticsAdminNumDistBuckets)	Максимальное количество аккумулирования статистических бакетов распределения. Поскольку этот индекс не наводится, сохраняться будет только первое **Количество бакетов распределения**. Последний бакет будет содержать все записи с начальной точки его интервала распределения до бесконечности. Данный объект не применим к зондам http и jitter.
Интервал бакетов распределения (rttMonStatisticsAdminDistInterval)	Интервал статистических бакетов распределения. Пример бакета распределения: **Количество бакетов распределения** = 5 бакетов **Интервал бакетов распределения** = 10 миллисекунд | Бакет 1 | Бакет 2 | Бакет 3 | Бакет 4 | Бакет 5 | | 0-9 мс | 10-19 мс | 20-29 мс | 30-39 мс | 40-Бесконечность мс | Случайный пример: **Количество бакетов распределения** = 1 бакет **Интервал бакетов распределения** = 10 миллисекунд | Бакет 1 | | 0-Бесконечность мс | Таким образом, этот случайный пример показывает, что значение **Интервала бакетов распределения** не применимо, когда **Количество бакетов распределния** равно одному. Этот объект не применим к зондам http и jitter.

19.1.24.4.11 Тестовая история IP SLA

Этот раздел описывает свойства истории тестов IP SLA. Более подробно о свойствах можно почитать в ссылках Cisco IP SLA и файле CISCO-RTTMON-MIB.

Эта таблица содержит расширенные свойства IP SLA. В большинстве случаев они не должны редактироваться вручную.

Свойства

Свойства	Описание
Количество жизней истории (rttMonHistoryAdminNumLives)	Максимальное количество жизней истории для записи. Жизнь определяется обратным отсчетом (или переходом) объекта rttMonCtrlOperRttLife на ноль. Новая жизнь создается, когда тот же концептуальный контрольный ряд RTT перезапускается через переход объекта rttMonCtrlOperRttLife и его последующего обратного отсчета. Нулевое значение остановит сбор всех исторических данных.
Количество бакетов (rttMonHistoryAdminNumBuckets)	Максимальное количество бакетов истории для записи. Когда **Тип** теста - 'pathEcho', это значение напрямую представляет путь к цели. Для всех других **Типов** тестов это значение должно быть выставлено на количество операций для сохранения на одну жизнь. Когда **Количество бакетов** заполняется, самые старые записи удаляются, а самые последние бакеты **Количества бакетов** остаются.
Количество образцов (rttMonHistoryAdminNumSamples)	Максимальное количество образцов истории для записи в один бакет. Когда **Тип** теста - 'pathEcho', это значение напрямую представляет количество хопов по пути к цели, таким образом, мы можем поддерживать только 30 хопов. Для всех других **Типов** тестов это значение будет выставлено агентом на один.
Фильтр (rttMonHistoryAdminFilter)	Определяет фильтр для добавления результатов RTT к буферу истории: **none** - история не записывается. **all** - записываются результаты всех времен завершения и неудавшихся завершений. **overThreshold** - записываются результаты времени завершения выше **Порога**. **failures** - записываются (только) результаты неудавшихся операций.

19.1.24.4.12 Пороги тревог IP SLA

Каждый тест IP SLA имеет три порога тревог со стороны сервера:

- Порог Предупреждения
- Порог Критически важного события
- Порог Сбоя

Каждый из этих порогов определяет, какое измерение времени прохождения сигнала в обоих направлениях вызовет появление тревоги на определенном уровне.

19.1.24.4.13 Понимание частоты опроса

Каждый тест IP SLA имеет собственное расписание, включающее *частоту* выполнения теста. Таким образом, статистическая информация, собранная тестом и предоставленная через протокол SNMP, обновляется с той же частотой.

Однако структура данных SNMP, предоставленных устройствами с IP SLA, предполагает, что статусы и статистика всех тестов хранятся в больших таблицах. Каждая таблица содержит определенную часть информации о статусе по всем тестам. Например, есть таблица **rttMonLatestHTTPOperTable**, которая содержит подробную информацию о статусе выполнения по всем тестам HTTP.

Архитектура драйвера устройства SNMP AtomMind Network Managera, в свою очередь, предполагает, что таблицы SNMP опрашиваются как атомарные значения. AtomMind Server не обновляет информацию о каждом тесте

отдельно, вместо этого оно читает все таблицы статистики тестов, используя запросы GET BULK для оптимизации производительности.

Таким образом, пользовательский период синхронизации 504 всех переменных учетных записей устройств Cisco, содержащих статистику IP SLA, должны быть равны или короче, чем частота любого теста, выполняемого этим устройством. Несоблюдение этого правила приведет к потере результатов тестов IP SLA.

Обычно модуль IP SLA автоматически настраивает пользовательский период синхронизации таблиц результатов тестов IP SLA во время создания 1938 или импорта 1937 тестов. Однако системные администраторы должны позаботиться о соответствии частоты тестов периодам синхронизации, если в документации выполняются изменения конфигурации тестов устройства или сервера.

19.1.24.5 Просмотр статуса и статистики IP SLA

Текущие метрики и статистика производительность сети, собранные тестами IP SLA, представлены в простых для использования инструментальных панелях, таблицах, диаграммах и графиках.

Следующие режимы просмотра обработки и визуализации данных IP SLA представлены с модулем IP SLA:

- Группы IP SLA 1952
- Инструментальные панели IP SLA 1952
- Тревоги IP SLA 1952

19.1.24.5.1 Тестовые группы IP SLA

Дистрибутив включает несколько предварительно настроенных групп 751 тестов IP SLA, помогающих отслеживать статус тестов, сгруппированных по типу 1935.

Каждая группа отражает статус всех членов теста.

- Если некоторые тесты находятся в Неопределенном состоянии, статус группы - Неопределенный, иной
- Если некоторые тесты находятся в Критическом состоянии, статус группы - Критический, иной
- Если некоторые тесты находятся в состоянии Предупреждения, статус группы - Предупреждение, иной
- Статус группы Нормальный

19.1.24.5.2 Инструментальные панели IP SLA

Дистрибутив AtomMind Network Manager включает несколько инструментальных панелей 912 IP SLA:

- **Обзор IP SLA**. Эта обзорная инструментальная панель представляет группы тестов IP SLA, тесты, которые в данный момент работают с небольшими проблемами, тесты, имеющие активные тревоги нарушения пороговых значений и суммарный график статусов тестов.
- **Топ 10 IP SLA**. Эта обзорная инструментальная панель представляет Топ 10 тестов с наибольшим временем прохождения сигнала в обоих направлениях, Топ 10 тестов TCP-соединений, Топ 10 тестов DHCP, Топ 10 тестов echo, Топ 10 тестов FTP и Топ 10 тестов HTTP.
- **Детали теста IP SLA**. Эта инструментальная панель обеспечивает подробную информацию о тесте. Она включает таблицу информации по тесту, последнее измерение RTT, график RTT, диаграмму доступности теста. Более специфичные типы тестов имеют собственные дополнительные метрики на этой инструментальной панели, например:
  - HTTP тесты предоставляют подробную информацию о разрешающем времени DNS, времени соединения TCP и времени загрузки страницы.
  - Тесты VoIP предоставляют графики задержки, джиттера, потери пакетов и MOS.
  - Тесты Path Echo предоставляют таблицу RTT хоп-за-хопом.

19.1.24.5.3 Тревоги IP SLA

Модуль IP SLA включает набор тревог 779:

- **Нарушение IP SLA**. Возникает, когда измерение времени прохождения сигнала в обоих направлениях теста превышает один из порогов 1951.
- **Отключенный**. Возникает, когда один из тестов сообщает о статусе **Отключен**.
- **Недоступный**. Возникает, когда один из тестов сообщает о статусе **Недоступен**.

- **Критический**. Возникает, когда один из тестов сообщает о статусе **Критический**.

19.1.25 Мониторинг VoIP, видео и IPTV

В дополнение к управлению классическими сетями данных, AtomMind Network Manager может прекрасно мониторить голосовые и видео сети, которые крайне чувствительны к задержкам, джиттеру и другим специфичным метрикам. Успешное внедрение приложения для VoIP и видео требует интегрированную систему управления производительностью, предоставляющую ИТ команде соответствующие ключевые показатели качества.

AtomMind Network Manager предоставляет комплексное управление VoIP, Видео и IPTV сетями:

- Мониторинг качества услуг (QoS) для голосовых/видео сетей
- Безагентная работа через стандартные протоколы 1801 без перехвата трафика и анализа пакетов
- Комплексный мониторинг соединений LAN и WAN через трассировку пути 1827
- Поддержка всех тестов Cisco IP SLA 1934, относящихся к потоку VoIP и видео/аудио (Джиттер и Джиттер пути, VoIP Звонок, RTP и другие)
- Измерение (через IP SLA или зонды ПО/аппаратных средств) и визуализация потери пакетов, односторонние и круговые задержки, двухсторонний джиттер и Усредненная оценка качества речи (Mean Opinion Score - MOS)
- Мониторинг доступности 1853 IP-телефона
- Автоматические оповещения 779 при обнаружении снижения производительности и долгосрочных негативных трендов качества услуг
- Поддержка вендоров VoIP включает **Cisco**, **Avaya**, **Asterisk** и другие
- "Коробочный" мониторинг Менеджера звонков VoIP 1953 (включая Cisco Call Manager и Call Manager Express)
- Сбор Call Detail Record (CDR) (через SNMP-ловушки 1886 или Syslog 1883), обработка и отчетность
- Бесшовная интеграция с другими модулями мониторинга сети, например, для быстрого доступа к данным использования пропускной способности интерфейса

19.1.25.1 Мониторинг Asterisk

AtomMind Network Manager обеспечивает коробочный мониторинг Asterisk.

Модуль мониторинга Asterisk включает в себя две основные инструментальные панели 912, которые зависят от протокола:

- Общее состояние сервера Asterisk (AMI)
- Общее состояние сервера Asterisk (SNMP)

Инструментальная панель общего состояния Asterisk

Инструментальная панель общего состояния Asterisk обеспечивает обзор VoIP инфраструктуры компании. Она показывает:

- Статистику сервера Asterisk
- Активные каналы Asterisk (при наличии AMI протокола)
- Активные звонки Asterisk (при наличии SNMP протокола)

19.1.25.2 Мониторинг менеджера унифицированных коммуникаций (Ca

AtomMind Network Manager предоставляет "коробочный" мониторинг Менеджера унифицированных коммуникаций (Unified Communications Manager - UCM, ранее CallManager) и UCM Express.

Мониторинг модуля UCM включает две основные инструментальные панели 912:

- Обзор VoIP
- Обзор CallManager

Инструментальная панель обзора VoIP

Инструментальная панель обзора VoIP предоставляет обзор VoIP-инфраструктуры компании. Она отображает:

- Список менеджеров звонков VoIP
- Процент зарегистрированных и отклоненных звонков
- Средние потеря пакетов, джиттер, задержка и усредненная оценка качества речи (MOS) для каждого CallManager

Инструментальная панель обзора CallManager

Инструментальная панель обзора CallManager отображает подробную статистику одного экземплра CallManager:

- Количество и процент IP-звонков, сгруппированных по статусу звонка
- Адреса, регистрационные метки и статусы всех зарегистрированных звонков
- Хронологическая история зарегистрированных, незарегистрированных и отклоненных телефонных номеров
- Исторические графики потери пакетов, джиттера, задержки и усредненной оценки качества речи (MOS)

19.1.26 Операции управления

AtomMind Network Manager предоставляет широкий выбор административных операций, которые пользователь может инициировать интерактивно или автоматически, например, в ответ на событие мониторинга.

В целом, административные операции, доступные для различных типов управляемых объектов, бесчисленны. В зависимости от типа управляемого устройства, запущенных на нем приложений и предоставляемых им сервисов, можно использовать различные административные операции, такие как:

- HTTP-запросы, если Ваш HTTP-сервис преобразовывает определенные URL в операции управления, которые он может выполнить;
- SQL-запросы для систем БД;
- SNMP-ловушки или другие типы не запрошенных уведомлений, отправленных устройствам, если SNMP-устройство настроено выполнять какое-то действие по поступлению определенного события и пр.

В этой главе содержится информацию об обычных операциях управления, поддерживаемых AtomMind Network Manager:

- SNMP-управление при помощи набора SNMP-операций 1954.
- Удаленное выполнение скрипта командного процессора 1955 с использованием SSH.
- Действия сетевого управления 1955, например, отправка SNMP-ловушек или Syslog-сообщений, или запуск удаленных ПК по Wake-on-LAN.

Эти действия можно активировать интерактивно или автоматически, как, например, действие тревоги по исправлению ошибки или запланированной задачи.

19.1.26.1 SNMP-управление

Некоторые типы сетевых устройств и сервисов можно контролировать по SNMP. В целом, SNMP-операции управления - это набор **SNMP-**операций. Дополнительную информацию и примеры см. в разделе Управляющие устройства, использующие SNMP 1847.

19.1.26.2 Управление по WMI

WMI можно использовать для удаленного управления компьютерами с ОС Windows. Дополнительную информацию и примеры см. в разделе WMI 1850.

19.1.26.3 Удаленное выполнение скрипта

AtomMind Network Manager может загружать скрипты из AtomMind Server AtomMinda для удаленных машин по безопасному SSH-соединению.

Удаленное выполнение скрипта - это один из наиболее мощных и гибких методов управления. В целом, он позволяет выполнить дистанционно любой вид скрипта или приложения на управляемом хосте.

Информацию о поддержке SSH в AtomMind Network Manager см в разделе Мониторинг SSH-сервера 609

19.1.26.4 Действия сетевого управления

Существует несколько действий 87 для управления, установленных в корневом контексте 1559. Чтобы активировать одно из этих действий, необходимо:

- Кликнуть правой кнопкой мыши по **корневому** узлу (сервера 376) в **системном дереве**.
- Выбрать подменю **Сетевое управление**
- Выбрать действие, которое необходимо запустить.

Появится окно, позволяющее задать параметры ввода для выбранного действия. Нажмите **OK** для продолжения работы с заданными параметрами.

Действия сетевого управления, доступные в меню **Сетевое управление**, кратко описаны в подразделе ниже.

19.1.26.4.1 Выполнить удаленный скрипт

Действие **Выполнить скрипт на удаленном сервере, используя SSH** (`executeRemoteScript`), загружает скрипт на удаленный хост по безопасному SSH-соединению и выполняет его там же.

См. разделы Выполнение удаленного скрипта 1955 и Мониторинг SSH-сервера 609.

19.1.26.4.2 Отправить SNMP-ловушку

Действие **отправить SNMP-ловушку** (`sendSnmpTrap`) позволяет передавать SNMP-ловушку удаленному хосту. См. раздел Отправка SNMP-ловушек 1848.

19.1.26.4.3 Отправить Syslog-сообщение

Действие **Отправить Syslog-сообщение** (`sendSyslogMessage`) позволяет передавать Syslog-сообщения удаленному хосту. См. раздел Отправка Syslog-сообщений 1885.

19.1.26.4.4 Wake-on-LAN

Действие **Wake-on-LAN** (`executeWakeOnLAN`) позволяет разбудить компьютер сетевым сообщением.

Действие Wake-on-LAN (WoL) транслирует специальный UDP-пакет, содержащий MAC-адрес получателя. Чтобы действие WoL работало, сетевой интерфейс получателя должен находится в состоянии ожидания адресованного ему этого пакета, после чего оно инициализирует пробуждение системы.

Чтобы активировать действие, Вы можете кликнуть правой кнопкой мыши по узлу сервера в Системном дереве, затем выбрать группу меню **Сетевое управление** и там выбрать `Wake-on-LAN`. Должно появится окно с параметрами. Вы можете дополнительно создать запланированную задачу 823 для данного действия.

У этого действия следующие параметры:

Имя	Тип	Описание	Информация
macAddress	Строка	MAC-адрес	MAC-адрес получателя в форме `XX:XX:XX:XX:XX:XX` либо `XX-XX-XX-XX-XX-XX`.
broadcastIP	Строка	Широковещательный IP-адрес	Широковещательный IP-адрес для Вашей сети. Например, если Ваш сетевой IP-адрес *192.168.0.x* , а маска *255.255.255.0* ,

			тогда широковещательный IP-адрес будет *192.168.0.255.*
udpPort	Целое	Порт Wake-on-LAN (UDP)	UDP-порт, куда следует отправить магический пакет. Значение по умолчанию равно 9.

19.1.27 Интеграция с Helpdesk / Системами поддержки клиент

AtomMind Network Manager можно объединить практически с любой системой заявок или службой поддержки клиентов (helpdesk). Основная цель интеграции - автоматическое создание заявок на устранение технических неисправностей, когда инициирована тревога 779 AtomMind.

В целом AtomMind уведомляет систему службы поддержки клиентов, отправляя ей сообщение и используя при этом правила уведомления 791 тревоги или автоматические действия по исправлению ошибки 793. Если требуется внимание администратора (например, для подтверждения о получении уведомления), можно использовать интерактивные действия по исправлению ошибок.

Ниже представлены подсказки для отправки тревожных сообщений службе поддержки клиентов:

- Отправка стандартного действия извещение по email 792 на адрес системы helpdesk
- Отправка пользовательских email системе helpdesk, используя действие по исправлению ошибок отправить E-mail 1559
- Отправить Syslog-сообщение
- Отправить SNMP-ловушку
- Выполнить скрипт 879, который подключается к серверу helpdesk, используя его API, и вводит новый тикет.

19.1.28 Управление активами

Плагин 207 *Управление активами* предназначен для связывания информации по отслеживанию активов с Вашими сетевыми устройствами.

Глобальная конфигурация плагина предоставляет следующие значения boolean:

- Информацию об активах
- Расположение активов
- События активов

Активирование этих значений заставляет AtomMind Server создать глобальную общую таблицу 2176 и заполнить ее всеми устройствами системы. Формат для каждой таблицы можно затем отредактировать в соответствии с необходимыми для Вашей компании требованиями для отслеживания.

Информацию об отслеживании активов, связанную с определенным устройством, можно отредактировать, используя действие Редактировать пользовательские свойства 2178.

Плагин Управление активами предоставляет удобный способ активировать хранилище используемой обычно информации об отслеживании активов. Вы можете задать новые параметры для отслеживания активов и создать таблицы, используя свойство пользовательские свойства 2177.

Структура информации об отслеживании активов по умолчанию

Пользовательское свойство Информация об активах:

- Серийный номер (строка)
- Производитель (строка)
- Модель (строка)
- Приобретено (дата, нулевой)
- Ответственное лицо (строка)
- Контактная информация (строка)
- Примечания (строка / редактор текстовой области)
- Фото (блок данных / редактор изображений)

Пользовательское свойство расположения активов:

- Строка (строка со значениями выборки)
- Город (строка)
- Строение (строка)
- Этаж (строка)
- Офис (строка)

Пользовательское свойство событий активов в форме таблицы:

- Дата (Дата / Редактор дат)
- Тип (строка)
- Комментарий (строка / редактор текстовой области)

19.1.28.1 Модели инвентаризации

Дистрибутив AtomMind Network Manager включает несколько моделей[810], которые находят подробную информацию об устройстве и активах.

Определитель типа устройств

Эта модель включает несколько наборов правил, каждый из которых проверяет, принадлежит ли устройство определенному типу. Базовый набор правил этой модели определяет последовательность вызова других наборов правил во время выполнения алгоритма определения типа устройства.

Наборы правил определения типа настраиваются с целью работы в двух "режимах":

- в одном случае они испоьзуют SNMP-данные, доступные в существующем контексте устройства
- в других случаях выполняются прямые SNMP-операции (которые происходят во время обнаружения устройств[1811], когда учетные записи устройств еще не созданы)

Инвентаризация устройств

Эта модель включает несколько наборов правил для определения операционной системы устройства. Модель будет расширяться в будущих версиях AtomMind Network Manager с целью обнаружения большего количества информации о ресурсах, предлагаемых хостом сети.

19.1.29 Управление IT услугами

Enter topic text here.

19.1.30 Справочник для контекста сетевого управления

В этой главе содержится информация о контекстах, переменных, функциях, событиях и действиях, которые относятся к сетевому управлению.

Плагин "Сетевое управление" осуществляет контроль за двумя системными контекстами: сетевым управлением[1967] и обнаружением[1962]. Они описаны в соответствующих разделах ниже. Кроме того, он добавляет действия, переменные, функции и события к другим (уже существующим) контекстам, как описано в этом разделе.

Управление

Плагин Сетевое управление использует контекст Администрирование[690] в качестве источника для SNMP- и Syslog-событий.

СОБЫТИЯ

Имя	Описание	Информация
syslog	Syslog-событие	Это событие генерируется при получении Syslog-сообщения. См. Консолидация и мониторинг Syslog-событий[1883].
trap	SNMP-ловушка	Это событие генерируется при получении SNMP-уведомления (Ловушки или Сообщения (Inform)). См Получение SNMP-ловушек и сообщений[1846].

Контекст устройства сетевого хоста

Плагин Сетевое управление использует устройства сетевого хоста 593 в качестве первичного источника для мониторинга данных и обеспечивает их переменными и действиями.

ПЕРЕМЕННЫЕ

Следующие переменные добавлены в контексты, представляющие устройства сетевого хоста:

Имя	Описание	Информация
processList	Список процессов	Хранит статистику о процессах, запущенных на узле (если эта информация доступна по SNMP.
vmCPU	Информация о CPU виртуальной машины	Хранит статистику об использовании CPU гостевыми виртуальными машинам, запущенными на узле (если там был обнаружен сервер VMware ESX; см. Мониторинг сервера VMware).
vmHBA	Информация о HDD виртуальной машины	Хранит статистику об использовании HDD гостевыми виртуальными машинами, запущенными на узле (если там обнаружен сервер VMware ESX; см. Мониторинг сервера VMware).
printerStates	Состояния принтеров	Для всех принтеров с SNMP хранит конфигурацию состояния, как описано в главе Мониторинг принтера 1917.

ДЕЙСТВИЯ

Имя	Описание	Информация
setupNetworkMonitoringProfile	Установить аккаунт мониторинга	Предлагает сервис "мастера настройки" для настройки инструментария для мониторинга 1808 устройства. Когда инструментарий создан, он добавляет действие, которое помогает активизировать инструментарий для устройства.
rediscoverServices	Сервисы повторного обнаружения	На этом устройстве доступны сервисы повторного обнаружения 1820.

Корень

Плагин Сетевое управление добавляет несколько действий, облегчающих задачу управления корневым контекстом 1559.

ДЕЙСТВИЯ

Имя	Описание	Информация
setupMonitoring	Настроить мониторинг	Запускает помощника по настройке мониторинга через определение средств мониторинга и устройства (путем выбора из существующих или создания нового). Для более подробной информации см. раздел Настройка мониторинга 1809.
servicesStatus	Статус сервисов	Отображает статистику для всех зарегистрированных сервисов.
sendSnmpTrap	Отправить SNMP-ловушку	Предоставляет действие отправки SNMP-уведомления, как описано в разделе Отправка SNMP-ловушек и сообщений 1845.
sendSyslogMessage	Отправить Syslog-сообщение	Предоставляет действие отправки Syslog-сообщений, как описано в разделе Отправка Syslog-сообщений 1885.
executeWakeOnLAN	Wake-on-LAN	Предоставляет действие отправки Wake-on-LAN сообщений. См раздел Wake-on-LAN 1955.

19.1.30.1 Политики соответствия

Этот контекст[41] является контейнером, содержащим все политики соответствия[1929] для определенного пользователя.

Уникальные действия [?[1450]]

СОЗДАТЬ НОВУЮ ПОЛИТИКУ (Действие по умолчанию[88])

Это действие используется для создания новых политик соответствия. Оно позволяет пользователю определять основные свойства[1930] новой политики.

Тип действия: Создать[105]

Права доступа: Доступны на уровне[486] с правами доступа *Менеджер*

Общие действия [?[1450]]

Создать на основе шаблона[105], Дублировать в дочерний контекст[111], Импорт[108], Экспорт[108], Редактировать права доступа контекста[106], Мониторить соответствующие события[109], Искать/Фильтровать[110], различные Групповые действия[101] в соответствии с дочерними контекстами.

Состояния и иконки контекста

У этого контекста нет состояний[44]. Он часто представляется иконкой  .

Расширенная информация

Информация о контексте

Тип контекста[43]: compliancePolicies

Имя контекста[42]: compliance_policies

Описание контекста[43]: Compliance Policies

Путь контекста[42]: users.USER_NAME.compliance_policies

Маска контекста[44]: users.*.compliance_policies

Права доступа контекста [?[44]]

Уровень	Описание
Отсутствует	Доступ не разрешен.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт политики.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (Свойства) [?[61]]

Этот контекст не имеет общих переменных (свойств).

Общие функции [?[70]]

Общие функции: makeCopy (Создать копию)[71], delete (Удалить)[72]

СОЗДАТЬ

Создает новую политику соответствия.

Имя функции:	create
Права доступа:	доступно на уровне 486 с правами доступа *Менеджер*
Записи ввода:	1
Формат 50 ввода:	Такой же, что и формат ввода переменной childInfo 1961 в контексте Политика соответствия 1960, только поле **текст** скрыто.
Записи вывода:	0
Формат 50 вывода:	отсутствует

Общие события [? 73]

Общие события: info (Информация) 77

19.1.30.2 Политика соответствия

Этот контекст 41 позоляет получить доступ и управлять одной политикой соответствия.

Уникальные действия [? 1450]

КОНФИГУРИРОВАТЬ (Действие по умолчанию 88)

Это действие используется для редактирования свойств 1930 политики соответствия.

Изменение **Имени** поля во время операции приведет к переименованию текущего контекста. Это может привести к неправильному функционированию других компонентов системы, использующих имя/путь контекста как основные идентификаторы.

Тип действия:	Конфигурировать 105

Общие действия [? 1450]

Удалить 106, Создать копию 109, Дублировать 110, Редактировать права доступа контекста 106, Мониторить соответствующие события 109, Посмотреть статус 111

Состояния и иконки контекста

У этого контекста нет состояний 44. Он всегда представлен иконкой  .

Расширенная информация

Информация о контексте

Тип контекста 43: compliancePolicy

Имя контекста 42: предоставляется пользователем

Описание контекста 43: предоставляется пользователем

Путь контекста 42: users.USER_NAME.compliancePolicies.SCRIPT_NAME

Маска контекста 44: users.*.compliancePolicies.*

Права доступа контекста [? 44]

Уровень	Описание
Отсутствует	Доступ не разрешен.
Наблюдатель	Мониторинг основных событий.

	Просмотр статуса.
Оператор	Тот же, что у Наблюдателя.
Менеджер	Конфигурация и удаление политики.
Инженер	Тот же, что у Менеджера.
Администратор	Тот же, что у Менеджера.

Общие переменные (Свойства) [?61]

Общие переменные: groupMembership (Членство группы)67, activeAlerts (Активные тревоги)

СВОЙСТВА

См. описание переменной этого поля здесь1930.

Имя переменной: childInfo

Записи: 1

Права доступа: Можно читать на уровне486 с правами доступа *Наблюдатель*, записывать с правами доступа *Менеджера*

Формат50 записи:

Имя поля	Тип поля	Примечания
name	Строка	1 - 50 символов
description	Строка	1 - 50 символов
severity	Целое	

ПРАВИЛА

См. описание переменной этого поля здесь1930.

Имя переменной: rules

Записи: 1

Права доступа: Можно читать на уровне486 с правами доступа *Наблюдатель*, записывать с правами доступа *Менеджера*

Формат50 записи:

Имя поля	Тип поля	Примечания
rule	Строка	
severity	Целое	
mode	Целое	
expression	Строка	

Общие функции [?70]

У этого контекста нет общих функций.

Общие события [?73]

Общие события: info (Информация) 77

19.1.30.3 Обнаружение

Обнаружение (**discovery**) - это системный контекст, который предоставляет операции по обнаружению сетевого устройства и данных. Он не отображается в видимой части контекстного дерева.

Уникальные действия [? 1450]

ОБНАРУЖЕНИЕ СЕТЕВОГО УСТРОЙСТВА

Это действие используется для обнаружения сетевых устройств и создания их учетных записей в AtomMind Network Manager. См. раздел Сетевое обнаружение 1811.

Иконка действия:	
Имя действия:	networkDeviceDiscovery
Неинтерактивный режим 99:	Не поддерживается
Права доступа:	Доступно на уровне 486 с правами доступа *Наблюдателя*.

Общие действия [? 1450]

Редактировать права доступа 106, Показать журнал событий 109.

Иконки и состояния контекста

У этого контекста нет состояний 44. Он всегда представлен иконкой .

Дополнительная информация

Информация о контексте

Тип контекста 43: обнаружение

Имя контекста 42: discovery

Описание контекста 43: Обнаружение

Путь к контексту 42: discovery

Маска контекста 44: discovery

Права доступа к контексу [? 44]

Уровень	Описание
Отсутствует	Доступ не разрешен.
Наблюдатель	Доступ не разрешен.
Оператор	Доступ не разрешен.
Менеджер	Доступ не разрешен.
Инженер	Доступ не разрешен.
Администратор	Все операции.

Общие переменные (свойства) [? 61]

У этого контекста нет общих переменных (свойств).

Общие функции [?70]

ОБНАРУЖЕНИЕ СЕТЕВЫХ УСТРОЙСТВ

Возвращает список обнаруженных устройств и связанных сервисов. В отличие от действия 87 **Обнаружение сетевых устройств** в контексте устройств 1494, функция не выполняет применение результатов обнаружения 1818, и поэтому не создает/не обновляет учетные записи устройств в AtomMind Network Manager.

Имя функции: discovery

Права доступа: Доступно на уровне 486 с правами доступа *Администратор*

Запись входа: 1

Формат 50 входа: Динамический, может включать любые из следующих полей (или вообще не иметь полей):

Имя	Тип	Описание
editServicesSelectionCommand	DataTable	Определяет *список сервисов* для обнаружения совместно с *параметрами,* использующимися для обнаружения доступности сервисов. Для большей информации см. Установка и выбор сервиса 1812.
seedRoutersCommand	DataTable	Определяет *Адреса исходных маршрутизаторов*. См. пункт (1) Определение адресов хоста 1815 для большей информации.
editRangesCommand	DataTable	Определяет *Диапазоны IP*. См. пункт (2) Определение адресов хоста 1815 для большей информации.
optionsCommand	DataTable	Определяет параметры Настроек обнаружения 1815.


При отсутствии какого-либо из вышеуказанных полей будут использованы параметры обнаружения по умолчанию 1820 вместо неопределенных параметров.


Чтобы увидеть пример того, как должны быть сконфигурированы эти поля, создайте *запланированную задачу* 823 для действия *Обнаружение сетевых устройств* и посмотрите на *параметры* конфигурации *запланированной задачи*. Вы можете экспортировать эту таблицу *параметров,* чтобы использовать её в качестве шаблона для входного аргумента функции *обнаружения*. Стоит заметить, однако, что для этого Вам не нужно поле *editAddressesCommand.*

Записи выхода: от 0 до бесконечности

Формат 50 выхода: Список полей выходных форматов и их описание см. в разделе Просмотр результатов обнаружения 1817.

Общие события [?73]

Общие события: info (информация) 77, contextStatusChanged (статус изменен) 83,

19.1.30.4 Тесты IP SLA

Этот контекст 41 является контейнером, содержащим все тесты IP SLA 1934 для определенного пользователя.

Уникальные действия [?1450]

ИМПОРТИРОВАНИЕ/ОБНОВЛЕНИЕ ТЕСТОВ

Это действие используется для импортирования или реимпортирования существующих тестов из устройства. См. детали здесь1937.

Имя действия:	importTests
Неинтерактивный режим99:	не поддерживается
Права доступа:	не поддерживаются

СОЗДАТЬ ТЕСТЫ (Действие по умолчанию88)

Это действие используется для создания новых тестов. См. подробности здесь1938.

Имя действия:	createTests
Неинтерактивный режим99:	не поддерживается
Права доступа:	не поддерживаются

Общие действия [?1450]

Создать на основе шаблона105, Дублировать в дочерний контекст111, Импорт108, Экспорт108, Редактировать права доступа контекста106, Мониторить соответствующие события109, Искать/Фильтровать110, различные Групповые действия101 в соответствии с дочерними контекстами.

Состояния и иконки контекста

У этого контекста нет состояний44. Он всегда представлен иконкой  .

Расширенная информация

Информация о контексте

Тип контекста43: ipSlaTests

Имя контекста42: ipsla

Описание контекста43: IP SLA

Путь контекста42: users.USER_NAME.ipsla

Маска контекста44: users.*.ipsla

Права доступа контекста [?44]

Уровень	Описание
Отсутствует	Доступ не разрешен.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менедежр	Создание, обновление, экспорт и импорт теста.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (Свойства) [?61]

У этого контекста нет общих переменных (свойств).

Общие функции [?70]

Общие функции: makeCopy (Создать копию)71, delete (Удалить)72

Общие события [?73]

Общие события: info (Информация)77

19.1.30.5 Тест IP SLA

Этот контекст41 позволяет получить доступ и управлять одним тестом IP SLA1934.

Уникальные действия [?1450]

СИНХРОНИЗИРОВАТЬ

Выполняет синхронизацию1939 настроек теста со стороны устройства и сервера.

Имя действия:	synchronizeTest
Неинтерактивный режим99:	не поддерживается
Права доступа:	Доступны на уровне486 прав доступа *Наблюдателя*

КОНФИГУРИРОВАТЬ

Это действие используется для редактирования свойств1939 Теста.

Изменение **Имени** поля во время операции приведет к переименованию текущего контекста. Это может привести к неправильному функционированию других компонентов системы, использующих имя/путь контекста как основные идентификаторы.

Тип действия:	Конфигурировать105

Общие действия [?1450]

Удалить106, Создать копию109, Дублировать110, Редактировать права доступа контекста106, Мониторить соответствующие события109, Просмотреть статус111

Состояния и иконки контекста

Иконка	Код	Состояние
	0	Состояние теста не определено.
	1	Результат теста критичный.
	2	Цель теста не достигнута.
	3	Цель теста отключена.
	4	Результат теста - предупреждение.
	5	Результат теста нормальный.

Расширенная информация

Информация о контексте

Тип контекста43: ipSlaTest

Имя контекста[42]: предоставляется пользователем

Описание контекста[43]: предоставляется пользователем

Путь контекста[42]: users.USER_NAME.ipsla.QUERY_NAME

Маска контекста[44]: users.*.ipsla.*

Права доступа контекста [?[44]]

Уровень	Описание
Отсутствует	Доступ не разрешен.
Наблюдатель	Мониторинг основных событий. Просмотр статуса и результатов теста.
Оператор	Те же, что у Наблюдателя.
Менеджер	Конфигурация, удаление и синхронизация теста.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (Свойства) [?[61]]

Общие переменные: groupMembership (Членство группы)[67], activeAlerts (Активные тревоги)[67]

СВОЙСТВА

Определяет основные свойства контекста теста.

Имя переменной: childInfo

Записи: 1

Права доступа: Можно читать на уровне[486] с правами доступа *Наблюдатель*, записывать на уровне с правами доступа *Менеджера*

Формат[50] записи:

Имя поля	Тип поля	Примечания
name	Строка	1 - 50 символов
description	Строка	1 - 50 символов
source	Строка	
testNumber	Целое	

ПОРОГИ ТРЕВОГ

Определяет пороги тревог для теста.

Имя переменной: alertThresholds

Записи: 1

Права доступа: Можно читать на уровне[486] с правами доступа *Наблюдатель*, записывать на уровне с правами доступа *Менеджера*

Формат[50] записи:

Имя поля	Тип поля	Примечания
warningThreshold	Длинное	
criticalThreshold	Длинное	
failureThreshold	Длинное	

КАНАЛЫ СТАТИСТИКИ

Эта переменная позволяет управлять статистическими каналами[2184] теста.

Имя переменной: statisticsProperties

Записи: 0...не ограничено

Права доступа: Можно читать на уровне[486] с правами доступа *Наблюдатель*, записывать на уровне с правами доступа *Менеджера*

Формат[50] записи:

Имя поля	Тип поля	Примечания
name	Строка	Имя канала.
variable	Строка	Имя переменной, на которой базируется канал.
properties	Таблица данных	Свойства[719] канала.

ДРУГИЕ СВОЙСТВА ТЕСТА

Другие переменные конфигурации теста:

- Параметры теста[1939]
- Расписание теста[1947]
- Реакции теста[1948]
- Статистика теста[1949]
- История теста[1951]

Общие функции [?[70]]

У этого контекста нет общих функций.

Общие события [?[73]]

Общие события: info (Информация)[77]

19.1.30.6 Сетевое управление

Сетевое управление (**netmanagement**) - это системный контекст, который предоставляет доступ к данным, относящимся ко всей работе пристроек/расширений сетевого управления. Он не отображается в видимой части контекстного дерева.

Плагин "Сетевое управление" использует SNMP[637], WMI[661] и драйверы устройств сетевого хоста[593] в качестве основы сетевой связи.

Дополнительная информация

Информация о контексте

Тип контекста[43]: networkManagement

Имя контекста[42]: netmanagement

Описание контекста[43]: сетевое управление

Путь к контексту[42]: netmanagement

Маска контекста[44]: netmanagement

Права доступа к контексту [?[44]]

Уровень	Описание
Отсутствует	Доступ не разрешен.
Наблюдатель	Обнаружение топологии сети.
Оператор	Тот же, что у Наблюдателя.
Менеджер	Отправка SNMP-ловушки. Отправка Syslog-сообщения. Отправка запросов Wake-on-LAN. Просмотр статуса услуг.
Инженер	Тот же, что у Менеджера.
Администратор	Тот же, что у Менеджера.

Общие переменные (свойства) [?[61]]

У этого контекста нет общих переменных (свойств).

Общие функции [?[70]]

ОТПРАВИТЬ SNMP-ЛОВУШКУ

Генерирует SNMP-уведомления (ловушки и сообщения), как описано в разделе Отправка SNMP-ловушек и сообщений[1848].

Имя функции: sendSnmpTrap

Права доступа: доступно на уровне[486] с правами доступа *Менеджера*

Записи ввода: 1

Формат[49] ввода: см. Отправка SNMP-ловушек и сообщений[1848]

Записи вывода: 0 (для ловушек) или 1 (для сообщений)

Формат[49] вывода: см. Отправка SNMP-ловушек и сообщений[1848]

ОТПРАВИТЬ SYSLOG -СООБЩЕНИЕ

Отправляет Syslog-сообщение, как описано в разделе Отправка Syslog-сообщений[1885].

Имя функции: sendSyslogMessage

Права доступа: доступно на уровне[486] с правами доступа для *Менеджера*

Записи ввода: 1

Формат[49] ввода: см Отправка Syslog-сообщений[1885]

Записи вывода: 0

Формат 49 вывода: нет

WAKE-ON-LAN

Активирует компьютер, отправляя ему специальное сообщение. См. Wake-on-LAN 1955.

Имя функции: executeWakeOnLAN

Права доступа: доступно на уровне 486 с правами доступа для *Менеджера*

Записи ввода: 1

Формат 49 ввода: см. Wake-on-LAN 1955

Записи вывода: 0

Формат 49 вывода: нет

ПОЛУЧИТЬ ИНФОРМАЦИЮ О ПРОЦЕССЕ

Функция утилиты, используемая инструментарием для мониторинга процесса 1860 для получения данных (параметров или строки пути) о процессах.

Имя функции: getProcessInfo

Права доступа: доступно на уровне 486 с правами доступа *Менеджера*

Записи ввода: 1

Формат 49 ввода:

Имя	Тип	Описаие
contextName	String	Имя контекста устройства.
processName	String	Имя процесса.
requestType	Integer	0 чтобы получить пути процессов с заданным именем, отфильтрованным параметрами; 1 чтобы получить параметры процессов с заданным именем, отфильтрованным путем.
processFilter	Integer	Строка фильтра, которую следует применить к параметрам или пути, в зависимости от заданного requestType.

Записи вывода: 0...не ограничено

Формат 49 вывода:

Имя	Тип	Описание
Значение	String	Имя контекста устройства.
Описание	String	Путь процесса или строка параметров.

СТАТУС СЕРВИСОВ

Функция утилиты, используемая инструментарием для мониторинга сервисов 1888 для получения информации о статусе от всех сервисов, зарегистрированных для всех устройств.

Имя функции: servicesStatus

Права доступа: Доступно на уровне 486 с правами доступа *Менеджера*

Записи ввода: 0

Формат 49 ввода: нет

Записи вывода: 0...не ограничено

Формат 49 вывода: Динамический формат со следующими полями:

- Поле типа String, именуемое "устройство" (содержащее путь к контексту)
- Одно поле типа Color для каждого зарегистрированного сервиса, обозначающего свой статус

ПОЛУЧИТЬ НЕСКОЛЬКО SNMP-ЗНАЧЕНИЙ

Функция "Получить несколько SNMP-значений" (snmpGetMulti) получает несколько "необработанных" значений с SNMP устройства. Функция отправляет один запрос SNMP GET каждому OID, обозначенному в параметрах входа и обрабатывает каждый ответ как отдельное значение, конвертируя его в String. Также см. комментарии и примеры к функции snmpGet 1967.

Имя функции: snmpGetMulti

Права доступа: Доступно на уровне 486 с правами доступа *Менеджера*

Записи ввода: 1

Формат 49 ввода:

Имя	Тип	Описание
oids	Table	Таблица OID для запроса. Каждый ряд должен представлять один OID для запроса в первом столбце (по умолчанию называемом 'oid').
settings	Table	SNMP настройки, используемые для соединения с SNMP устройством. Для более подробной информации см. Настройки опроса SNMP 640.

Записи вывода: 0...не ограничено

Формат 49 вывода:

Имя	Тип	Описание
name	String	Имя запрашиваемой переменной SNMP (включая ее OID).
value	String	Полученное значение, конвертированное в String.

SNMP READ

Извлекает "полноценные" SNMP переменные из SNMP устройства. В отличие от SNMP Get 1967, эта функция вызывает SNMP Walk для извлечения всех значений в пределах обозначенных OID и обрабатывает результаты вместе с метаданными из директории MIB-файлов 644 с составлением полных таблиц.

Пример: Выполнение функции SNMP Read с OID=1.3.6.1.2.1.2.2 возвращает полное содержимое ifTable.

Имя функции: snmpRead

Права доступа: Доступно на уровне 486 с правами доступа *Менеджера*

Записи ввода: 1

Формат 49 ввода:

Имя	Тип	Описание
oids	Table	А Таблица OID для запроса. Каждый ряд должен представлять один OID для запроса в первом столбце (по умолчанию называемом 'oid').
settings	Table	SNMP настройки, используемые для соединения с SNMP устройством. Для более подробной информации см. Настройки опроса SNMP 640.

Записи вывода: 0...не ограничено

Формат 49 вывода:

Имя	Тип	Описание
name	String	Имя полученной переменной SNMP (включая ее OID).
value	Table	Значение переменной SNMP в виде таблицы.

ПРОВЕРКА СВЯЗИ (PING)

Отправляет запрос на проверку связи определенному хосту.

Имя функции: ping

Права доступа: Доступно на уровне 486 с правами доступа *Менеджера*

Записи ввода: 1

Формат 49 ввода:

Имя	Тип	Описание
address	String	IP адрес или имя хоста для ping.
packetDataSize	Integer	Размер ICMP пакета данных (исключая заголовки).
pingCount	Integer	Число ping запросов в течение каждой синхронизации (цикла опроса).
storageSize	Integer	Количество пакетов, используемое для вычисления процента потери. Для расчета процента потери пакетов берутся последние результаты пинга в количестве, определенном данной настройкой.
offlineCriterion	Integer	Количество последовательных оставшихся без ответа ping запросов, которое переведет сетевой хост в статус подключения 'нет подключения'.
timeout	Long	Если ответ не получен по истечению данного времени, запрос ping считается оставшимся без ответа.

Записи вывода: 1

Формат 49 вывода: см. Результаты мониторинга пинга 597

Общие события [? 73]

У этого контекста нет общих событий.

19.2 SCADA/HMI

AtomMind SCADA/HMI -- это автоматизированная система, позволяющая осуществлять контроль за производственными процессами, основанная на ядре AtomMind. Она хорошо интегрируется с другими интеллектуальными решениями на базе AtomMind по управлению устройствами.

Основные возможности AtomMind SCADA/HMI

- Кросс-платформенная, работает под управлением Windows, Linux, MacOS и другими операционными системами с Java.
- Архитектура, ориентированная на использование на баз данных, большинство современных СУБД можно использовать для хранения данных, включая Oracle, MS SQL, MySQL, PostgreSQL и пр.
- Расширенная поддержка протоколов: OPC, Modbus, BACnet, SNMP, SQL, DCOM и др.
- Комплект разработчика драйверов (DDK) с открытым исходным кодом
- Виртуальные устройства для тестирования и отладки
- Инновационная, патентованная технология нормализации данных
- Улучшенный редактор HMI и интегрированный редактор отчетов
- Поддержка табличной и абсолютной разметки HMI
- Более 50 визуальных компонентов и тысячи динамически настраиваемых параметров
- Обширная библиотека для автоматизации и контроля
- Унифицированная разработка/среда выполнения, тестирование "на лету"

- Поддержка сигналов оповещений и составления графиков
- Планировщик для выполнения периодических задач
- Интеграция с ERP и другими системами через Веб-сервис и API с открытыми исходными текстами
- Гибкая, основанная на ролях модель контроля доступа
- Интерфейс пользователя в виде приложения и веб-страниц
- Отказоустойчивая кластеризация и распределенная архитектура
- Поддержка локалазации, интернационализации и брендинга

Документация раздела AtomMind SCADA/HMI довольно короткая, поскольку основная часть функционала SCADA/HMI принадлежит AtomMind. Самые важные модули SCADA (тревоги 779, отчеты 928, видежеты 943 и т.д.) описаны в разделе AtomMind Server 144.

19.2.1 Архитектура

AtomMind SCADA/HMI базируется на AtomMind. Большая часть функций поставляется базовом варианте ПО.

С технической точки зрения, решение SCADA/HMI - это набор драйверов устройств 518 и плагинов 207, расширяющих AtomMind.

Ниже приведен список разделов руководства к AtomMind, которые в основном относятся к решению SCADA/HMI:

- Основы системы 38
- Внедрение и администрирование 144
- Пользователи 478
- Устройства 497
- Драйверы устройств 518
- Фильтры событий 762
- Тревоги 779
- Отчеты 928
- Запланированные задачи 823
- Виджеты 943
- Скрипты 879
- Разработка и интеграция 1339

Отсутствие среды разработки и выполнения

В отличие от традиционных систем SCADA, у AtomMind SCADA/HMI есть унифицированное окружение, которое используется для разработки, отладки, тестирования и обычных операций. Система не разделена на отдельные модули для разработки и выполнения.

Ядро системы - AtomMind Server. Сервер поддерживает соединение с базой данных 692, которая используется для хранения системных настроек, HMI-шаблонов, свойств PLC, кэширования каналов и меток, а также данных за истекший период.

Сервер общается с контроллерами, сенсорами и исполнительными механизмами через предназначенные для них драйвера устройств 518. В то время как современные контроллеры, как правило, выполняют операции, близкие к реальному времени, согласно собственной логике, AtomMind SCADA/HMI позволяет серверу запускать операции управления, которыми можно управлять через специальные скрипты 879 или действия 87.

Отсутствие проектов

У AtomMind SCADA/HMI нет проектов. Согласно первичной концепции, на производственном объекте используется один "первичный" сервер. Во время начальной фазы внедрения, системные инженеры могут подсоединиться к серверу локально или удалено, чтобы построить HMI, создать учетные записи PLC, настроить хранилище данных и пр. После завершения начального этапа настройки, в обычном режиме работы будет использоваться тот же пользователь. Не требуется переключения на другую среду для работы.

Можно легко внести изменение в систему, не прерывая ее работы:

- Сделать временные копии одного или нескольких компонентов системы (например, HMI или Тревоги)
- Изменять, сравнивать и тестировать компоненты
- Восстановить одну из временных копий, чтобы использовать вместо оригинальной

В любое время можно сделать или восстановить 173 резервную копию 173 AtomMind SCADA/HMI.

Каналы ввода-вывода и теги

В рамках AtomMind SCADA/HMI, каналы ввода-вывода (теги) называются **Настройками устройства** (или переменными устройства). Эти настройки могут быть как скалярными, так и табличными 49.

Ввиду технологии нормализации данных, используемой унифицированной моделью данных 41 AtomMind, значение каждого канала ввода-вывода (тега), даже скалярное, представляется в виде таблицы данных 49. Эта технология во многом увеличивает гибкость AtomMind SCADA/HMI, позволяя тем самым оперировать любыми системными компонентами, (как сырыми данными устройства, так и подготовленными данными). Кроме того, это позволяет выстраивать комплексные многоуровневые цепочки обработки данных и позволяет внешним системам иметь доступ к любым данным системы.

19.2.2 Начало работы

Подробную пошаговую инструкцию по запуску вашего продукта/решения по автоматизации производства или зданий см. в разделе Процесс разработки приложений 40.

Документация раздела AtomMind SCADA/HMI довольно короткая, поскольку основная часть функционала SCADA/HMI принадлежит AtomMind.

19.2.3 Подключение к ПЛК

AtomMind SCADA/HMI получает доступ к ПЛК, сенсорам, SQL-базам данных и другим источникам данных через Учетные записи устройства 497. Каждая учетная запись использует определенный Драйвер Устройства 518 для того, чтобы подключиться к физическому устройству, прочитать его мета-данные и выполнить ввод-вывод данных. О том, как создать учетную запись для нового устройства, можно узнать в главе Добавление Новых Устройств 498 .

После того, как учетная запись нового устройства создана, будет выполнена полная синхронизация 514 устройства с сервером. После ее завершения устройство должно переключится на режим обычной работы, который обозначен иконкой с зеленой галочкой: . Это состояние означает, что канал ввода-вывода и операции, выполненные устройством, кэшированы сервером и открыты для доступа такими системными средствами, как HMI и Тревоги.

Если необходимо включить хранилище значений канала ввода-выводы, теперь это можно выполнить, изменив **Время запоминания истории** в диалоговом окне Опции синхронизации настроек устройства 502.

19.2.4 Создание HMI

В рамках AtomMind SCADA/HMI, интерфейс "Пользователь-Машина" (HMI) - это стандартный виджет 943. Для виджета HMI, как правило, характерно наличие следующих элементов:

- Абсолютная (без привязки к сетке) компоновка, которая упрощает размещение компонентов
- Опциональное фоновое изображение, отображающее общий вид управляемого промышленного процесса
- Динамические изображения 993 (трубы, котлы, вентиляторы и пр.), размещенные на подложке, отображающие состояния соответствующего оборудования
- Метки 956, Текстовые поля 958, Регуляторы 987 и другие компоненты 955, используемые для мониторинга определенных метрических значений и изменения установленных значений
- Кнопки для открытия других HMI через действия 87 сервера или в процессе работы оборудования.

Создание нового HMI

Чтобы создать новый виджет HMI в AtomMind Client, следует:

- Щелкнуть правой кнопкой мыши по узлу Виджеты ( ), расположенном в Системном Дереве
- Выберите из контекстного меню пункт **Создать HMI**
- Введите **имя** HMI и его **описание**. Обратите внимание, что имя может содержать лишь латинские буквы, цифры и символ подчеркивания.
- Выберите фоновое изображение для HMI. Размер виджета HMI будет автоматически вычислен пропорционально размеру фонового изображения.

- С этого момента запустится GUI приложение Редактор виджетов 423. Вы можете модифицировать Ваш HMI, добавлять для него новые компоненты и пр.

В избранном 2186 AtomMind SCADA/HMI имеется пункт **Создать HMI** для быстрого перехода к действию Создать HMI.

19.2.4.1 Библиотека символов

Дистрибутив AtomMind SCADA/HMI включает обширную библиотеку символов контроля и автоматизации. Символы представлены в распространенном формате SVG (Scalable Vector Graphics). Однако, все изображения в библиотеке дополнены для более удобного обращения с ними в HMI:

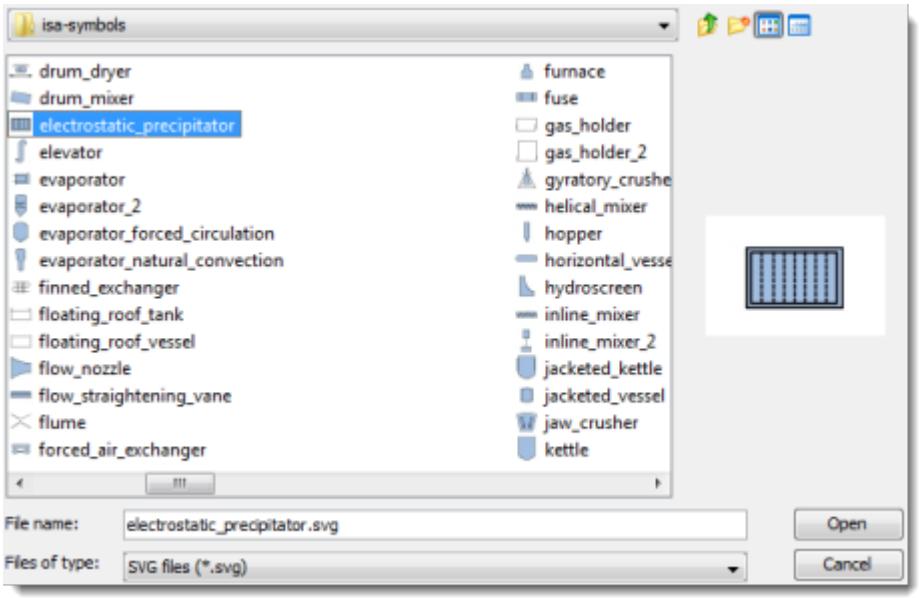
- Начало/останов анимации
- Перемещаемые/масштабируемые/поворачиваемые части изображения (например, контроль уровня жидкости в резервуаре)
- Отображаемые/скрытые части (например, при изменении положения)

Все дополнения сохраняют совместимость с исходным форматом SVG.

Использование компонентов

Чтобы вставить компонент в виджет HMI, используйте элемент виджета Векторная графика 995. После того, как в виджет добавлен новый элемент Векторная Графика, отредактируйте его свойство в файле **SVG и** загрузите один из компонентов, который располагается в поддиректории `/images` установки AtomMind.

Браузер файлов SVG отобразит предварительный просмотр, который упрощает выбор изображения:



Символы SCADA не включены в дистрибутив AtomMind Client. Если Вам необходимо создать HMI, используя независимую установку Клиента, скопируйте папку `/images` из Вашей установочной папки AtomMind Server в установочную папку AtomMind Client.

Категории символов

Список категорий, доступных в библиотеке символов AtomMind SCADA/HMI:

- Кондиционирование
- Стрелки
- Контрольное оборудование ASHRAE
- Лаборатория
- Механообработка
- Обработка материалов
- Добыча ископаемых
- Месилки

- Вентиляционные каналы ASHRAE
- Гидросистема ASHRAE
- Простые фигуры
- Воздухозаборники
- Бойлеры
- Химикаты
- Контейнеры
- Контроллеры
- Элементы управления
- Конвейеры
- Трубопровод
- Электрические установки
- Доводка
- Гибкие трубопроводы
- Измеритель расхода жидкости
- Пищевые продукты
- Общие
- Система отопления
- HVAC
- Двигательные установки
- Интерфейс оператора
- Панели
- Трубы
- Промышленное оборудование
- Силовая установка
- Охлаждение
- Обогрев
- Целлюлозно-бумажная промышленность
- Насосы
- Части труб
- Сенсоры
- Цистерны
- Пневмоаппараты
- Транспортные средства
- Водоочистка и водоподготовка
- Кабельные линии

19.2.5 Работа с графиками и трендами

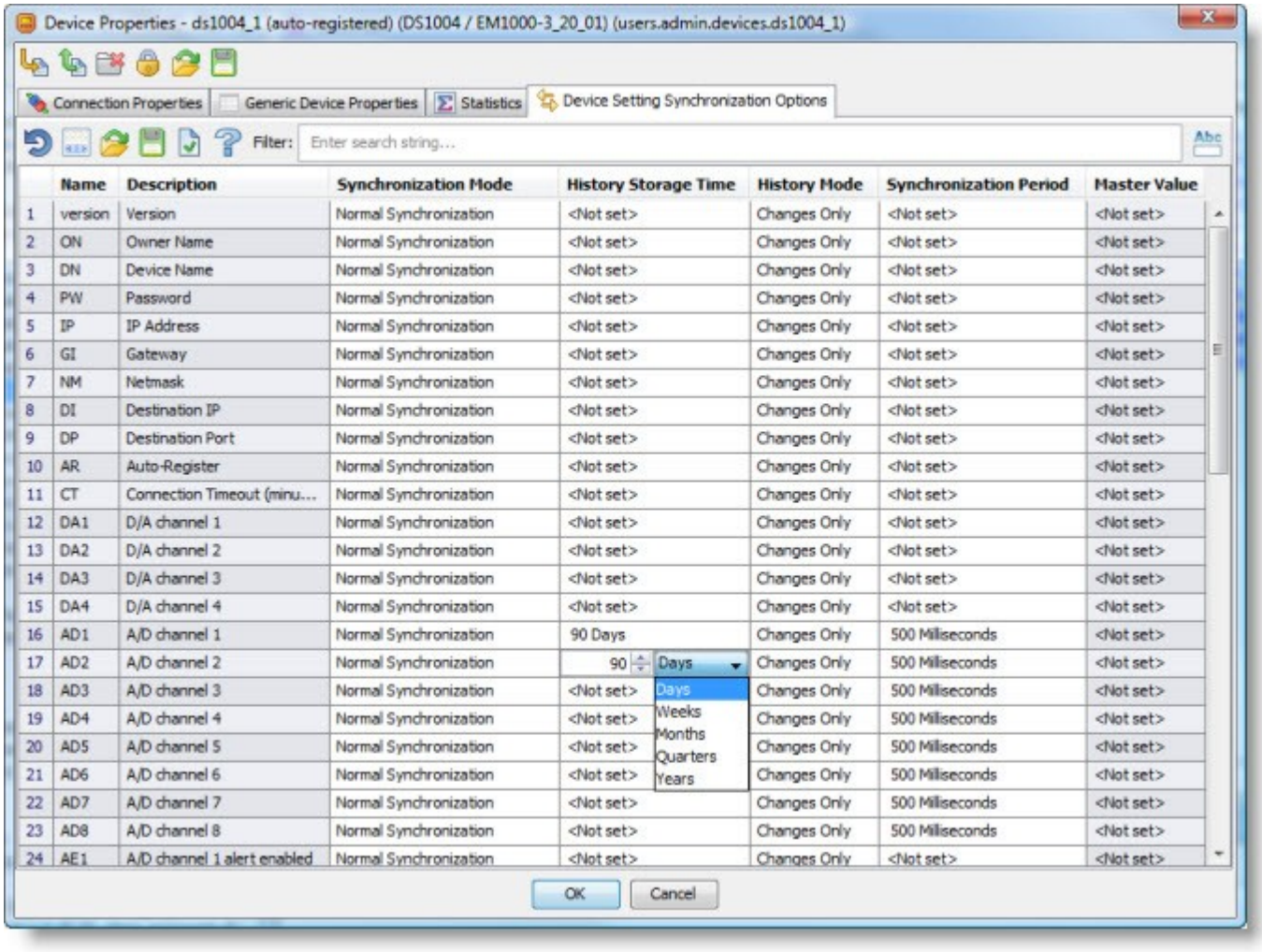
Построение графиков и диаграмм для значений каналов ввода-вывода включает в себя два этапа:

- Активация хранения истории канала (если ранее зафиксированные значения должны отражаться в тренде)
- Создание виджета с диаграммой

Активация истории канала

По умолчанию любой канал ввода-вывода в AtomMind SCADA/HMI не хранит ранее полученные значения. Чтобы активировать хранение истории через AtomMind Client, следует:

- Щелкнуть правой кнопкой мыши по учетной записи устройства ( ) в Системном Дереве.
- Выбрать **Редактировать свойства аккаунта устройства** ( ).
- Переключиться на вкладку **Параметры синхронизации настроек устройства** ( ) и установить требуемое время хранения для одного или нескольких свойств:



- Кликните OK, чтобы сохранить изменения.

Дополнительная информация содержится в главе Параметры синхронизации настроек устройства 502.

Построение виджета с графиками и диаграммами

В AtomMind SCADA/HMI любой график или диаграмма является компонентом виджета 943. См. раздел Диаграммы 1051, где описывается модель данных графиков/диаграмм, свойства и случаи использования.

Не сложно создать график/диаграмму для отображения ранее зафиксированных или текущих значений канала:

- Щелкните правой кнопкой мыши по узлу **Устройство** ( ) в Системном Дереве.
- Выберите **Конфигурировать устройство** ( ). Развернется диалоговое окно с настройками устройства (каналов ввода-вывода).
- Щелкните правой кнопкой мыши по каналу ввода-вывода, чтобы создать для него диаграмму, и выберите **Создать Диаграмму**.
- Следуйте инструкциям, описанным в разделе Создать диаграмму для переменной 1624.

Кроме того, возможно построить диаграмму для виджета с нуля, создав пустой виджет, добавив в него компонент диаграммы и настроив его свойства.

19.2.6 Безопасность SCADA

Созданный на AtomMind, AtomMind SCADA/HMI унаследовал его мощную и гибкую ролевую модель безопасности 477. У операторов есть отдельные учетные записи 478. Эти учетные записи могут владеть ресурсами (учетными записями контроллера, HMI, тревогами 779, отчетами 928, фильтрами событий 762). Система также позволяет делить функции просмотра/редактирования прав доступа между разными учетными записями. См. Безопасность и права доступа 477.

Безопасность HMI

Некоторые системы SCADA сопоставляют уровни права доступа и сверяют их с каждым компонентом интерфейса HMI. AtomMind SCADA/HMI использует более продвинутый метод контроля за тем, что тот или иной HMI может делать, а что нет:

- У HMI компонентов нет связанной с ними секретной информации.
- Текущий HMI наследует права доступа пользователя, который его запустил.
- Каждый запрос о чтении или записи данных сервера использует тот же самый уровень прав доступа (т.е. права доступа пользователя, который запустил виджет HMI)
- Права доступа проходят проверку на сервере и делают попытки взлома виджета клиента невозможными.
- Тонко настраиваемые права доступа настраиваются путем распределения между системными операторами различных уровней доступа к разным системным ресурсам и устройствам.

19.2.7 Обнаружение серверов OPC

AtomMind SCADA/HMI поддерживает *обнаружение серверов OPC*. Обнаружение OPC - это процесс сканирования множественных сетевых IP хостов, обнаружение запущенных на них серверов OPC и создание учетных записей для этих серверов.

Чтобы выполнить обнаружение OPC:

- Щелкните правой кнопкой мыши по узлу **Устройства** ( ) в Системном Дереве и выберите **Обнаружение OPC**( ).
- В диалоговом окне **Выбор и настройка сервисов для обнаружения** выберите список учетных рекизитов для Windows, которые будут использоваться для регистрации на хосты серверов OPC. Этот список доступен в поле **Параметры Доступа и Опции Сервиса**.
- Затем задайте диапазон **значений IP-адресов** или **IP-адрес**, которые будут просканированы**.**
- Когда процесс обнаружения закончится, откроется диалоговое окно **Результаты**. Теперь необходимо выбрать серверы OPC, чтобы создать для них учетные записи устройств.

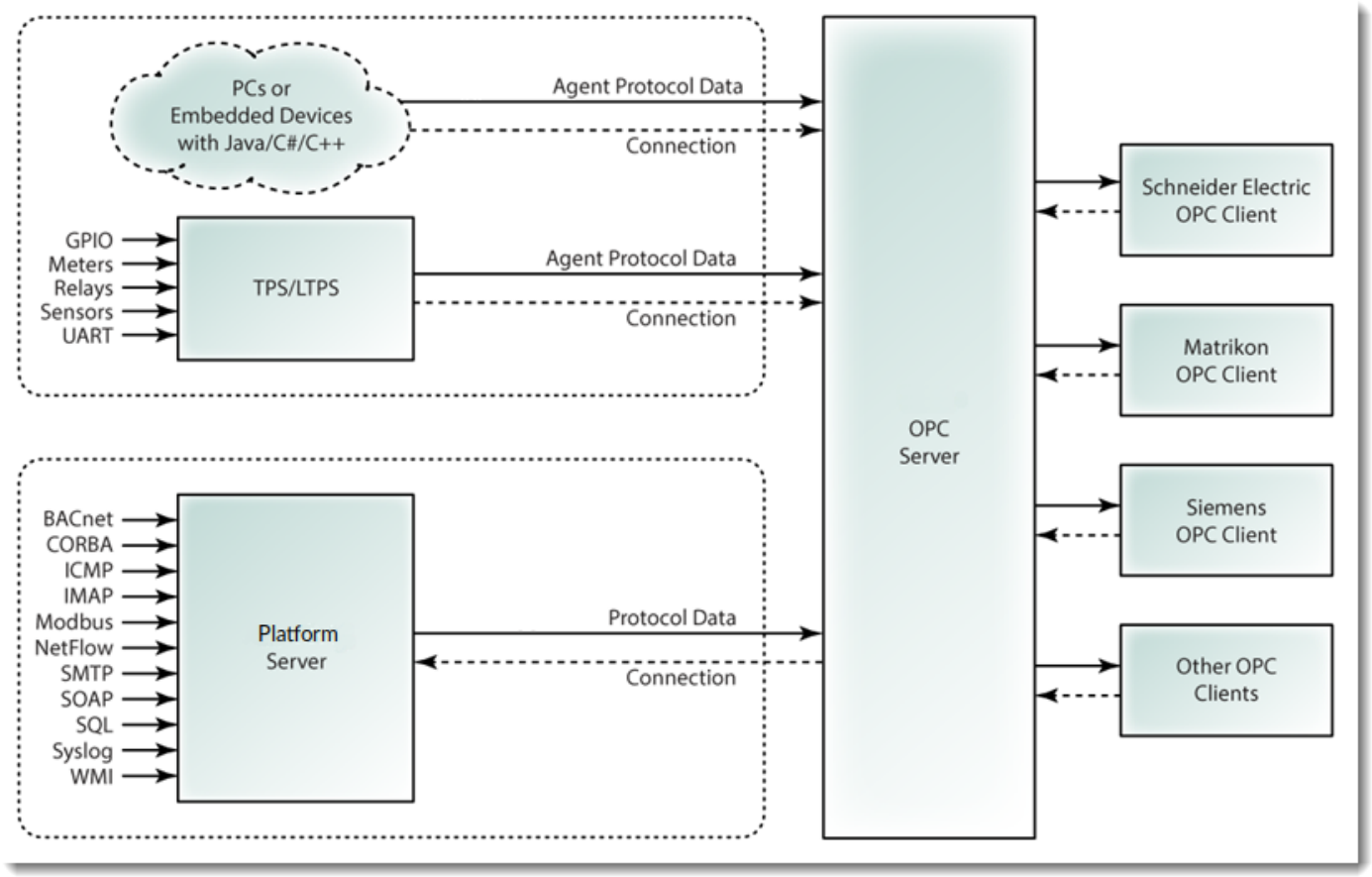
AtomMind SCADA/HMI требует использования сконфигурированного OPC-сервера с активированной опцией Просмотр.

19.2.8 OPC сервер

AtomMind OPC-сервер - это автономное Windows-приложение, которое раскрывает сторонним OPC-клиентам определенный набор переменных контекста единой модели данных 41 в форме OPC-тегов.

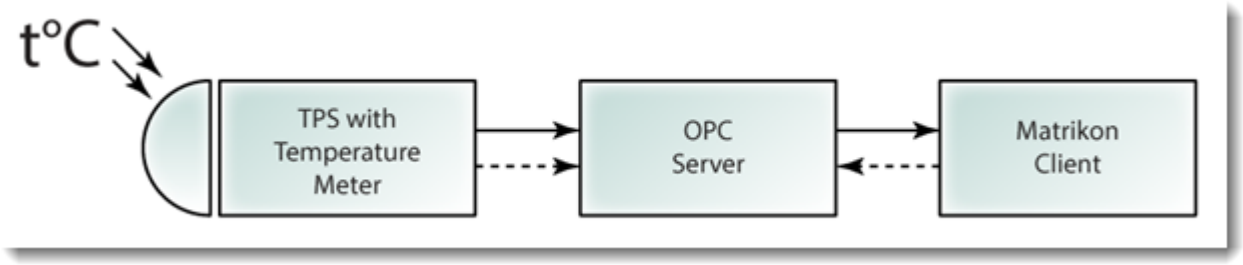
OPC- сервер поддерживает самую распространенную спецификацию OPC - OPC Data Access (OPC DA), которая используется для чтения и записи данных в режиме реального времени. Источником данных может быть любое устройство, поддерживающее протокол AtomMind.

OPC-сервер можно скачать с вебсайта АО "ТВЭЛ".

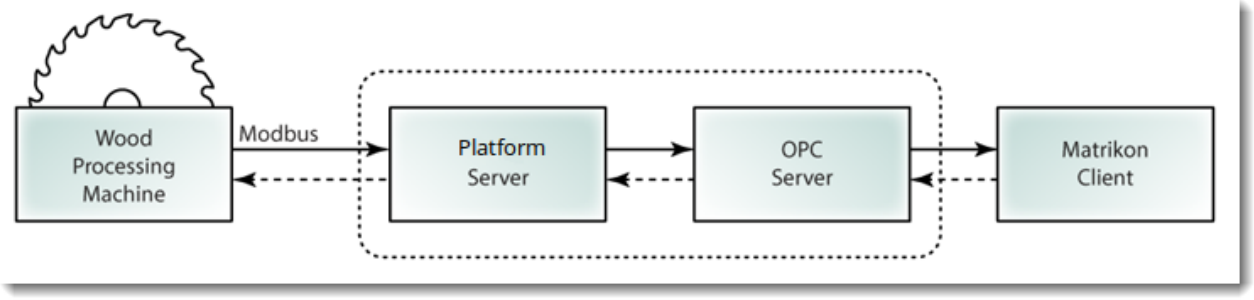


У OPC-сервера встроенный сетевой протокол AtomMind; он может взаимодействовать с любыми устройствами посредством протокола Agent, а также подключаться к AtomMind Server. Реализация с открытым исходным кодом протокола Agent опубликована для таких языков программирования как Java, C# и C++, поэтому схемы подключения не ограничиваются устройствами ТВЭЛ или AtomMind Server. Обновление данных происходит, когда изменения поступают от источника данных асинхронно.

Самый простой пример: платформа TPS считывает температуру и отправляет данные на OPC-сервер при помощи протокола Agent:



Более сложный пример. Допустим, что у Вас ПК с операционной системой Windows, контролирующий работу деревообрабатывающего станка при помощи AtomMind Server через протокол Modbus. Если OPC-сервер подключен к AtomMind Server, данные о работе станка передаются на OPC-сервер. В результате, у Вас есть возможность отслеживать и управлять оборудованием через любой OPC-клиент (например, Matrikon Client).



19.2.8.1 Технические характеристики

- Совместим с Windows XP/2003 и поздними версиями (требуется Microsoft Visual C++ 2013 без ограничений на свободное распространение; устанавливается автоматически)

- Поддерживает DA 2.0 асинхронный и синхронный с COM/DCOM.

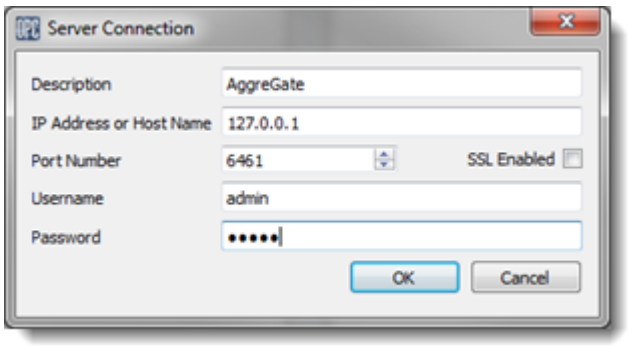
OPC-сервер отправляет приложениям OPC-клиента информацию о Value, Quality и Timestamp объекта (тэга). Эти поля считываются из переменных AtomMind Server.Если OPC-сервер теряет связь с их источником данных (AtomMind Server или Agent), значениям процесса присваивается состояние Bad [Configuration Error]. Если значение переменной пустое, указывается состояние Uncertain [Non-Specific]. Ниже представлена сравнительная таблица типов данных AtomMind Server и OPC:

Типы	**Типы OPC**
Integer	VT_I4.
Long	VT_I8.
Float	VT_R4.
Double	VT_R8.
String	VT_BSTR.
Boolean	VT_BOOL.
Color	VT_I4.
Date	VT_DATE.
Data Table	OPC VT_BSTR(по умолчанию).

Кроме того, поддерживается ряд простых типов данных, перечисленных выше (VT_ARRAY). Переменная типа Data Table может соответствовать различным типам данных OPC. По умолчанию таблица преобразуется в строку. Но если вы используете "Value field" в конфигурации, то из таблицы будет взято значение указанного поля.

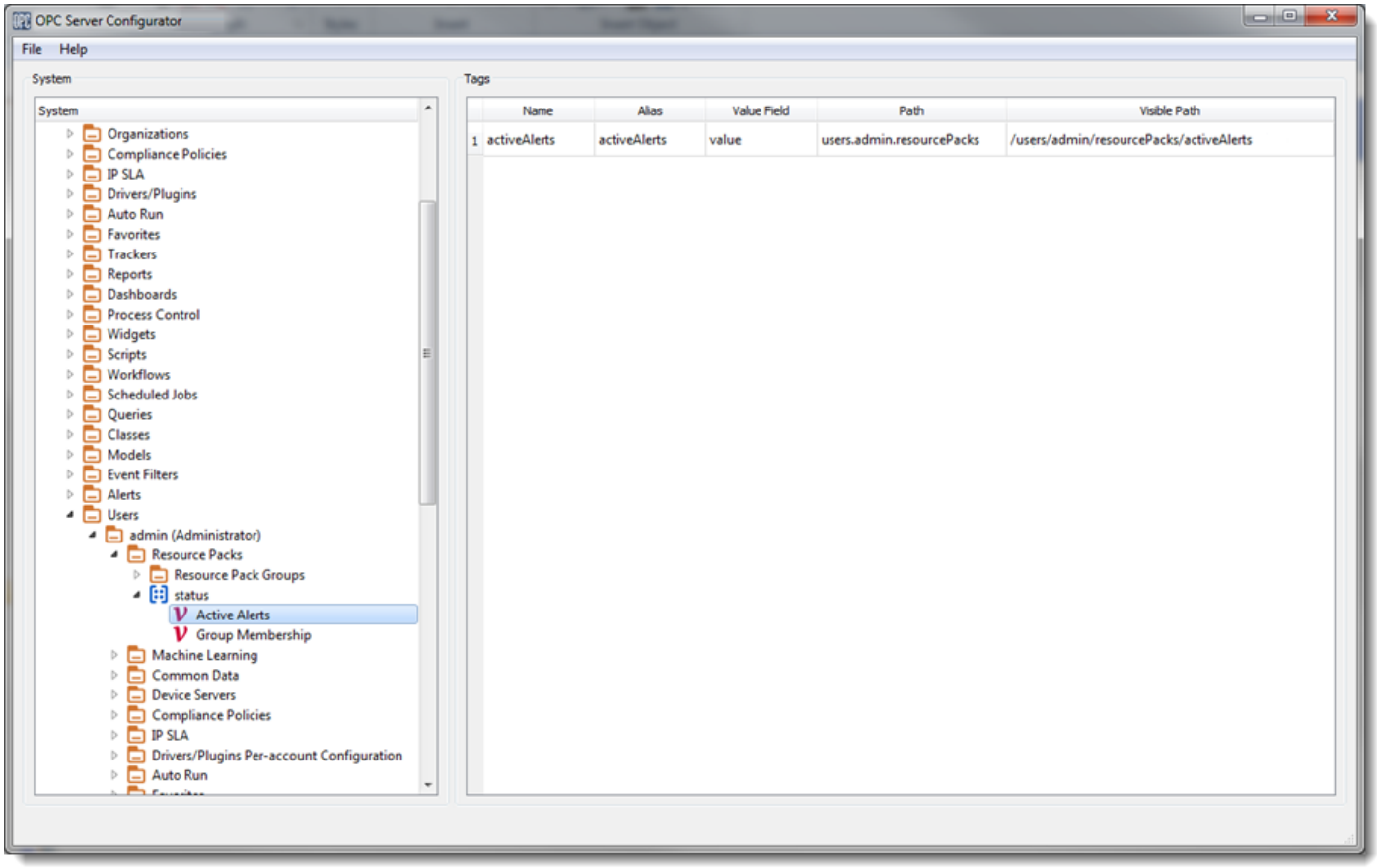
19.2.8.2 Конфигурация сервера

Для начала, давайте примем конфигурацию AtomMind Server за источник данных. После этого нажмите *Start -> All Programs* -> ТВЭЛ *OPC Server* -> ТВЭЛ *OPC Server Configurator*. Выбираем к меню File -> New Connection-> Server, вводим параметры сервера и нажимаем *OK*:



OPC-сервер не поддерживает SSL шифрование. Для корректной работы разрешите небезопасное подключение клиентов к AtomMind Server.

Если соединение прошло успешно, то на левой панели появится дерево контекстов. Перетащите переменные из контекста в конфигурацию OPC-сервера:



Алиас - это имя тега в OPC-сервере. *Value Field* применяется только к переменным в таблице. Скалярное значение выбранного столбца будет передаваться в OPC-сервер. В противном случае, вся таблица конвертируется в строку. Path - путь контекста переменной AtomMind. Данные поля доступны только для чтения. Чтобы удалить тег, нажмите Remove в контекстном меню. Наконец, нажмите пункт меню Register and Save Configuration для сохранения конфигурации и регистрации сервера.

19.2.9 OPC-агент (собственный OPC-клиент)

OPC-агент AtomMind - это Windows-приложение, которое подключается к локальному OPC-серверу, запущенному на той же машине, и раскрывает OPC-данные AtomMind Servery, подключаясь к нему как агент[684].

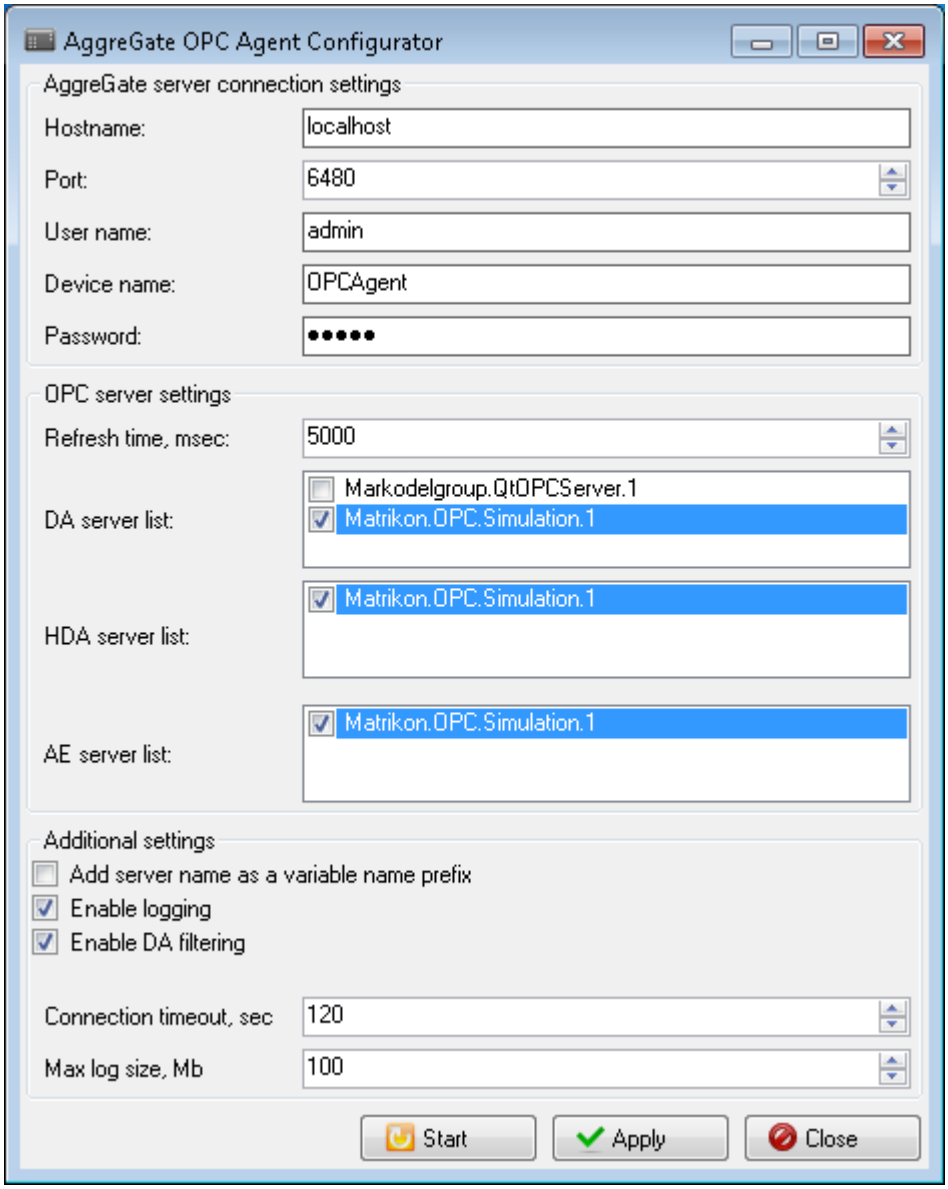
OPC-агент поддерживает спецификации OPC DA, OPC HDA и OPC AE, разрешая AtomMind Servery доступ к текущим и историческим значениям OPC-тегов, а также к OPC-тревогам/событиям, унифицированным с помощью единой модели данных.

Агент можно скачать с вебсайта АО "ТВЭЛ". Он не подлежит отдельному лицензированию, однако OPC-данные, предоставляемые Агентом, учитываются в ограничениях лицензии AtomMind Server.

конфигурирование OPC-агента

Ниже показано, как настроить OPC-агент, а именно:

- настроить параметры подключения AtomMind Server (адрес сервера, порт, имя пользователя, имя устройства типа Агент и пароль)
- выбрать серверы OPC DA/HDA и AE, чьи данные будут раскрыты AtomMind Servery
- указать период обновления значений
- настроить дополнительные опции, такие как журналирование, фильтрация тегов и пр.



19.2.10 Техобслуживание

AtomMind SCADA/HMI включает модуль *Техобслуживание* (плагин)207, который упрощает отслеживание техосблуживания ваших аппартаных активов. Целями этого модуля являются:

- Обеспечение регулярного техобслуживания активов и компонентов процессов с целью избежания простоя
- Обеспечение оптимального срока службы активов благодаря регулярному техобслуживанию

Выполнение техобслуживания

1) Прежде всего, необходимо запустить действие **Запланировать техобслуживание**, позволяющее запланировать823 периодические задачи по техобслуживанию. Это действие доступно в контексте Запланированные задачи1586 ( ), если установлен модуль техобслуживания.

Действие Запланировать техобслуживание также доступно в таблице Избранное2186.

2) После запуска действия Запланировать техосблуживание необходимо ввести параметр *Акт техобслуживания*, в частности:

   a) **Имя действия техобслуживания** (который будет соответствовать имени контекста задач1586 техобслуживания)

b) **Описание действия техобслуживания** (будет соответствовать описанию контекста задач техобслуживания)

c) **Предмет техобслуживания**, т.е. устройство или компонент для обслуживания

d) Условия **Регламента техобслуживания**

3) Когда заканчивается создание задачи по техобслуживанию, ее можно найти в **Запланированные задачи => Запланированные группы задач => Техобслуживание**. Для этой задачи необходимо запустить действие **Конфигурировать** ( ) и определить расписание техобслуживания. Доступны два типа расписаний: простое 828 (с фиксированныйм периодом) и с расширенными возможностями 828.

4) Каждый раз при выполнении задачи по техобслуживанию оно вызывает тревогу 779 **Техобслуживание**, которая хранится в базе данных для индикации того, что должно быть совершено действие по техобслуживанию. Если существуют ожидающие (неподтвержденные) задачи по техобслуживанию, эта тревога переключится в активное состояние (с такой иконкой). Наконец, если есть более 5 неподтвержденных действий по техобслуживанию или, по крайне мере, одно из них не было подтверждено дольше одного дня, тревога по техобслуживанию будет эскалироваться 805 (это действие представлено иконкой). Тревоги эскалации настраиваются 799.

5) Как только техобслудивание физически завершено, системный оператор должен подтвердить 806 этот экземпляр тревоги и, возможно, также ввести комментарии о прогрессе/результате техобслуживания. Существует два способа подтверждения тревоги техобслуживания:

a) Запустить действие Управление неподтвержденными тревогами 1454 из контекста тревоги ( ), щелкнуть правой кнопкой на экземпляре тревоги, выбрать **Подтвердить** из контекстного меню и ввести текст подтверждения.

b) Активировать фильтр событий 762 **Действия техобслуживания** (что также является частью модуля Техобслуживание) в историческом разделе Журнала событий 398. Этот фильтр показывает все действия по техобслуживанию, но неподтвержденные действия выделяет морковно-красным цветом. Для подтверждения любого действия щелкните по нему правой кнопкой мыши, выберите **Подтвердить** из контекстного меню, затем введите текст подтверждения.

6) Наконец, чтобы просмотреть историю прошлых действий по техобслуживанию, используйте отчет 928 **Обзор действий техобслуживания**.

19.2.11 Демонстрационные мнемосхемы

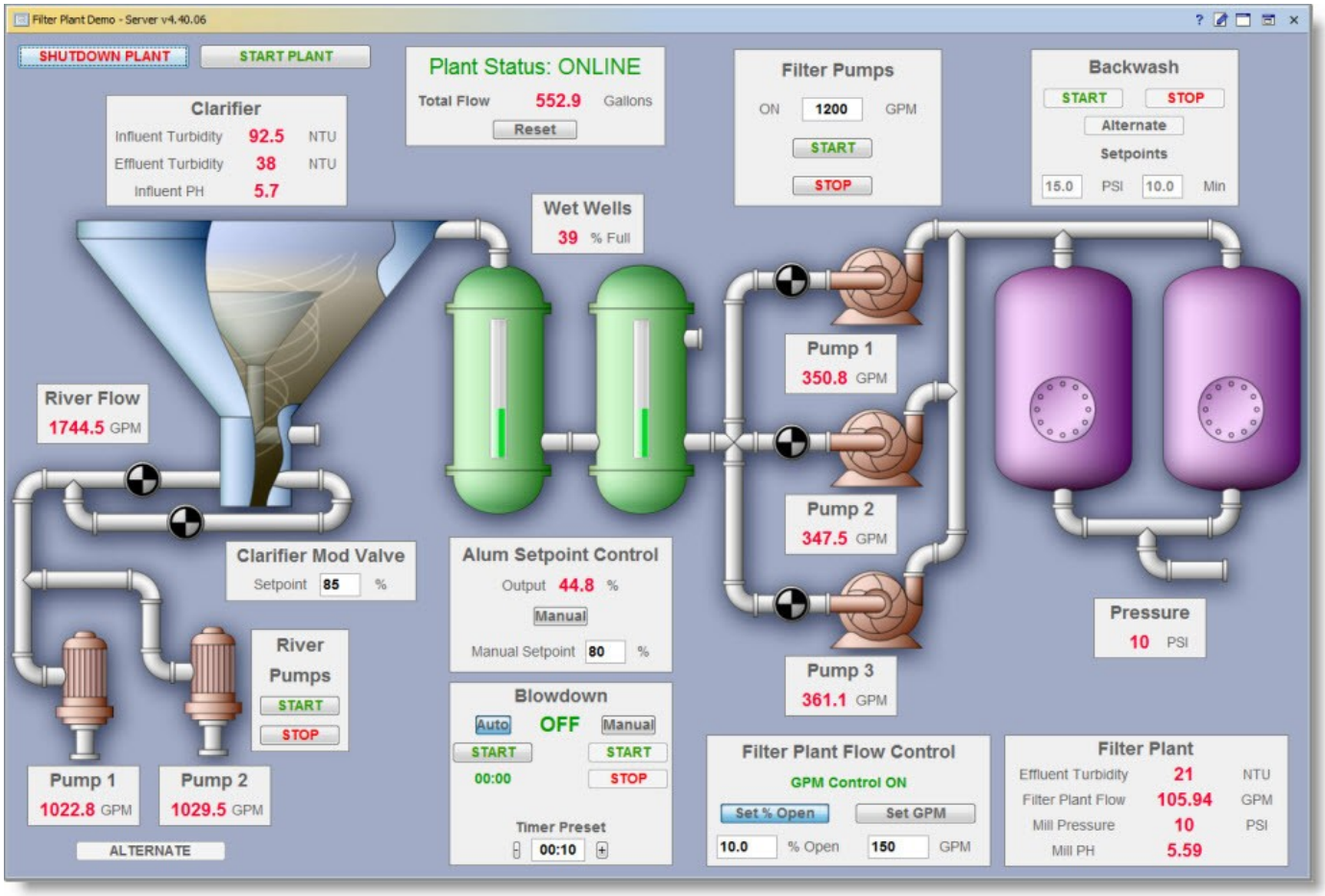
Дистрибутив AtomMind SCADA/HMI включает несколько демо-виджетов HMI. Некоторые из них довольно простые, иллюстрирующие базовые концепции создания HMI (компоненты 955, свойства 1274, привязки 1295 и т.д.). Другие демо-виджеты гораздо более сложные -- такие демо-виджеты, на самом деле, являются нормальными HMI, которые можно использовать для контроля всего промышленного процесса. Разница лишь в том, что эти демо-виджеты не подключены к реальному "железу". Однако технология нормальзации данных устройств в AtomMind делает возможным переключение подобного HMI на реальное "железо". В большинстве случаев просто необходимо исправить несколько привязок в пределах самого HMI.

Все демо-виджеты расположены в **Виджеты => Группы виджетов => Демо-виджеты**. Вот список таких виджетов:

- **Демо компонентов**. Предоставляет обзор доступных компонентов 955 виджетов и их свойств.
- **Демо привязок**. Иллюстрирует, как различные части привязок виджетов 1295 можно использовать для решения разнообразных задач по анимации HMI.
- **Демо графиков**. Содержит несколько вкладок, которые показывают доступные типы графиков 1051 и настроек.
- **Демо SVG**. Объясняет функциональность динамического компонента векторное рисование 995.
- **Демо скриптов**. Делает обзор использования скриптов виджетов 1308.
- **Демо фильтровальной установки**. Полномасштабный демо HMI, эмулирующий фильтровальную установку 1983.
- **Демо линии бутылочного розлива**. Полномасштабный HMI, эмулирующий бутылочный розлив.
- **Демо бутылочного конвейера**. Подвиджет 1049, используемый для реализации и анимирования конвейерной ленты в пределах демо Бутылочный розлив. Не должен использоваться отдельно.

19.2.11.1 Демонстрация работы фильтровальной установки

AtomMind SCADA/HMI включает демо-HMI **Фильтровальная установка**. Это довольный сложный HMI, который можно менять, запускать и использовать с целью изучения:



Описание демо-виджета Фильтровальная установка

Демо-виджет Фильтровальная установка эмулирует функционирование простой фильтровальной установки, оборудованной двумя уровнями насосов, водоприемных колодцев и т.д. Однако это демо не зависит от каких-либо реальных или даже виртуальных 654 устройств.

Сложная логика фильтровальной установки, работа всех ее компонентов эмулируется привязками 1295 виджетов с целью отображения их производительности и гибкости. Демо-виджет Фильтровальная установка включает более пятидесяти привязок, симулирующих потоки воды, управление насосом, различные измерения и действия по управлению. Все эти привязки можно легко анализировать или исправлять в Редакторе виджетов 423.

Использование плавающей панели

Демо-виджет Фильтровальная установка иллюстрирует другую важную концепцию HMI-виджета: использование плавающих панелей с компоновкой 950 "сетка" для создания "контрольных форм" различного оборудования. Эти панели можно легко перемещать в главном окне, имеющем абсолютную компоновку. Однако у каждой панели есть компоновка "сетка", которая гораздо больше подходит для создания форм и интерфейсов ввода данных.

Эти плавающие панели упрощают управление более 200 HMI-компонентами фильтровальной установки при помощи их визуальной и логической группировки. Компоновку каждой панели можно редактировать отдельно.

19.2.12 Управление процессами

Управление процессами включает в себя четыре языка программирования:

- SFC - Последовательностные функциональные диаграммы;
- FBD - Функциональные блоковые диаграммы;
- LD - Релейно-Контактные Схемы;

- ST - Структурированный текст.

Отличительной особенностью является то, что данная реализация языков поддерживает выражения AtomMind 112 и стандартный элемент данных Data Table 49. Это означает, что языки являются частью единой модели данных AtomMind, и это позволяет взаимодействовать с его инфраструктурой. Взаимодействие выражений AtomMind 112 и языков наделяет пользователя мощными и широкими возможностями реализации алгоритмов на AtomMind.

Управление процессами разработано на основе стандарта IEC 61131-3. Данный раздел не является руководством по стандарту IEC 61131-3. Предполагается, что вы уже знакомы со стандартом IEC 61131-3. Мы не пытаемся усложнить вашу задачу - стандарт IEC 61131-3 представляет собой объемную тему, по которой написано достаточное количество книг и статей. Поэтому, если вы не знакомы с IEC 61131-3, вам необходимо получить общее представление о предмете. Как только вы этого добьетесь, возвращайтесь к данному разделу, чтобы узнать, как мы его выполняем в AtomMind . Далее мы приводим перечень источников, где вы можете найти более подробную информацию о стандарте IEC 61131-3.

Источники

Для получения более подробной информации о стандарте IEC 61131-3 перейдите по следующим ссылкам:

https://ru.wikipedia.org/wiki/IEC_61131-3

https://webstore.iec.ch/publication/4552

19.2.12.1 Общие элементы

Прежде чем переходить к непосредственно описанию языков, необходимо познакомиться с общими элементами этих языков. Общие элементы служат единым фундаментом, позволяющим объединить многоязычные компоненты в одном проекте. В главе будут рассмотрены данные, переменные, основные компоненты, форматы их представления и наиболее общие приемы и тонкости работы с ними.

19.2.12.1.1 Литералы

ЧИСЛОВЫЕ ЛИТЕРАЛЫ

Тип	Пример	Описание
Целочисленный литерал	-12, 0, 123_4, +9854	
Действительный литерал	0.0, 0.45, 3.1415_59	
Действительный литерал с экспонентой	-1.34E-12, -1.34e-12, 1.0E+6, 1.0e+6, 1.234E6, 1.234e6	
Двоичный литерал	2#1111_1111	255 в десятичной форме
Восьмеричный литерал	8#377	255 в десятичной
Шестнадцатеричный литерал	16#FF или 16#ff	255 в десятичной форме
Логические ноль и единица	FALSE TRUE	
Типизированный литерал	INT#-123	Представление INT десятичного значения -123
	INT#16#7FFF	Представление INT десятичного значения 32767
	WORD#16#AFF	Представление WORD десятичного значения 0AFF
	WORD#1234	Представление WORD десятичного значения 1234
	UINT#16#89AF	Представление UINT шестнадцатеричного значения 89AF
	BOOL#0	

	BOOL#1	
	BOOL#FALSE	
	BOOL#TRUE	

Символьно-строковые литералы поддерживает **только** двухбайтовые кодированные символы.

СИМВОЛЬНО-СТРОКОВЫЕ ЛИТЕРАЛЫ

Тип	Пример	Описание
Строка	"OK", 'OK'	
Символ	"O", 'O'	
Типизированная строка	WSTRING#"OK"	Типизированная двухбайтовая строка с использованием двойных кавычек.
Типизированная символ	WCHAR#"O"	Типизированный двухбайтовый символ с использованием двойных кавычек.
Типизированная строка	WSTRING#'OK'	Типизированная двухбайтовая строка с использованием одинарных кавычек.
Типизированная символ	WCHAR#'O'	Типизированный двухбайтовый символ с использованием одинарных кавычек.

Литералы продолжительности времени **не поддерживают** микросекунды и наносекунды.

ЛИТЕРАЛЫ ПРОДОЛЖИТЕЛЬНОСТИ ВРЕМЕНИ

Тип	Пример
Короткий префикс без символов подчеркивания.	T#14ms, T#-14ms, T#14m, T#14h t#14d t#25h15m, t#5d14h12m18s3ms, t#12h4m34ms.
Длинный префикс без символов подчеркивания.	TIME#14ms, TIME#-14ms, time#14s.

ЛИТЕРАЛЫ ДАТЫ И ВРЕМЕНИ СУТОК

Тип	Пример
Дата	DATE#1984-06-25, date#2010-09-22, D#1984-06-25
Время суток	TIME_OF_DAY#15:36:55.36, TOD#15:36:55.36
Дата и время	DATE_AND_TIME#1984-06-25-15:36:55.360, DT#1984-06-25-15:36:55.360

19.2.12.1.2 Типы данных

Поддерживаются следующие типы данных:

Тип	Описание
BOOL	Логический. Соответствует типу Boolean 113.
SINT	Короткое целое.
INT	Целое
DINT	Двойное целое. Соответствует типу Integer 113.

LINT	Длинное целое. Соответствует типу Long 113.
USINT	Короткое целое без знака.
UINT	Целое без знака.
UDINT	Двойное целое без знака.
ULINT	Длинное целое без знака.
REAL	Действительные числа. Соответствует типу Float 113.
LREAL	Длинные действительные числа. Соответствует типу Double 113.
BYTE	Битовая строка длины 8.
WORD	Битовая строка длины 16.
DWORD	Битовая строка длины 32.
LWORD	Битовая строка длины 64.
TIME_OF_DAY или TOD	Время суток.
TIME	Продолжительность времени.
DATE	Соответствует типу Date 113.
DATE_AND_TIME или DT	Время и дата суток.
WSTRING	Соответствует типу String 113.
WCHAR	Двухбайтовый символ.
OBJECT	Соответствует типу Object 113.
DATATABLE	Соответствует типу Data Table 49.

LTIME, LDATE, LDATE_AND_TIME, LTIME_OF_DAY не поддерживаются.

19.2.12.1.3 Переменные

ОБЪЯВЛЕНИЕ

Каждое объявление программного компонента (POU), то есть функции, функционального блока или программы, начинается с описания секций переменных. Переменные объявляются внутри секции переменных.

Пример объявления секции переменных:

```
VAR
    isEnabled : BOOL;
    bits : ARRAY[0..7] OF BOOL;
    data : INT := 100;
END_VAR
```

ПОДДЕРЖИВАЕМЫЕ СЕКЦИИ ПЕРЕМЕННЫХ

Тип	Пример
VAR ... END_VAR	Доступны только внутри компонента, вне компонента доступа нет. Могут иметь начальные значения. Для функций локальные переменные размещаются в динамической памяти. По окончании работы функции

	память освобождается. В программах и экземплярах функциональных блоков переменные VAR сохраняют свои значения между вызовами программ и экземпляров. В графическом представлении компонента локальные переменные не отображаются.
VAR_INPUT ... END_VAR	Передаются по значению путем копирования. При вызове блока такой переменной можно присвоить значение другой переменной (совместимого типа) или выражения. Любые изменения такой переменной внутри компонента никак не отображаются на данные вызывающего компонента. Применяется в любых компонентах. Могут иметь значения по умолчанию. Отражаются в графическом представлении с левой стороны компонента.
VAR_IN_OUT ... END_VAR	Этот параметр одновременно является входом и выходом. Передача переменной экземпляру блока выполняется по ссылке. Это означает, что внешняя переменная как бы сама работает внутри блока на правах внутренней переменной. В компонент передается только адрес ее расположения в памяти данных. Для переменной VAR_IN_OUT нельзя: • присваивать начальное значение; • присваивать константу как актуальный параметр. Присваивание внешней переменной для VAR_IN_OUT можно производить только при вызове блока. Важнейшим свойством VAR_IN_OUT является отсутствие копирования внешних данных. Параметры VAR INPUT и VAR_OUTPUT могут оперировать с массивами и структурами, но всякий раз при обращении к компоненту будет происходить полное копирование данных. Это может отнимать много времени. Присваивание одного массива другому для VAR_IN_OUT означает фактически переключение компонента с одного массива на другой. Локальная копия данных в этом случае не создается. Параметры VAR_IN_OUT нарушают идеологию независимости компонентов. Правильный компонент не должен иметь возможности испортить чужую память. Поэтому применять их нужно очень аккуратно и только в случаях, когда это действительно необходимо.
VAR_OUTPUT ... END_VAR	Отражают результаты работы компонента. Передаются из компонента по значению путем копирования. Чтение значения выходов обычно имеет смысл после выполнения блока. Вне компонента параметры VAR_OUTPUT доступны только по чтению. Могут иметь начальные значения. Отражаются в графическом представлении справа.

19.2.12.1.4 Функции

Функции выполняют обработку входных параметров и возвращают какое-либо значение. Некоторые функции (такие как **SPLIT_DATE**) не имеют выходных параметров.

Также добавлена поддержка всех функций обработки таблицы данных 124, некоторых функций, относящихся к контексту 124 и функция evaluate 124

ФУНКЦИИ ОБРАБОТКИ ЧИСЕЛ

Функция	Описание	Тип результата

ABS(ANY_NUM IN1)	Возвращает абсолютное значение. Если аргумент не отрицательный, он возвращается. Если же аргумент отрицательный, возвращается отрицание аргумента. Если тип аргумента является Целым или Длинным, возвращаемое значение будет типа Длинное.	ANY_NUM
ADD(ANY_NUM IN1, ANY_NUM IN2,...)	Сложение чисел (то же, что и оператор +).	ANY_NUM
DIV(ANY_NUM IN1, ANY_NUM IN2)	Деление чисел (то же, что и оператор /).	ANY_NUM
SUB(ANY_NUM IN1, ANY_NUM IN2)	Вычитание чисел (то же, что и оператор -).	ANY_NUM
MUL(ANY_NUM IN1, ANY_NUM IN2,...)	Умножения чисел (то же, что и оператор *).	ANY_NUM
MOD(ANY_INT IN1, ANY_INT2)	Остаток по модулю (то же, что и оператор IN1 MOD IN2).	ANY_INT
SQRT(LREAL IN1)	Квадратный корень.	LREAL
LOG(LREAL IN1)	Возвращает натуральный логарифм.	LREAL
LOG10(LREAL IN1)	Возвращает десятичный логарифм.	LREAL
EXP(LREAL IN1)	Возвращает экспоненту.	LREAL
EXPT(LREAL IN1, LREAL IN2)	Возведение IN1 в степень IN2 (то же, что и оператор **).	LREAL
SIN(LREAL IN1)	Синус от входного значения в радианах.	LREAL
COS(LREAL IN1)	Косинус от входного значения в радианах.	LREAL
TAN(LREAL IN1)	Тангенс от входного значения в радианах.	LREAL
ASIN(LREAL IN1)	Возвращает значение арксинуса.	LREAL
ACOS(LREAL IN1)	Возвращает значение арккосинуса.	LREAL
ATAN(LREAL IN1)	Возвращает значение арктангенса.	LREAL
ATAN2(LREAL IN1, LREAL IN2)	Возвращает угол между положительным направлением оси X и точкой, заданной координатами (x,y).	LREAL

ПОБИТОВЫЕ ЛОГИЧЕСКИЕ ФУНКЦИИ

Функция	**Описание**	**Тип результата**
AND(ANY_BIT IN1, ANY_BIT IN2, ...)	Побитовое И (то же, что и оператор AND или &).	ANY_BIT
OR(ANY_BIT IN1, ANY_BIT IN2, ...)	Побитовое ИЛИ (то же, что и оператор OR).	ANY_BIT
XOR(ANY_BIT IN1, ANY_BIT IN2, ...)	Исключающие ИЛИ (то же, что и оператор XOR).	ANY_BIT
NOT(ANY_BIT IN1)	Побитовое НЕ (то же, что и оператор NOT).	ANY_BIT

ФУНКЦИИ ВЫБОРА

Функция	**Описание**	**Тип результата**
LIMIT(ANY_ELEMENTARY MN, ANY_ELEMENTARY IN1, ANY_ELEMENTARY MX)	Возвращает MN, если IN1 меньше MN. Возвращает MX, если IN1 больше MX, иначе возвращает IN1.	ANY_ELEMENTARY
MAX(ANY_ELEMENTARY IN1, ANY_ELEMENTARY IN2, ...)	Возвращает большее из двух значений.	ANY_ELEMENTARY
MIN(ANY_ELEMENTARY IN1, ANY_ELEMENTARY IN2, ...)	Возвращает меньшее из двух значений.	ANY_ELEMENTARY
MUX(ANY_INT K, ANY_ELEMENTARY IN1, ...)	Выбирает одну из N входных переменных в зависимости от входной переменной K.	ANY_ELEMENTARY

SEL(BOOL G, ANY IN1,ANY IN2)	Возвращает IN1 , если G равно FALSE, иначе возвращает IN2.	ANY

ФУНКЦИИ СРАВНЕНИЯ

Функция	Описание	Тип результата
SHL(ANY_BIT IN1, ANY_INT IN2)	Сдвиг влево на IN2 бит, биты справа заполняются нулями.	ANY_BIT
SHR(ANY_BIT IN1, ANY_INT IN2)	Сдвиг вправо на IN2 бит, биты слева заполняются нулями.	ANY_BIT
ROL(ANY_BIT IN1, ANY_INT IN2)	Циклический сдвиг влево на IN2 бит.	ANY_BIT
ROR(ANY_BIT IN1, ANY_INT IN2)	Циклический сдвиг вправо на IN2 бит.	ANY_BIT

ФУНКЦИИ БИТОВОГО СДВИГА

Функция	Описание	Тип результата
SHL(ANY_BIT IN1, ANY_INT IN2)	Сдвиг влево на IN2 бит, биты справа заполняются нулями.	ANY_BIT
SHR(ANY_BIT IN1, ANY_INT IN2)	Сдвиг вправо на IN2 бит, биты слева заполняются нулями.	ANY_BIT
ROL(ANY_BIT IN1, ANY_INT IN2)	Циклический сдвиг влево на IN2 бит.	ANY_BIT
ROR(ANY_BIT IN1, ANY_INT IN2)	Циклический сдвиг вправо на IN2 бит.	ANY_BIT

ФУНКЦИИ ОБРАБОТКИ СТРОК

Функция	Описание	Тип результата
LEN(ANY_STRING IN1)	Возвращает длину *строки*.	INT
LEFT(ANY_STRING IN1, ANY_INT L)	Возвращает подстроку начиная с первого символа и длины L символов.	ANY_STRING
RIGHT(ANY_STRING IN1, ANY_INT L)	Возвращает подстроку начиная с последнего символа и длины L символов.	ANY_STRING
MIDDLE(ANY_STRING IN1, ANY_INT L, ANY_INT P)	Возвращает подстроку начиная с P позиции и длины L символов.	ANY_STRING
CONCAT(ANY_CHARS IN1, ANY_CHARS IN2, ...)	Конкатенация двух строк.	ANY_STRING
INSERT(ANY_STRING IN1, ABY_CHARS IN2, ANY_INT P)	Вставить IN2 в строку IN1 после P-й позиции символа.	ANY_STRING
REPLACE(ANY_STRING IN1, ANY_CHARS IN2, ANY_INT L, ANY_INT P)	Заменить L символов строки IN1 строкой IN2, начиная в P-й позиции символа.	ANY_STRING
FIND(ANY_STRING IN1, ANY_CHARS IN2)	Найти позицию символа первого вхождения IN2 в строку IN1. Если вхождения строки IN2 не обнаружены, то вернет 0.	INT

ФУНКЦИИ ОБРАБОТКИ ДАТЫ/ВРЕМЕНИ

Функция	Описание	Тип результата
ADD_DT_TIME(DT IN1, TIME IN2)	Сложение времени и даты суток (то же, что и оператор +).	DT
ADD_TIME(TIME IN1, TIME IN2)	Сложение продолжительности времени (то же, что и оператор +).	TIME

ADD_TOD_TIME(TOD IN1, TIME IN2)	Сложение времени суток и продолжительности времени (то же, что и оператор +).	TOD
SUB_DATE_DATE(DATE IN1, DATE IN2)	Вычитание двух дат (то же, что и оператор -).	TIME
SUB_DT_DT(DT IN1, DT IN2)	Вычитание двух дат (то же, что и оператор -).	TIME
SUB_DT_TIME(DATE IN1, TIME IN2)	Вычитание двух дат (то же, что и оператор -).	DT
SUB_TIME_TIME(TIME IN1, TIME IN2)	Вычитание двух дат (то же, что и оператор -).	TIME
SUB_TOD_TIME(TOD IN1, DATE IN2)	Вычитание двух дат (то же, что и оператор -).	TOD
SUB_TOD_TOD(TOD IN1, TOD IN2)	Вычитание двух дат (то же, что и оператор -).	TIME
MUL_TIME(TIME IN1, ANY_NUM IN2)	Умножения продолжительности времени на число (то же, что и оператор *).	TIME
DIV_TIME(TIME IN1, ANY_INT IN2)	Деление продолжительности времени на число (то же, что и оператор /).	TIME
CONCAT_DATE_TOD(DATE IN1, TOD IN2)	Конкатенация даты и времени.	DT
CONCAT_DATE(ANY_INT YEAR, ANY_INT MONTH, ANY_INT DAY)	Конкатенация даты и времени.	DATE
CONCAT_TOD(ANY_INT HOUR, ANY_INT MINUTE, ANY_INT SECOND, ANY_INT MILLISECOND)	Конкатенация времени.	TOD
CONCAT_DT(ANY_INT YEAR, ANY_INT MONTH, ANY_INT DAY,ANY_INT HOUR, ANY_INT MINUTE, ANY_INT SECOND, ANY_INT MILLISECOND)	Конкатенация даты и времени.	DT
SPLIT_DATE(DATE IN1, DINT YEAR, DINT MONTH, DINT DAY)	Разбиение даты.	
SPLIT_DT(DT IN1, DINT YEAR, DINT MONTH, DINT DAY,DINT HOUR, DINT MINUTE, DINT SECOND, DINT MILLISECOND)	Разбиение даты и времени.	
SPLIT_TOD(DINT HOUR, DINT MINUTE, DINT SECOND, DINT MILLISECOND)	Разбиение времени.	
DAY_OF_WEEK(DATE IN1)	Возвращает день недели.	DINT

ФУНКЦИИ ПРОВЕРКИ ТИПОВ

Функция	**Описание**	**Тип результата**
IS_DATATABLE(OBJECT IN1)	Возвращает TRUE, если аргумент является DATATABLE.	BOOL
IS_LINT(OBJECT IN1)	Возвращает TRUE, если аргумент является LINT.	BOOL
IS_DINT(OBJECT IN1)	Возвращает TRUE, если аргумент является DINT.	BOOL
IS_BOOL(OBJECT IN1)	Возвращает TRUE, если аргумент является BOOL.	BOOL
IS_LREAL(OBJECT IN1)	Возвращает TRUE, если аргумент является LREAL.	BOOL
IS_REAL(OBJECT IN1)	Возвращает TRUE, если аргумент является REAL.	BOOL
IS_WSTRING(OBJECT IN1)	Возвращает TRUE, если аргумент является WSTRING.	BOOL
IS_DT(OBJECT IN1)	Возвращает TRUE, если аргумент является DT.	BOOL

ФУНКЦИИ ПРИВЕДЕНИЯ ТИПОВ

Данная таблица отображает функции приведения типов. Каждая функция называется как ИСХОДНЫЙ_ТИП_ТО_ЦЕЛЕВОЙ_ТИП (например, функция приведения типа LREAL в BOOL , будет LREAL_TO_BOOL).

Исходный тип	**Целевой тип**																					
	real		integer				unsigned				bit					date & times				char		other	
	LREAL	REAL	LINT	DINT	INT	SINT	ULINT	UDINT	UINT	USINT	LWORD	DWORD	WORD	BYTE	BOOL	TIME	DT	DATE	TOD	WSTRING	WCHAR	DATATABLE	OBJECT
LREAL		X	X	X	X	X	X	X	X	X	X	X	X	X	X					X		X	
REAL			X	X	X	X	X	X	X	X					X					X		X	
LINT	X	X		X	X	X	X	X	X	X	X	X	X	X	X					X		X	
DINT		X			X	X	X	X	X	X		X	X	X	X					X		X	
INT						X	X	X	X	X					X					X		X	
SINT							X	X	X	X					X					X		X	
ULINT	X	X	X	X	X	X		X	X	X					X					X		X	
UDINT		X		X	X	X			X	X					X					X		X	
UINT					X	X				X					X					X		X	
USINT						X									X					X		X	
LWORD	X		X	X	X	X	X	X	X	X		X	X	X	X					X		X	
DWORD		X	X	X	X	X	X	X	X	X			X	X	X					X		X	
WORD			X	X	X	X	X	X	X	X				X	X					X		X	
BYTE			X	X	X	X	X	X	X	X					X					X		X	
BOOL			X	X	X	X	X	X	X	X										X		X	
TIME																				X			
DT																		X	X	X			
DATE																				X			
TOD																				X			
WSTRING															X								
WCHAR											X	X	X	X						X			
DATATABLE			X	X	X	X	X	X	X	X					X			X	X	X			
OBJECT	X	X	X	X											X		X			X		X	

ПОЛЬЗОВАТЕЛЬСКИЕ ФУНКЦИИ

Тип функций (тип возвращаемого значения) может быть любым из числа стандартных типов данных или типов, созданных пользователем. Тело функции может быть описано на языках ST или FBD. Использовать SFC нельзя. Для того, чтобы функция имела возвращаемый тип и ее можно было использовать в выражениях, необходимо в секции VAR_OUTPUT ... END_VAR создать переменную с именем, совпадающим с именем функции (контекста).

Пример: Предположим, что мы хотим создать функцию с именем foo, которая будет возвращать значение типа BOOL. Следовательно, в секцию, необходимо добавить переменную:

```
VAR_OUTPUT
    BOOL : foo;
END_VAR
```

19.2.12.1.5 Функциональные блоки

Функциональный блок - программный компонент, отображающий множество значений входных переменных на множество выходных. После выполнения экземпляра функционального блока все его переменные сохраняются до следующего выполнения. Следовательно, функциональный блок, вызываемый с одними и теме же параметрами, может производить различные выходные значения. Сохраняются все переменные, включая входные и выходные. Так, если мы вызовем экземпляр функционального блока, не определяя значения некоторых входных параметров, он будет использовать ранее установленные значения. Возможность задания переменного числа входных значений заложена по определению и не требует каких-либо усилий. Извне доступны только входы и выходы функционального блока, получить доступ к внутренним переменным блока нельзя.

С позиции *объектно-ориентированного программирования* (ООП) функциональные блоки - это объекты, великолепно реализующие инкапсуляцию, т.е. сокрытие деталей реализации. Объединение кода и данных в "одном флаконе" роднит функциональные блоки с классами ООП. Возможность *наследования* и *полиморфизма,* к сожалению, пока отсутствует.

Прежде чем использовать функциональный блок, необходимо создать его экземпляр. Эта операция аналогична по смыслу объявлению переменной. Описав новый блок, мы фактически создали новый тип данных, подобный структуре. Каждый функциональный блок может иметь любое количество экземпляров. Так, различные экземпляры блока "таймер" совершенно независимы друг от друга. Каждый из них имеет свои настройки и живет собственной жизнью. Каждый экземпляр функционального блока имеет свой собственный идентификатор и свою область в статической памяти данных. Объявление еще одного экземпляра блока приводит к выделению еще одной области в памяти данных. Код, очевидно, как и для функции, остается общим. Экземпляр функционального блока создается в разделе объявлений переменных функции, функционального блока и программы. Как и переменные, он должен получить уникальный идентификатор.

ТАЙМЕРЫ

Функциональный блок	Описание
TP(BOOL IN, TIME PT, BOOL Q, TIME ET)	Импульсный таймер. Запуск таймера происходит по фронту импульса на входе IN. Вход PT задает длительность формируемого импульса. После запуска таймер не реагирует на изменение значения входа IN. Выход ET отсчитывает прошедшее время. При достижении ET значения PT счетчик останавливается, и выход Q сбрасывается в FALSE.
TOF(BOOL IN, TIME PT, BOOL Q, TIME ET)	Таймер с задержкой выключения. По фронту входа IN выход Q устанавливается в TRUE. Сброс счетчика ET и начало отсчета времени происходит по каждому спаду входа IN. Выход Q будет сброшен через заданное PT время после спада входного сигнала. Если во время отсчета вход IN будет установлен в TRUE, то отсчет приостанавливается. Таким образом, выход Q включается по фронту, а выключается логическим нулем продолжительностью не менее PT.
TON(BOOL IN, TIME PT, BOOL Q, TIME ET)	Таймер с задержкой включения. По фронту входа IN выполняется обнуление счетчика и начинается новый отсчет времени. Выход Q будет установлен в TRUE через заданное PT время, если IN будет продолжать оставаться в состоянии TRUE. Спад входа IN останавливает отсчет и сбрасывает выход Q в FALSE. Таким образом, выход Q включается логической единицей продолжительностью не менее PT, а выключается по спаду входа IN.

ТРИГГЕРЫ

Функциональный блок	Описание
SR(BOOL SET1, BOOL RESET, BOOL Q1)	Переключатель с доминантой включения.

RS(BOOL SET, BOOL RESET1, BOOL Q1)	Переключатель с доминантой выключения.

ДЕТЕКТОРЫ ИМПУЛЬСОВ

функциональный блок	**Описание**
R_TRIG(BOOL CLK, BOOL Q)	Генерирует единичный импульс по переднему фронту входного сигнала.
F_TRIG(BOOL CLK, BOOL Q)	Генерирует единичный импульс по заднему фронту входного сигнала.

СЧЕТЧИКИ

Функциональный блок	**Описание**
CTU(BOOL CU, BOOL RESET, LINT PV, BOOL Q, LINT CV, BOOL M)	По каждому фронту на входе CU значение счетчика (выход CV) увеличивается на 1, Выход Q устанавливается в TRUE, когда счетчик достигнет или превысит заданный PV порог. Логическая единица на входе сброса (RESET = TRUE) останавливает счет и обнуляет счетчик (CV := 0).
CTD(BOOL CD, BOOL LOAD, LINT PV, BOOL Q, LINT CV, BOOL M)	По каждому фронту на входе CD счетчик (выход CV) уменьшается на 1. Выход Q устанавливается в TRUE, когда счетчик достигнет нуля. Счетчик CV загружается начальным значением, равным PV по входу LOAD = TRUE.
CTUD(BOOL CU,BOOL CD, BOOL RESET,BOOL LOAD, LINT PV, BOOL QU, BOOL QD, LINT CV, BOOL M, BOOL M2)	По значению входа RESET = TRUE счетчик CV сбрасывается в 0. По значению входа LOAD = TRUE счетчик CV загружается значением, равным PV. По фронту на входе CU счетчик увеличивается на 1. По фронту на входе CD счетчик уменьшается на 1 (до 0). Выход QU равен TRUE, если CV >= PV, иначе FALSE. Выход QD равен TRUE, если CV = 0, иначе FALSE.

19.2.12.1.6 Программы

Программа — глобальный программный элемент, отображающий множество значений входных параметров на множество выходных. Программа очень похожа на функциональный блок. Из всех программных компонентов программа является самой крупной. При помощи программ определяется верхний уровень проекта. Программы являются глобальными компонентами.

Обращение к переменным и вызов программы ничем не отличается от работы с экземпляром функционального блока. Правильный с позиций стандарта проект должен включать одну или несколько программ, ассоциированных с задачами. Число функций и функциональных блоков, как правило, значительно больше.

19.2.12.1.7 Комментарии

Выражения могут включать комментарии в одну и более строк.

Комментарий в одну строку начинается с последовательности символов // и продолжается до конца следующей строки.

Вот пример комментария в одну строку:

```
temperature > 50 // Low threshold
AND
temperature < 10 // High threshold
```

Комментарий в несколько строк начинается с последовательности символов /* и заканчивается последовательностью */ или (* *)

Вот пример комментария в несколько строк:

```
temperature > 50 /* Low threshold */ AND temperature < 10 /* High threshold */
temperature > 50 (* Low threshold *) AND temperature < 10 (* High threshold *)
```

19.2.12.2 Структурированный текст (ST)

Язык ST - это язык высокого уровня. Синтаксически ST представляет собой несколько адаптированный язык Паскаль. Вместо процедур Паскаля в ST используются компоненты программ стандарта. Для специалистов, знакомых с языком C, освоение ST также не вызовет никаких сложностей. В качестве иллюстрации сравним эквивалентные программы на языках ST и C:

ST:

```
WHILE counter <> 0 DO
   counter := counter - 1;
   var1 := var1 * 2;
   IF var1 > 100 THEN
      var1 := 1;
      var2 := var2 + 1;
    END_IF
END WHILE
```

C:

```
while (counter != 0){
   counter--;
   var1 *=  2;
   if(var1 > 100){
      var1 = 1;
      var2++;
   }
}
```

Язык ST по умолчанию предлагается для описания действий и условий переходов SFC. Это действительно максимально мощный тандем, позволяющий эффективно решать любые задачи.

19.2.12.2.1 Условные операторы

УСЛОВНЫЙ ОПЕРАТОР IF

Оператор выбора позволяет выполнить различные группы выражений в зависимости от условий, выраженных логическими выражениями.

Полный синтаксис оператора IF (if) выглядит так:

```
IF <Boolean expression IF>
THEN
    <expression IF>;
[
ELSIF <Boolean expression ELSEIF 1>
THEN
    < expression ELSEIF 1> ;
ELSIF <Boolean expression ELSEIF n>
THEN
    < expression ELSEIF n> ;
ELSE
    < expression ELSE> ;
]
END_IF
```

Если <Boolean expression IF> TRUE, то выполняются выражения первой группы — <expression IF> . Прочие выражения пропускаются, альтернативные условия не проверяются. Часть конструкции в квадратных скобках является необязательной и может отсутствовать. Если <Boolean expression IF> FALSE, то одно за другим проверяются условия ELSIF. Первое истинное условие приведет к выполнению соответствующей группы выражений. Прочие условия ELSIF анализироваться не будут. Групп ELSIF может быть несколько или не быть совсем. Если все логические выражения дали ложный результат, то выполняются выражения группы ELSE, если, она есть. Если группы ELSE нет, то не выполняется ничего.

В простейшем случае оператор IF содержит только одно условие:

```
IF bReset THEN
    iVar1 := 1;
    iVar2 := 0;
END_IF
```

На первый взгляд конструкция IF с несколькими группами ELSIF выглядит сложной, но на самом деле оказывается достаточно выразительной:

```
IF bReset THEN
    iVar1 := 1;
ELSIF byLeft < 16 THEN
    iVar1 := 2;
ELSIF byLeft < 32 THEN
    iVar1 := 3;
ELSIF byLeft < 64 THEN
    iVar1 := 4;
ELSE
    bReset ;= TRUE;
END_IF
```

ОПЕРАТОР МНОЖЕСТВЕННОГО ВЫБОРА CASE

Оператор множественного выбора CASE позволяет выполнить различные группы выражений в зависимости от значения одной целочисленной переменной или выражения.

Синтаксис:

Syntax:

```
CASE <integer expression> OF
<value 1>:
    <expression 1>:
<value 2> , value 3> :
    <expression 3> ;
<value 4>..value 5> :
    <expression 4> ;
    ...
[
ELSE
    <expression ELSE>;
]
END CASE
```

Если значение выражения совпадает с заданной константой, то выполняется соответствующая группа выражений. Прочие условия не анализируются(<value 1>: <expression 1> ;).

Если несколько значений констант должны соответствовать одной группе выражений, их можно перечислить через запятую (<value 2> , <value 3> : <value 3> ;).

Диапазон значений можно определить через двоеточие (<value 4>..<value 5> : <expression 4> ;).

Группа выражений ELSE является необязательной. Она выполняется при несовпадении ни одного из условий (<expression ELSE> ;).

Пример:

```
CASE byLeft/2 OF
0,127:
    bReset := TRUE;
    Varl :=0;
16..24:
    Varl := 1;
ELSE
    Varl := 2;
END_CASE
```

19.2.12.2.2 Циклы

ЦИКЛЫ WHILE И REPEAT

Циклы WHILE и REPEAT обеспечивают повторение группы выражений, пока верно условное логическое выражение. Если условное выражение всегда истинно, то цикл становится бесконечным.

Синтаксис:

```
WHILE <Conditional Boolean expression> DO
   <Expressions – loop body>
END_WHILE
```

Условие в цикле WHILE проверяется до начала цикла. Если логическое выражение изначально имеет значение FALSE, тело цикла не будет выполнено ни разу.

Синтаксис:

```
REPEAT
    <Expressions – loop body >
UNTIL <Conditional Boolean expression>
END_REPEAT
```

Условие в цикле REPEAT проверяется после выполнения тела цикла. Если логическое выражение изначально имеет значение FALSE, тело цикла будет выполнено один раз. Правильно построенный цикл WHILE или REPEAT обязательно должен изменять переменные, составляющие условие окончания в теле цикла, постепенно приближаясь к условию завершения. Если этого не сделать, цикл не закончится никогда.

Пример:

```
iMax := 5+x;
iPoly := 2*x*x + 1;
WHILE ci < iMax DO
    Var := Varl + iPoly;
    ci := ci + 1;
END_WHILE
```

ЦИКЛ FOR

Цикл FOR обеспечивает заданное количество повторений группы выражений.

Синтаксис:

```
FOR <Integer counter> := <Initial value> TO <End value> [BY <Step>] DO
    <Expressions – loop body>
END FOR
```

Перед выполнением цикла счетчик получает начальное значение. Далее тело цикла повторяется, пока значение счетчика не превысит конечного значения. Счетчик увеличивается в каждом цикле. Начальное и конечное значения и шаг могут быть как константами, так и выражениями. Счетчик изменяется после выполнения тела цикла. Поэтому если задать конечное значение меньшее начального, то при положительном приращении цикл не будет выполнен ни разу. При одинаковых начальном и конечном значениях тело цикла будет выполнено один раз. Часть конструкции BY в скобках необязательна, она определяет шаг приращения счетчика. По умолчанию счетчик увеличивается на единицу в каждой итерации. В качестве счетчика можно использовать переменную любого целого типа.

Пример:

```
Var1 := 0;
FOR cw := 1 TO 10 DO
    Var1 := Var1 + 1;
END_FOR
```

19.2.12.2.3 Операторы прерывания итераций

ОПЕРАТОРЫ EXIT, RETURN И CONTINUE

Оператор EXIT, помещенный в теле циклов WHILE, REPEAT и FOR, приводит к немедленному окончанию цикла. Хороший стиль программирования призывает избегать такого приема, но иногда он весьма удобен. Рассмотрим, например, поиск элемента массива с определенным значением x.

Проще всего организовать линейный перебор при помощи цикла FOR:

```
bObtained:= FALSE;
FOR cN := 1 TO MaxIndex DO
   IF x = aX[cN] THEN
      Index := cN;
      bObtained := TRUE;
      EXIT;
   END_IF
END_FOR
```

Для вложенного цикла оператор EXIT завершает только "свой" цикл, внешний цикл будет продолжать работу.

Оператор RETURN осуществляет немедленный возврат из компонента. Это единственный способ прервать вложенные итерации без введения дополнительных проверок условий. Но не стоит им злоупотреблять. Поскольку в тексте компонента, имеющего, например, 50 выходов, разобраться весьма не просто.

Оператор CONTINUE приводит к немедленному переходу на следующую итерацию цикла, опуская следующие за CONTINUE выражения.

Операторы EXIT и CONTINUE могут использоваться только внутри циклов WHILE, REPEAT и FOR.

19.2.12.3 Последовательные функциональные схемы (SFC)

В Управление процессами SFC (Sequential Function Chart) диаграммы стоят особняком, а точнее, выше по отношению к остальным трем языкам. Диаграммы SFC являются высокоуровневым графическим инструментом. Благодаря SFC идея превращения модели системы в законченную программу стала реальностью. В отличие от применения вспомогательных средств моделирования SFC дает действующую программу. SFC имеют выраженную направленность сверху вниз и отражаются прямыми линиями. Задать несколько стартовых шагов в SFC нельзя, только один шаг диаграммы является начальным. Графическая диаграмма SFC состоит из шагов и переходов между ними. Разрешение перехода определяется условием, с шагом связаны определенные действия. Описания действий выполняются на языке ST. Сам SFC не содержит каких-либо управляющих команд.

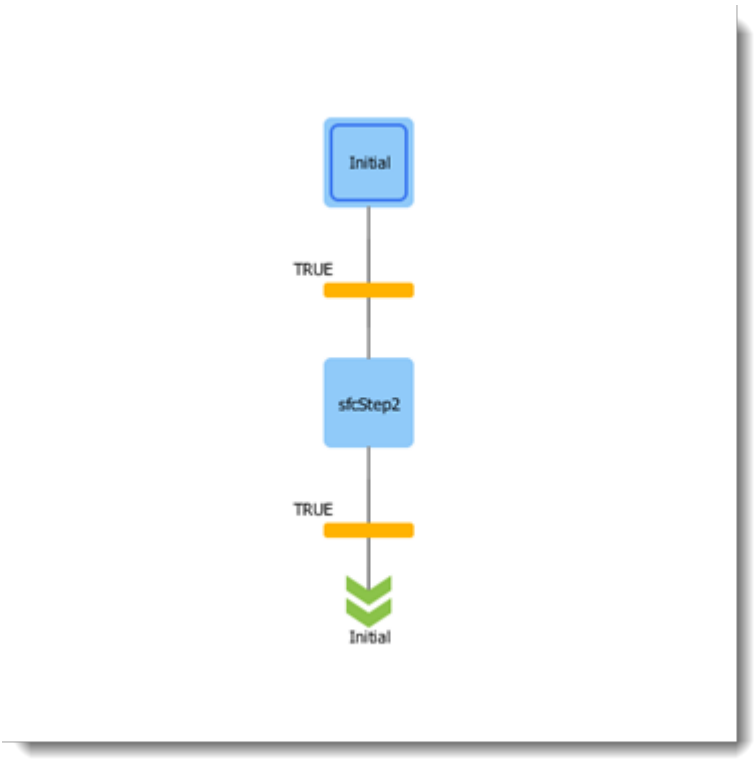
Из-за необходимости внутренней памяти только функциональные блоки и программы могут быть реализованы в SFC, функции такой возможности лишены.

Целью применения SFC является разделение задачи на простые этапы с формально определенной логикой работы системы. SFC дает возможность быстрого построения прототипа системы без программирования. Причем для отработки верхнего уровня не требуется детальное описание действий.

Применение SFC в объемных компонентах позволяет сократить время выполнения и, соответственно, время реакции системы. При помощи шагов монолитная программа разбивается на короткие фрагменты, выполняющиеся в разных рабочих циклах. В других языках реализация объемных и в то же время быстрых программ требует дополнительного кодирования механизма поэтапного выполнения. SFC стимулирует к равномерному распределению вычислительной мощности процессора практически без дополнительных усилий программиста. Реализация функциональных блоков и программ в SFC имеет существенную особенность. Отсутствуют первая и последняя инструкции. Оператор RETURN также не используется. Программа как бы не имеет конца. Каждый вызов SFC компонента равноценен выполнению одного цикла. Что конкретно будет выполнять компонент, зависит от его предыдущего состояния.

19.2.12.3.1 Шаги

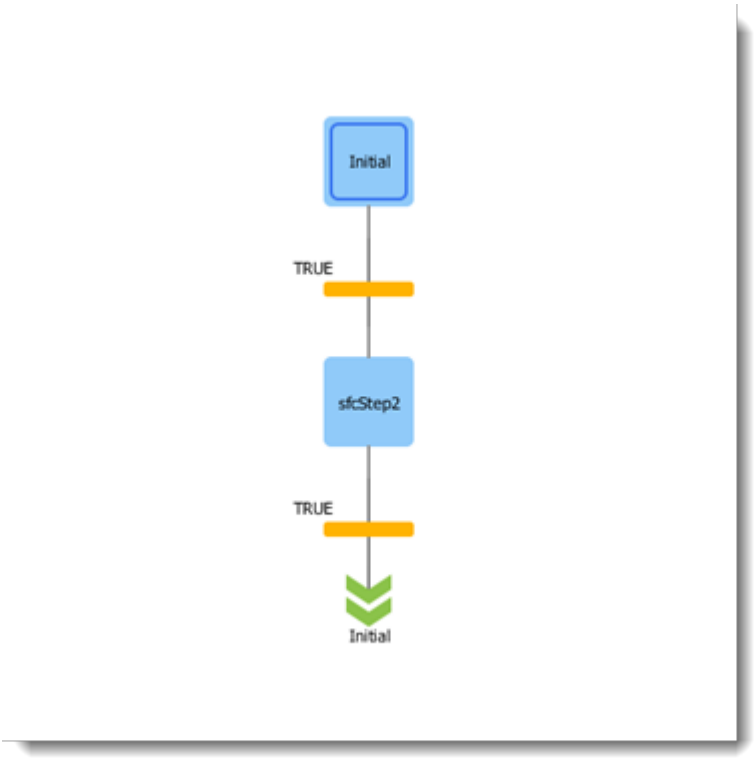
Любая SFC-схема составляется из элементов, представляющих шаги и условия переходов. Шаги показаны на схеме прямоугольниками:



Реальная работа шага (действия) описывается в отдельном окне системы программирования и не отражается на диаграмме. О назначении шага SFC говорит только его название. Шаги на схеме могут быть пустыми, что не вызывает ошибки при компиляции проекта. Пустые шаги являются нормой при применении программирования сверху вниз, характерного для SFC. Определить действия, соответствующие шагу, можно в любое время. Нет ничего удивительного, если пустые шаги останутся и в законченном проекте. Задачей пустого шага является ожидание перехода. Каждая SFC-схема начинается с шага, выделенного графически двойными вертикальными линиями по всему периметру. Это — начальный шаг. Начальный шаг присутствует обязательно и только в одном экземпляре, хотя и может быть пустым.

19.2.12.3.2 Переходы

Ниже шага на соединительной линии присутствует горизонтальная черта, обозначающая переход:



Условием перехода может служить логическое выражение. Переход выполняется при соблюдении двух условий:

1) переход разрешен (соответствующий ему шаг активен);

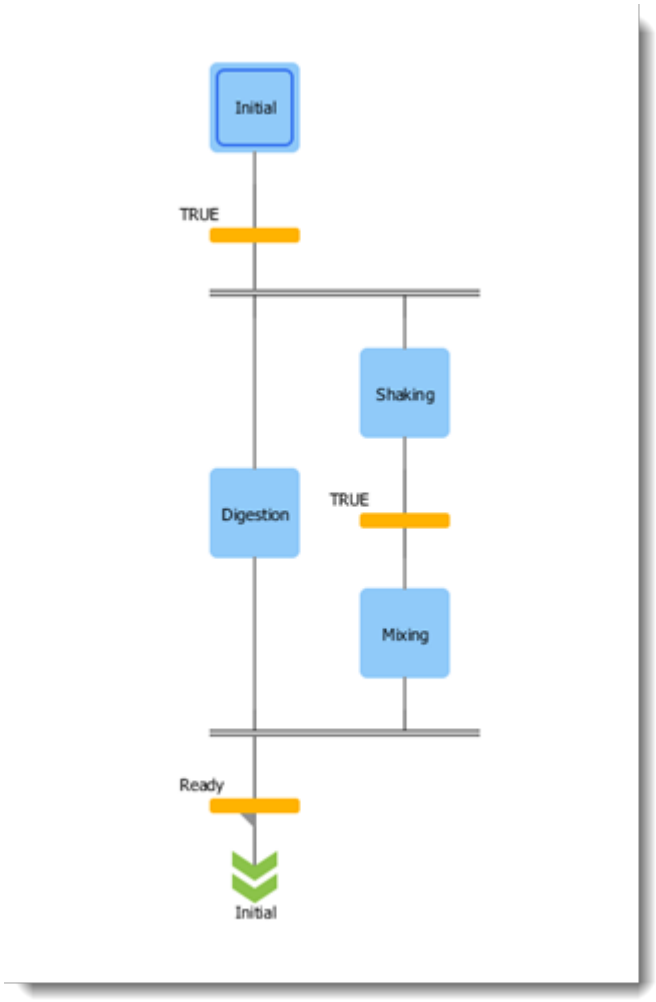
2) условие перехода имеет значение TRUE.

На диаграмме записывается только идентификатор перехода. Само же условие описывается в отдельном окне с применением языка ST. В условном выражении перехода нельзя использовать операцию присваивания. Признаком того, что идентификатор перехода на диаграмме является отдельно реализованным условием, а не простой логической переменной, служит закрашенный угол перехода.

В качестве условия перехода может быть задана логическая константа. Если задано TRUE, то шаг будет выполнен однократно, за один рабочий цикл, далее управление перейдет к следующему шагу. Если задано условие FALSE, то шаг будет выполняться бесконечно.

19.2.12.3.3 Параллельные ветви

Несколько ветвей SFC могут быть параллельными. Признаком параллельных ветвей на схеме является двойная горизонтальная линия. Каждая параллельная ветвь начинается и заканчивается шагом. То есть условие входа в параллельность всегда одно, условие выхода тоже одно на всех:

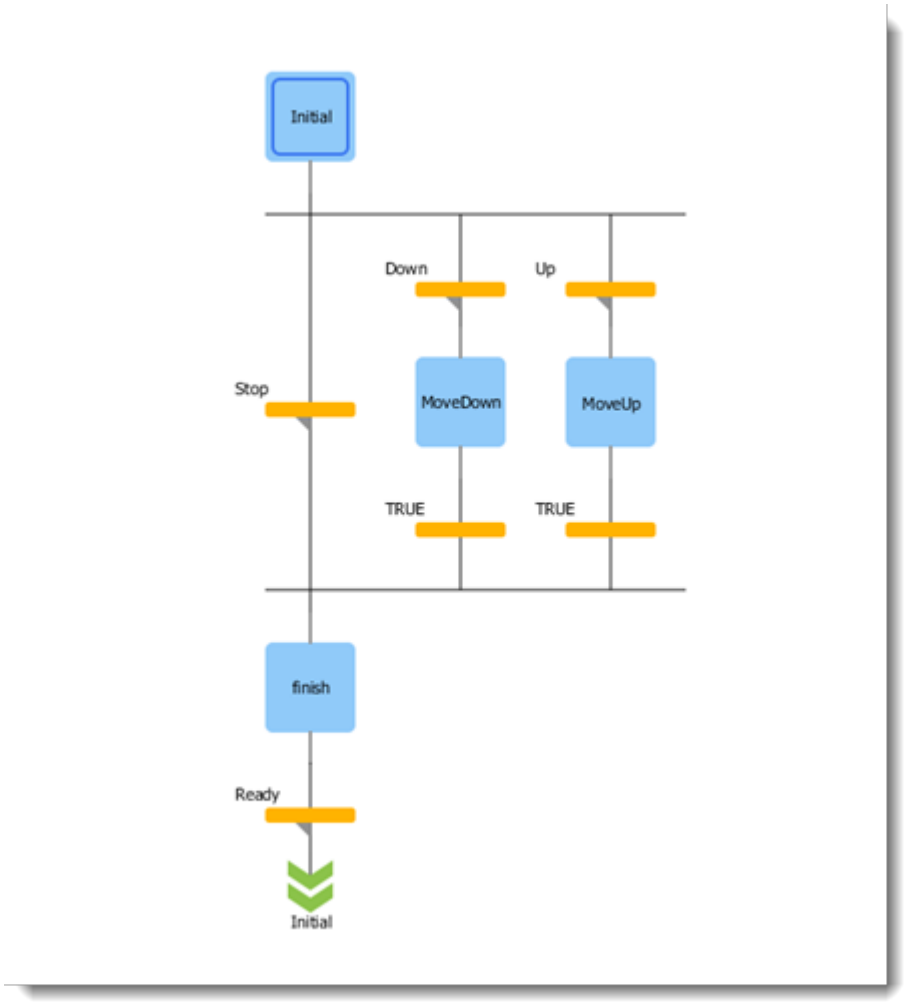


Теоретически, параллельные ветви выполняются одновременно. В жизни это означает в одном рабочем цикле, слева направо. Условие перехода, завершающее параллельность, проверяется только в случае, если в каждой параллельной ветви активны последние шаги. В данном примере Shaking будет выполнен однократно, далее Digestion и Mixing будут работать параллельно до выполнения условия Ready.

19.2.12.3.4 Альтернативные ветви

Несколько ветвей SFC могут быть альтернативными ветвями. Признаком альтернативных ветвей на схеме является одинарная горизонтальная линия. Каждая альтернативная ветвь начинается и заканчивается собственным условием перехода. Проверка альтернативных условий выполняется слева направо. Если верное условие найдено, то прочие альтернативы не рассматриваются. В альтернативных ветвях всегда работает только одна из ветвей, поэтому ее окончание и будет означать переход к следующему за альтернативной группой шагу.

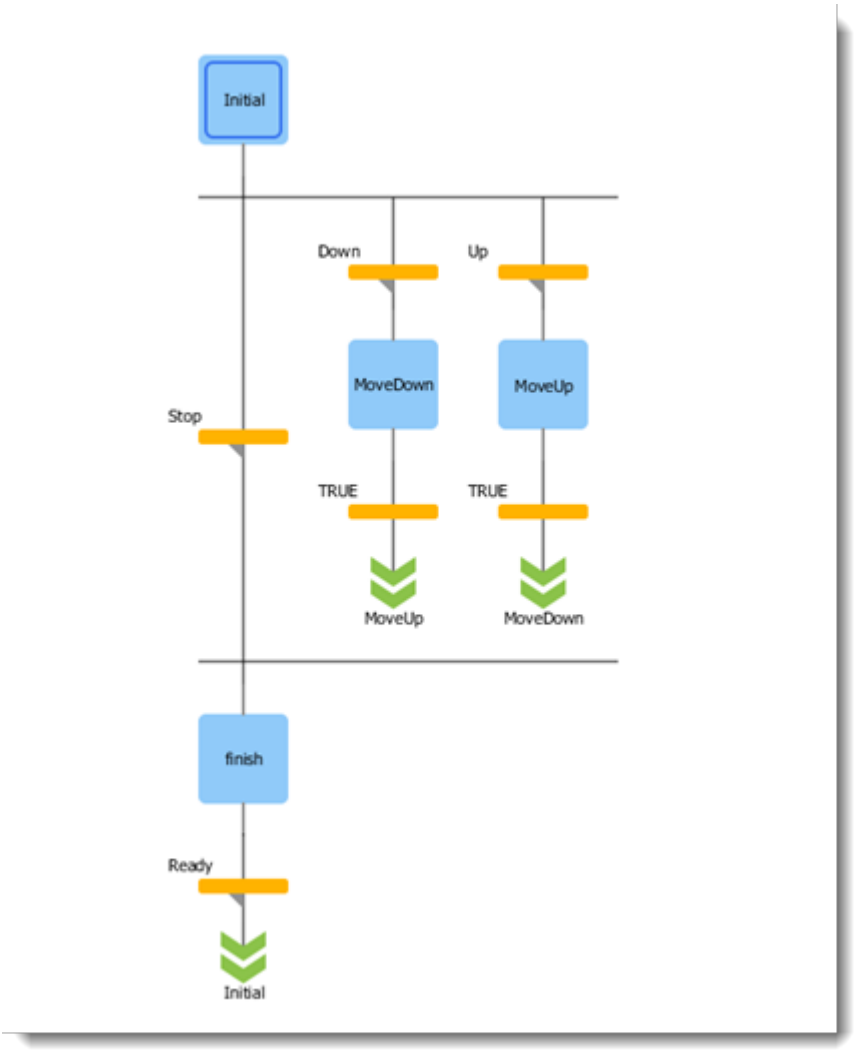
В данном примере альтернатива Stop оценивается первой. Шаги `MoveDown` и `MoveUp` имеют шанс стать активными, только если Stop равен FALSE.



При создании альтернативных ветвей желательно задавать взаимоисключающие условия. В этом случае вероятность допустить ошибку при анализе или в процессе доработки диаграммы значительно ниже.

19.2.12.3.5 Переход на произвольный шаг

В общем случае SFC-схема выполняется сверху вниз. Стандартом допускается создание переходов на произвольный шаг. Для этого применяются соединительные линии с промежуточными стрелками или поименованные переходы. То есть переход выполняется на шаг, имя которого указано под стрелкой. В примере, шаги Move_Dwn и Move_Up последовательно активируют друг друга:



Заметьте, что условие Stop при этом проверяться не будет, шаги MoveDown и MoveUp соединены в логическое кольцо, имеющее 2 варианта входа, но ни одной возможности выхода. Маркер активности будет перемещаться исключительно в этом кольце.

Прыжок из одной ветви параллельного блока наружу вызывает эффект размножения маркера. Прыжок внутрь параллельного блока нарушает параллельность ветвей. Подобных трюков необходимо избегать.

19.2.12.3.6 Действия

При применении действий подход несколько иной. Сначала определяются действия (виды работ), которые должна выполнять система, а затем уже составляется диаграмма, в которой определяется их порядок и взаимосвязь. Каждое действие сопоставляется одному или нескольким шагам. Причем вполне возможно, что некоторое действие должно запускаться в одном шаге и останавливаться в другом. Также возможно, что начатое действие должно закончить свою работу вообще независимо ни от каких шагов. Например, начав движение, кабина лифта должна как минимум доехать до ближайшего этажа и выпустить пассажиров, даже если дана команда на окончание работы. Действия показываются на SFC-диаграмме в виде прямоугольников, расположенных справа от шага и привязанных к нему графически. Существенно важным здесь является то, что одно и то же действие можно многократно использовать в разных шагах. Так, например шаги Cooling (охлаждение) и Drying (сушка) используют действие air-cooling (воздушный обдув). Действия не принадлежат конкретному шагу, а являются самостоятельными программными элементами SFC-компонента.

Идентификаторы действий должны быть уникальны в пределах компонента и не должны совпадать с идентификаторами шагов и переходов.

КЛАССИФИКАТОРЫ ДЕЙСТВИЙ

Тип	Описание
N - несохраняемое действие	Данное действие будет выполняться в каждом рабочем цикле, пока активен шаг.
P - импульс	Действие выполняется один раз при активации и второй раз после деактивации шага.

P0 - импульс (задний фронт)	Действие выполняется один раз при деактивации.
P1 - импульс (передний фронт)	Действие выполняется один раз при активации.
S - сохраняемое	Действие активируется и остается активным до сброса. Действие продолжит выполняться в каждом цикле даже тогда, когда шаг уже не активен.
R - сброс	Действие деактивируется.
L - ограниченное по времени	Действие активируется вместе с шагом и остается активным на заданное время, но не дольше, чем шаг.
SL - сохраняемое и ограниченное по времени	Действие активируется вместе с шагом и остается активным заданное время, вне зависимости от активности шага. Действие можно деактивировать досрочно из другого шага с классификатором R.
D - отложенное	Действие активируется через заданное время после активации шага и остается активным, пока активен шаг. Если шаг окажется активным меньше заданного времени, то действие не будет активировано.
DS - отложенное сохраняемое	Действие активируется через заданное время после активации шага и остается активным до сброса. Если шаг активен меньше заданного времени, то действие не будет активировано. При параллельном выполнении сброса в процессе отсчета времени (в другом шаге с классификатором R) действие не будет активироваться.
SD - сохраняемое отложенное	Действие активируется через заданное время после активации шага, даже если шаг уже не активен. Но если в процессе отсчета задержки активации выполнить сброс (в другом шаге с классификатором R), то активация не произойдет. Активированное действие остается активным до сброса.

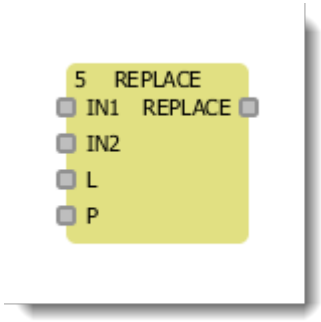
Классификаторы L, D, SD, DS и SL требуют указания константы времени в формате TIME. Например: T#10s.

19.2.12.3.7 Внутренние переменные шага

Для каждого шага неявно объявлена структура из двух доступных по чтению переменных. Первая переменная типа BOOL носит название `state` и является признаком активности шага. По смыслу она равноценна логической переменной шага `<StepName>` в упрощенной реализации SFC. Логическая единица является признаком активности шага. Вторая переменная типа TIME называется `currentTime` и указывает время активности шага. Доступ к переменным шага возможен через имя шага и точку, как к данным структуры или экземплярам функционального блока — `<StepName>.currentTime`. Переменные шага можно использовать в условиях переходов.

19.2.12.4 Функциональные блоковые диаграммы (FBD)

Диаграмма FBD строится из компонентов, отображаемых на схеме прямоугольниками. Входы POU изображаются слева от прямоугольника, выходы справа. Внутри прямоугольника указывается тип POU и наименования входов и выходов. Для экземпляра функционального блока его наименование указывается сверху, над прямоугольником. Выполнение программы на FBD происходит согласно порядку, указанному в верхнем правом углу блока. Размер прямоугольника зависит от числа входов и выходов и устанавливается графическим редактором автоматически:

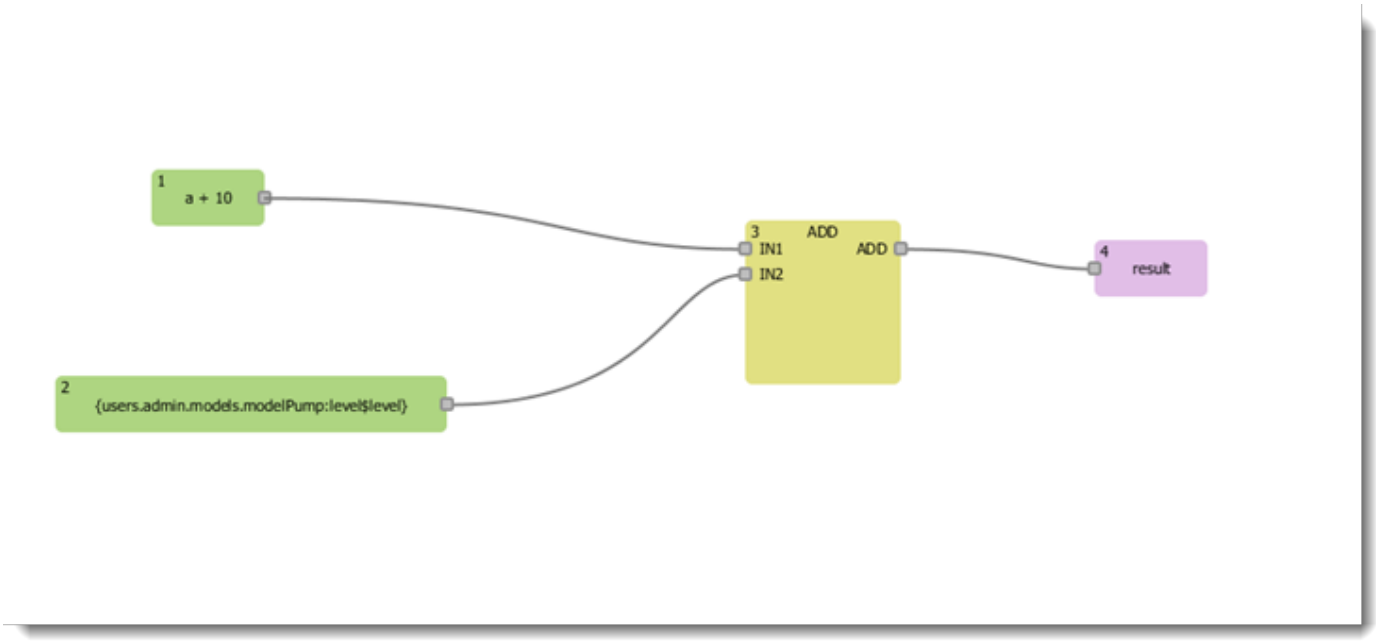


СОЕДИНИТЕЛЬНЫЕ ЛИНИИ

Прямоугольники POU в FBD соединены линиями связи. Соединения имеют направленность слева направо. Вход блока может быть соединен с выходом блока, расположенного слева от него. Помимо этого, вход может быть соединен с переменной или константой. Соединение должно связывать переменные или входы и выходы одного типа. Ширина соединительной линии в FBD роли не играет.

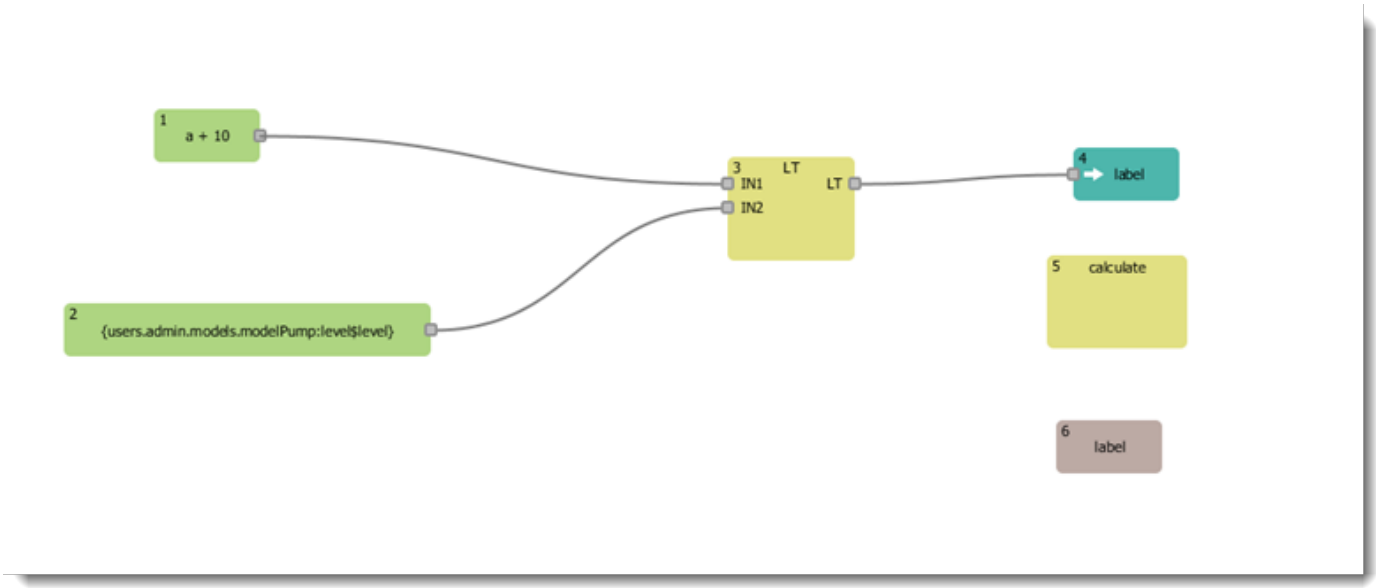
ВЫРАЖЕНИЯ ST В FBD

Также есть возможность записывать выражения ST на входе графических блоков. Такой прием расширяет стандартный FBD и часто оказывается достаточно удобным. Компактная форма представления выражений облегчает запись и чтение функциональных диаграмм.

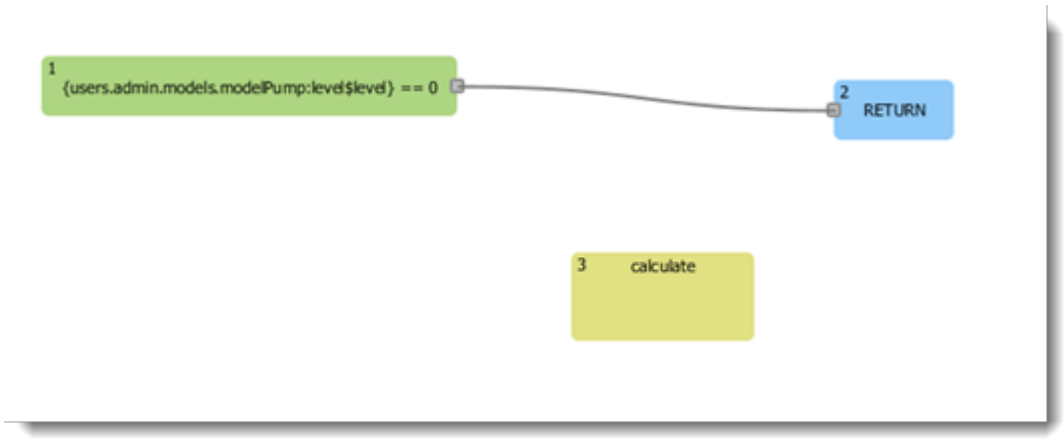


МЕТКИ, ПЕРЕХОДЫ И ВОЗРАТ

Порядок выполнения блоков FBD можно принудительно изменять, используя метки и переходы. Графический редактор автоматически нумерует блоки. Переход обязательно связан с логической переменной и выполняется, если выражение истинно. Метки и переходы представлены на рисунке:



Оператор возврата RETURN используется так же, как и переход на метку, т.е. в связке с логической переменной:



19.2.12.5 Релейные диаграммы (LD)

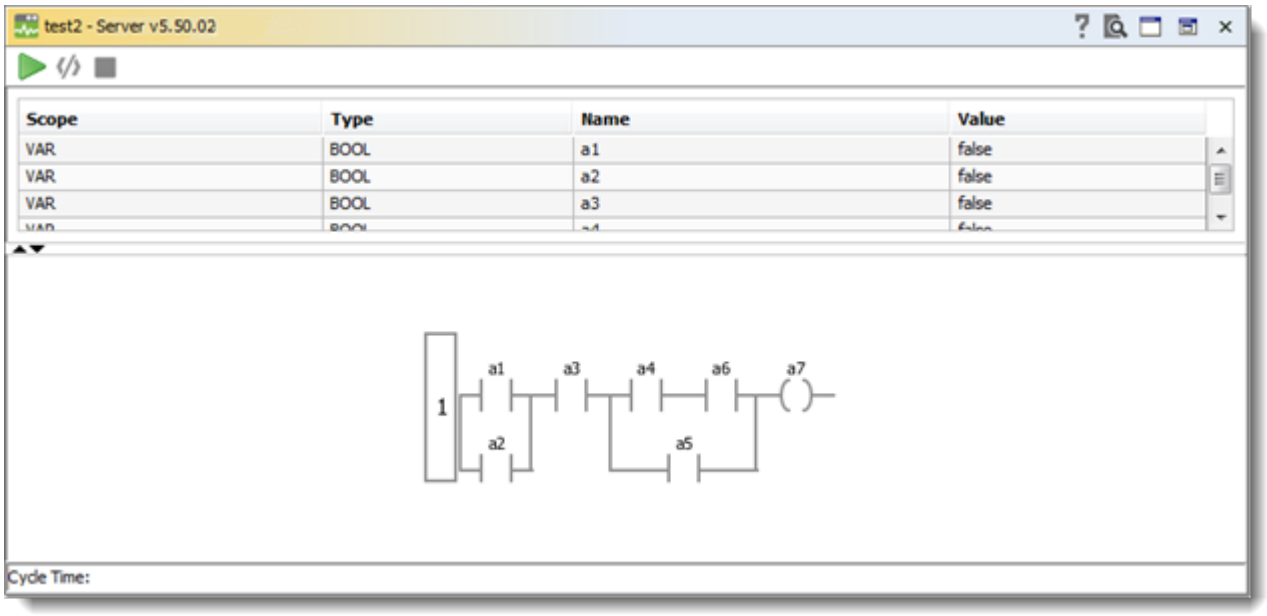
Язык релейных или релейно-контактных схем (Ladder Diagrams - LD) – графический язык, реализующий структуры электрических цепей.

Лучше всего LD подходит для построения логических переключателей, но достаточно легко можно создавать и сложные цепи, как в FBD. Кроме того, LD достаточно удобен для управления другими программными компонентами.

Релейная схема представляет собой вертикальную шину питания, которая всегда находится в состоянии "ON". Справа расположены горизонтальные цепи, образованные контактами и катушками. Количество контактов и катушек в цепи произвольно. Если последовательно соединенные контакты замкнуты, ток идет по цепи, и катушка включается ("ON"). При необходимости можно включить параллельно несколько катушек.

В LD каждому контакту ставится в соответствие переменная типа Boolean. Если переменная имеет значение TRUE, это значит, что условие передается с левого на правый полюс по соединительной линии. В противном случае, правый полюс получает значение OFF. Имя переменной или выражение пишется над контактом.

Последовательное соединение контактов или цепей равноценно логической операции И. Параллельное соединение образует ИЛИ. Цепь может быть либо замкнутой "ON", либо разомкнутой "OFF". Это как раз и отражается на катушке и, соответственно, на значении логической переменной катушки (TRUE/FALSE).



Приведенная выше схема эквивалентна выражению:

```
a7:= (a1 OR a2) AND a3 AND ((a4 AND a6) OR a5);
```

Контакты

Контакты обозначаются двумя параллельными линиями: | |, и могут иметь состояния "ON" или "OFF". Эти состояния соответствуют значениям TRUE или FALSE.

Каждому контакту соответствует выражение типа Boolean. Если переменная имеет значение TRUE, это значит, что состояние передается по соединению слева направо, в противному случае, правое соединение имеет значение "OFF".

Контакты могут быть соединены параллельно, тогда соединение передает состояние "ON", когда хотя бы одна из ветвей передает "ON". Если контакты соединены последовательно, то для того, чтобы соединение передало "ON", необходимо, чтобы оба контакта передавали "ON". Это соответствует электрической параллельной и последовательной схемам.

ТИПЫ КОНТАКТОВ

Тип	Символ	Описание
Нормально разомкнутый контакт		Состояние левого соединения копируется в правое соединение, если значение соответствующей логической переменной TRUE. В противном случае, состояние правого соединения OFF.
Нормально замкнутый контакт		Состояние левого соединения копируется в правое соединение, если значение соответствующей логической переменной FALSE. В противном случае, состояние правого соединения OFF.
Контакт для обнаружения заднего фронта (спад сигнала)		Правое соединение устанавливается в состояние ON, если переход связанного фактического параметра происходит из TRUE в FALSE, и состояние левого соединения далее остается FALSE. Иначе, состояние правого соединения OFF.
Контакт для обнаружения переднего фронта (нарастание сигнала)		Правое соединение устанавливается в состояние ON, если переход связанного фактического параметра происходит из FALSE в TRUE, и состояние левого соединения далее остается TRUE. Иначе, состояние правого соединения OFF.

Катушки

В правой части схемы может находиться любое количество катушек, которые обозначаются круглыми скобками (). Катушка передает значение соединения слева направо и копирует его в соответствующую логическую переменную. На линии входа может быть значение ON (соответствует значению логической переменной TRUE), или значение OFF (соответствует FALSE).

ТИПЫ КАТУШЕК

Тип	Символ	Описание
Катушка		Состояние левого соединения передается в соответствующую логическую переменную и правое соединение.
Инверсная катушка		Состояние левого соединения передается в правое соединение.

		Инверсное состояние левого соединения передается в соответствующую логическую переменную. Таким образом, если состояние левого соединения OFF, состояние соответствующей переменной - TRUE, и наоборот.
Катушка по спаду		В катушке обнаружения спада фронта состояние левого соединения передается в правое соединение. Связанная логическая переменная будет установлена в состояние TRUE для цикла программы, если произошел переход левого соединения из ON в OFF.
Катушка по фронту		В катушке обнаружения нарастания фронта состояние левого соединения передается в правое соединение. Связанная логическая переменная будет установлена в состояние TRUE для цикла программы, если произошел переход левого соединения из OFF в ON.
Катушка сброса (RESET катушка)		В катушке сброса состояние левого соединения передается в правое соединение. Связанный логический фактический параметр устанавливается в состояние FALSE, если левое соединение имеет состояние ON, иначе он не изменяется. Связанный логический фактический параметр может устанавливаться только катушкой установки.
Катушка установки (SET катушка)		В катушке установки состояние левого соединения передается в правое соединение. Связанный логический фактический параметр устанавливается в состояние TRUE, если левое соединение имеет состояние ON, иначе он не изменяется. Связанный логический фактический параметр может сбрасываться только катушкой сброса.

Функции и функциональные блоки

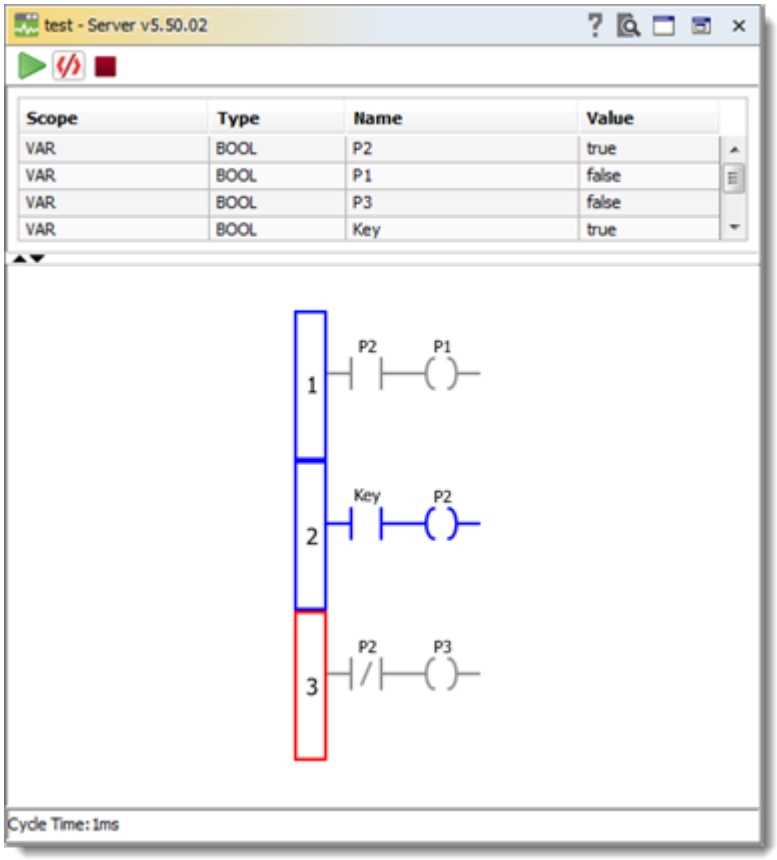
В LD-диаграмму можно вставить функции и функциональные блоки. Функциональные блоки должны иметь логические вход и выход и могут использоваться так же, как контакты.

Порядок исполнения диаграммы

Идеология релейных схем подразумевает параллельную работу всех цепей. Ток во все цепи подается одновременно. В LD решение диаграммы выполняется последовательно слева направо и сверху вниз. В каждом рабочем цикле однократно выполняются все цепи диаграммы, что и создает эффект параллельности работы цепей. Любая переменная в рамках одной цепи всегда имеет одно и то же значение. Если даже катушка в цепи изменит переменную, то новое значение поступит на контакты только в следующем цикле.

Цепи, расположенные ниже, получат новое значение переменной сразу. Цепи, расположенные выше, — только в следующем цикле. Строгий порядок выполнения схемы очень важен. Случайный или даже истинно параллельный порядок выполнения цепей может приводить к эффекту "гонок", встречающемуся в электронных схемах с триггерами. Благодаря жесткому порядку выполнения LD-диаграммы сохраняют устойчивость при наличии обратных связей.

В приведенной схеме включение Key вызовет мгновенное (в том же цикле) включение P2 и отключение P3. Реле P1 будет включено только в следующем цикле, причем даже если Key уже в обрыве (FALSE):



УПРАВЛЕНИЕ ПОРЯДКОМ ИСПОЛНЕНИЯ

Порядок выполнения цепей диаграммы можно принудительно изменять, используя переходы (jumps). Переход равнозначен выходной катушке и выполняется, если выходная переменная имеет значение TRUE. Переход осуществляется на начало цепи, номер которой указан в переходе. Используя переход, можно пропустить выполнение части диаграммы. Пропущенные цепи не сбрасываются, а именно не выполняются — замирают в том положении, в котором были ранее. Переход вверх допускается и позволяет создавать циклы. Проверка условий окончания цикла, естественно, лежит на совести программиста.

Специальный оператор RETURN прекращает выполнение LD диаграммы. Если RETURN встречается в основной программе, рабочий цикл прерывается. В функциях и функциональных блоках происходит возврат в место вызова. Иными словами, использование перехода RETURN аналогично по смыслу оператору RETURN в текстовых языках.

19.3 Контроль доступа

В этой главе говорится о **AtomMind Access Control**, многопользовательской системе контроля доступа на основе AtomMind.

Для продолжения перейдите к Обзору 2008.

19.3.1 Обзор

AtomMind Access Control это многопользовательская система контроля доступа сотрудников, основанная на ядре AtomMind. Она тесно взаимосвязана с **AtomMind Time and Attendance** и другими решениями AtomMind, разработанными для управления умным домом.

Основные возможности AtomMind Access Control

- Система контроля рабочего доступа на базе контроллера TIBBO Access Control.
- Улучшенный сервис оповещения, журналирования и обработки событий
- Поддержка крупных организаций со сложными многоступенчатыми иерархиями персонала
- Импорт данных о сотруднике/владельце смарт карты

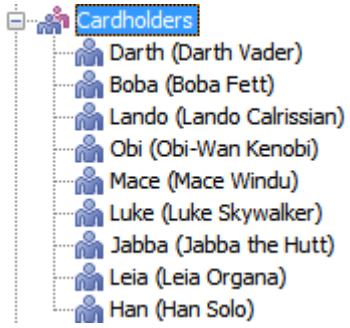
- Ролевая модель управления доступом и контроль масштабируемости предприятия
- Интегрированные GUI-интерфейс и Редактор Отчетов
- Интеграция со сторонними системами (дверные замки, турникеты, RFID-считыватели и т.д.) через открытые протоколы
- Лёгкое внедрение и администрирование

19.3.2 Настройка структуры организации

AtomMind - масштабируемое решение. Это означает, что AtomMind может использовать как компания, состоящая из десяти человек, так и крупное предприятие со штатом в 30,000 сотрудников. Таким образом, у нас есть два основных метода добавления штатных служащих, или, говоря на языке AtomMind *пользователей карт*:

Горизонтальная структура

Этот формат подходит для малых организаций. Если Вы собираетесь использовать учёт рабочего времени AtomMind в компании, состоящей из 30 человек, Вы, возможно, не собираетесь разделять её на отделы и подразделения. Вы, скорее всего, пожелаете внести своих сотрудников в список наиболее простым способом. Для этого предназначен узел **Пользователи карт**, расположенный в системном дереве. Горизонтальная структура выглядит следующим образом:



Иерархическая структура

Эта опция предназначена для более крупных организаций. Она позволяет разделить организацию на три иерархических уровня.

- **Организация ():** это самый верхний уровень. Учитывается вся корпорация целиком. Обратите внимание, что у Вас может быть несколько организаций, поэтому Вы могли бы использовать эту структуру для каждого отдельного физического места (например, для каждого из филиалов международной компании).
- **Подразделение ():** это средний уровень. Сюда может относится, например, R&D или Коммерческий отдел.
- **Отдел ():** Это нижний уровень, например, Отдел по работе с корпоративными клиентами - это отдел, входящий в подразделение "Коммерческий отдел".

Вам не нужно вдаваться в детали. Например, Вы могли бы хранить все информацию о Вашей организации на уровне подразделения. Кроме того, если Вы хотите использовать терминологию, в соответствии с Вашей текущей структурой организации, и разделяете её на "отделы" без "подразделений", Вы можете создать единую организацию, содержащую одно подразделение, и поместить все отделы в это подразделение. Основное требование заключается в том, что иерархия фиксирована - отдел можно создать *лишь* в рамках подразделения.

Добавление организаций, подразделений и отделов

Для того, чтобы добавить новую организацию, подразделение или отдел, щёлкните дважды мышкой по соответствующему узлу. Например, двойной щелчок мышью по узлу ***Организации*** добавил бы новую организацию (Вам необходимо лишь выбрать для неё **имя** и **описать** её). После того, как Вы создали организацию, Вам следует кликнуть дважды по ней, чтобы добавить подразделение. Аналогичным образом создаются отделы.

ГРУППЫ

Вы можете создать группы подразделений, отделов и владельцев карт для использования в отчётах и выполнения различных групповых операций. Группы 751 описаны в соответствующей теме.

ТИПОВЫЕ ДЕЙСТВИЯ

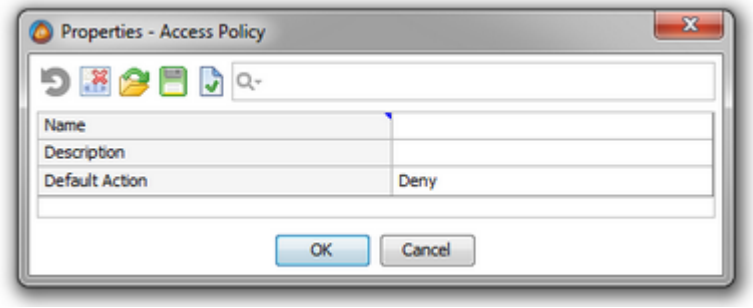
Создание, настройка и удаление контекстов (контекст 41 - это типичное название для объектов AtomMind, таких как организации, подразделения или отделы) называются *типовыми действиями.* Существует ряд других действий, которые Вы можете выполнить практически с любым контекстом в AtomMind. К таким действиям относится копирование контекста, редактирование разрешений к его доступу, мониторинг относящихся к нему событий и пр., эти действия включены в типовые действия 103. Обычно они не используются при типовом использовании системы.

19.3.3 Управление правами доступа

Для создания новой временной зоны вам нужно:

- Дважды щёлкнуть по узлу корня **Права доступа**

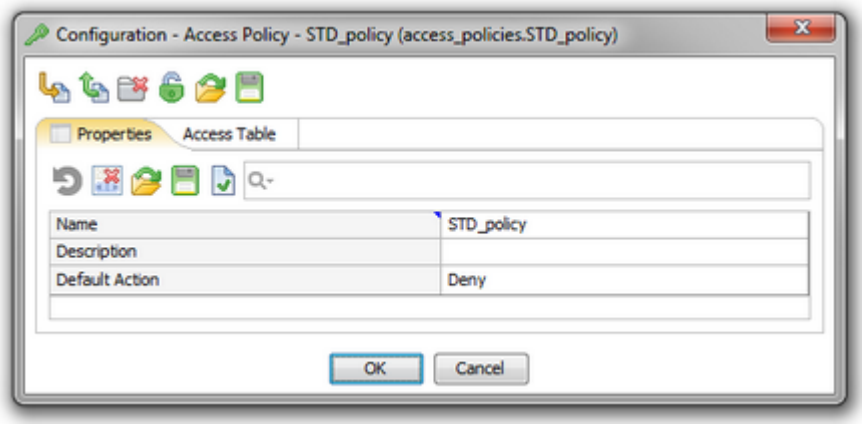
Диалог для добавления нового права доступа выглядит следующим образом:



- **Имя** – это *имя контекста* права доступа в системе. Поэтому, требуется подтверждение для названий контекстов 42 AtomMind. Звучит, конечно, немного сложно, но обычно это означает, что Вам не стоит использовать пробелы в этом поле, и его значение должно быть уникальным для данного уровня иерархии.
- **Описание** – это ваш комментарий о текущем праве доступа.
- **Действие по умолчанию** описывает поведение по умолчанию.

Редактирование прав доступа

При двойном щелчке по праву доступа (или кликайте правую кнопку мыши и выбирайте **Редактировать право доступа**) вы увидите следующее диалоговое окно:



Заметьте, что здесь появляется новая вкладка. При добавлении права доступа, вам доступна лишь первая вкладка (свойства права доступа). Теперь у вас есть дополнительная таблица – **Таблица доступа**.

Таблица доступа: Именно сюда вы вводите информацию о праве доступа. Больше информации в Управлении элементами доступа 201.

Выбор многопользовательского доступа

Вы можете выбрать многопользовательский доступ (используя Shift+клик для сплошного выбора и Ctrl+click для выбора многопользовательского доступа в различных местах системного дерева). Если вы затем сделаете щелчок правой кнопкой мыши по выбранным элементам, вы сможете редактировать их все (или удалить их и т.д.).

19.3.4 Управление элементами доступа

В AtomMind каждая политика доступа может иметь более одного элемента доступа.

Создание элемента доступа

- Нажмите правой кнопкой мыши по праву доступа, для которого вы хотите добавить элемент доступа и выберите **Конфигурировать право доступа**. Вы также можете сделать двойной щелчок по праву доступа.
- Кликните по вкладке **Таблица доступа**.
- Кликните по кнопке **Добавить ряд (** **)** и заполните следующие поля:

Точка доступа: Выберите одно устройство Контроля Доступа для использования в данном праве.

Считыватели: Выберите считыватели из текущего устройства Контроля доступа для использования в этом элементе доступа.

Временная зона: Выберите одну временную зону для этого элемента доступа.

Действие: Действие Разрешить или Отклонить для этого элемента доступа.

Удаление элемента доступа

Вы можете удалить ряд, содержащий информацию об элементе доступа, щёлкнув по кнопке **Удалить выбранный ряд (** **)**.

19.3.5 Управление электронными картами доступа

В AtomMind у каждого владельца электронного пропуска может быть больше одной карты или бейджа. Бейдж может быть как активным, так и неактивным. Если сотрудник теряет свой бейдж (или Вы переходите на другую карточную систему), можно просто добавить новый бейдж для этого человека, а прежний сделать неактивным.

У одного человека может быть несколько активных бейджей. Это означает, что есть множество способов отметить приход и уход с работы и это правильно отобразится в системе.

Создание карт

- Щёлкните правой кнопкой мыши по владельцу карты, для которого Вы хотите добавить карту, и выберите **Настроить пользователя карты**. Можно также дважды щёлкнуть мышью по элементу Пользователь Карты.
- Щёлкните мышью по вкладке **Карты/Бейджи**.
- Щёлкните мышью по кнопке **Добавить ряд (** **)** и заполните следующие поля:

ID карты, бейджа: должны соответствовать тому, что Ваше устройство передает AtomMind Server. Значение может быть от 1 до 100 знаков длиной и не может содержать пробелы.

Статус*: Активный, Приостановлен, Утрачен, Украден или Повреждён*. Любой статус, отличный от *Активного* деактивирует карту.

Дата активации: это та дата, с которой карта считается активной. Для карты, которая считается активной, дата активации должна относится к прошедшему периоду.

Дата деактивации: это та дата, с момента которой карта перестает быть активной. Для карты, которая считается активной, дата деактивации относится в будущему периоду (или же "не установлено").

Удаление карт

Технически Вы можете удалить ряд, содержащий информацию о карте, щёлкнув мышью по кнопке **Удалить выбранный ряд** ( ). Однако это не лучшее решение, поскольку это привело бы к тому, что все записи для этой карты исчезнут из последующих отчетов об учёте времени. Поэтому будет лучше деактивировать карту (отменить флажок **Активна**).

Когда Вы просто деактивируете карту (а не удаляете её), система будет знать, что ранее она принадлежала определённому владельцу карты и принимает это во внимание при последующих, связанных с ним отчётах.

ПОДРОБНЕЕ О ЛОГИКЕ АКТИВАЦИИ/ДЕАКТИВАЦИИ

Это может представлять некоторую сложность:

Если карта отмечена как **Активная**, система проверяет, соответствует ли её дата активации заявленной. Если её **дата Активации** соответствует заявленной, система посчитает карту активной.

Если карта не отмечена как **Активная**, система не посчитает её активной и даже не будет проверять её дату активации.

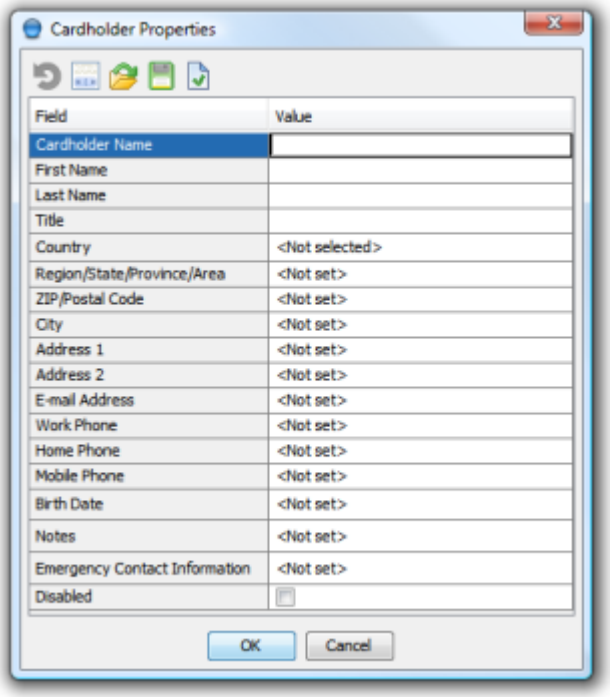
Если карта **Активная**, но её **дата Активации** относится к будущему периоду, или же **дата Деактивации** к прошлому, система посчитает карту неактивной.

19.3.6 Управление сотрудниками

Чтобы создать нового сотрудника, сделайте одно из следующих действий:

- Щёлкните дважды по корню узла **Сотрудники** или
- Щёлкните дважды по узлу **Сотрудники** интересующего Вас отдела, подразделения или организации чтобы внести информацию о новом сотруднике в этой административной единице. У Вас также есть возможность перетащить сотрудника в другое подразделение позже..

Диалог создания нового сотрудника (или редактирования уже существующего) выглядит следующим образом:

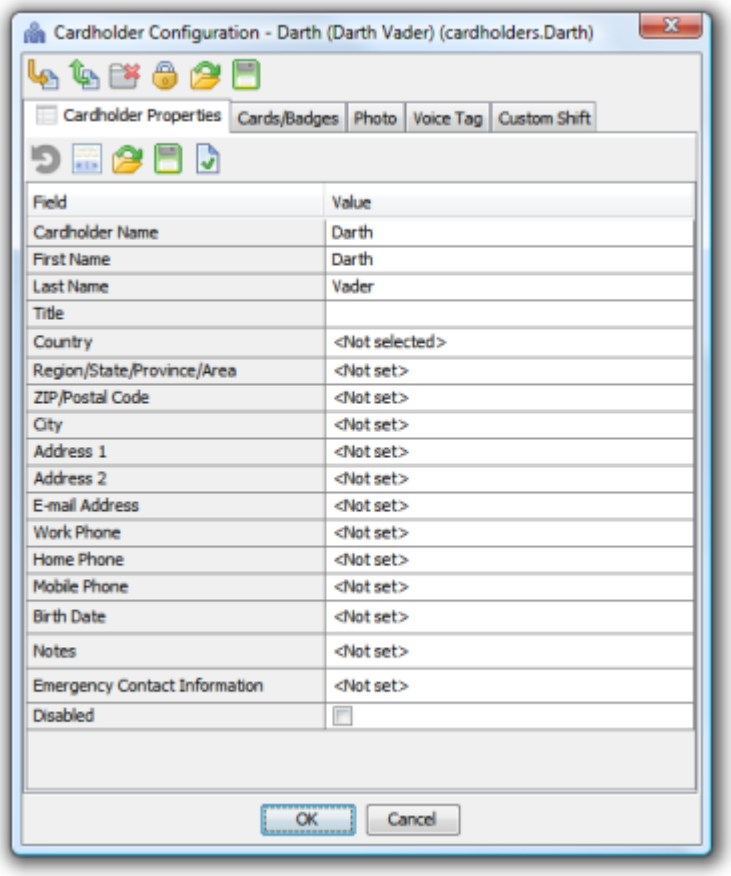


Свойства имеют привычные названия (фамилия, имя, контактный телефон и пр.). Есть только пара замечаний, на которые необходимо обратить внимание:

- **Имя** - имя *контекста* владельца в системе. Нет необходимости указывать реальное имя сотрудника (для этого существуют поля **Фамилия** и **Имя**). Это поле должно соответствовать правилу именования контекстов 42 AtomMind. Звучит сложно, но обычно это означает, что Вам не стоит использовать пробелы в этом поле и его значение должно быть уникальным для данного уровня иерархии (Вы не можете иметь два контекста *Joe* внутри одного и того же отдела. Одному из них следует дать имя *Joe1*).
- Одна ключевая деталь, которую нельзя указать в этом диалоге - действительный *номер карты,* с сотрудник регистрируется в системе! После добавления каждого сотрудника Вам необходимо отредактировать его свойства и указать номер его карты.
- Вы также можете импортировать 2045 список держателей карт из любой другой программной среды, поддерживающей экспорт в CSV.

Редактирование сотрудников

Когда вы дважды щёлкните по имени сотрудника (или щёлкните правой кнопкой мыши и выберите **Настроить**), Вы увидите следующий диалог:



Обратите внимание на дополнительные вкладки в этом диалоге. При добавлении сотрудника Вам был доступна только первая вкладка (свойства). Теперь появилось ещё четыре: **Идентификационные карточки, Фотография, Образец голоса** и **Особая рабочая смена**.

Идентификационные карточки: В этой вкладке вводят информацию о карте. Как следует из названия, каждый сотрудник может держать по нескольку карт. Подробнее об этом в Управление картами/беджиками 2048.

Фотография: Изображение сотрудника.

Образец голоса: Здесь указывается файл с записью образца голоса сотрудника с целью проверок.

Особая рабочая смена: Позволяет Вам указать индивидуальное расписание сотрудника, если оно отличается от общепринятого в его отделе. Подробнее об этом в Управление Расписанием 2017.

Перенос сотрудника или создание по образцу уже существующего

При перетаскивании узла сотрудника в другое место дерева организации создаётся копия перетаскиваемого узла. После этого Вы можете отредактировать только часть информации (имя, беджики и пр.), создав таким образом нового сотрудника по образцу уже существующего. После перетаскивания/копирования Вы можете удалить оригинальный узел, что будет эквивалентно переносу в новое местоположение.

Вы также можете реплицировать 1101 сотрудника (или любой другой контекст) или выполнить любые Общие действия 103 на нём.

Выбор множества сотрудников

Вы можете выбрать множество сотрудников (через Shift+клик для выделения соседей и Ctrl+клик для индивидуального выбора сотрудников в различных местах поддерева). Если следом за этим Вы кликните правой кнопкой мыши по выделенным записям, Вы сможете отредактировать их все (или удалить их все или пр.).

Пользовательские свойства сотрудников

Возможно, Ваша организация требует учёта некоторых свойств сотрудников, которые по умолчанию не представлены в AtomMind. Например, номер водительской лицензии или номер парковочного места или таблица данных об имуществе, за которое отвечает сотрудник (если сотрудник работает вне офиса и подключается к системе удалённо).

К счастью, Вы можете добавлять пользовательские свойства каждому сотруднику. Почитайте об этом в обучающем руководстве Добавление пользовательских свойств 1686.

19.3.6.1 Импорт сотрудников

В случае, если Вы производите миграцию с Вашей старой системы учета рабочего времени на AtomMind Time and Attendance и Ваша старая система способна выполнять экспорт списка сотрудников/владельцев карт в формате CSV, AtomMind может помочь Вам сохранить Ваше время с помощью функции импорта этих данных вместо выполнения длительного ввода данных вручную.

Покажем это на практическом примере. Ниже представлен CSV-файл, экспортированный из реально используемой программы учёта рабочего времени. Вы можете скопировать содержимое в файл `employees.csv` и проделать все самостоятельно.

```
EmployeeID,LastName,FirstName,Title,TitleOfCourtesy,BirthDate,HireDate,Address,City,Re
gion,PostalCode,Country,HomePhone,Extension,Photo,Notes,ReportsTo

1,Davolio,Nancy,Sales Representative,Ms.,12/08/48,05/01/92,507 - 20th Ave. E.\r\nApt.
2A,Seattle,WA,98122,USA,(206) 555-9857,5467,Object Type17,Education includes a BA in
psychology from Colorado State University in 1970.  She also completed "The Art of the
Cold Call."  Nancy is a member of Toastmasters International.,2

2,Fuller,Andrew,"Vice President, Sales",Dr.,02/19/52,08/14/92,908 W. Capital
Way,Tacoma,WA,98401,USA,(206) 555-9482,3457,Object Type17,"Andrew received his BTS
commercial in 1974 and a Ph.D. in international marketing from the University of
Dallas in 1981.  He is fluent in French and Italian and reads German.  He joined the
company as a sales representative, was promoted to sales manager in January 1992 and
to vice president of sales in March 1993.  Andrew is a member of the Sales Management
Roundtable, the Seattle Chamber of Commerce, and the Pacific Rim Importers
Association.",

3,Leverling,Janet,Sales Representative,Ms.,08/30/63,04/01/92,722 Moss Bay
Blvd.,Kirkland,WA,98033,USA,(206) 555-3412,3355,Object Type17,Janet has a BS degree in
chemistry from Boston College (1984).  She has also completed a certificate program in
food retailing management.  Janet was hired as a sales associate in 1991 and promoted
to sales representative in February 1992.,2

4,Peacock,Margaret,Sales Representative,Mrs.,09/19/37,05/03/93,4110 Old Redmond
Rd.,Redmond,WA,98052,USA,(206) 555-8122,5176,Object Type17,Margaret holds a BA in
English literature from Concordia College (1958) and an MA from the American Institute
of Culinary Arts (1966).  She was assigned to the London office temporarily from July
through November 1992.,2

5,Buchanan,Steven,Sales Manager,Mr.,03/04/55,10/17/93,14 Garrett Hill,London,,SW1
8JR,UK,(71) 555-4848,3453,Object Type17,"Steven Buchanan graduated from St. Andrews
University, Scotland, with a BSC degree in 1976.  Upon joining the company as a sales
representative in 1992, he spent 6 months in an orientation program at the Seattle
office and then returned to his permanent post in London.  He was promoted to sales
manager in March 1993.  Mr. Buchanan has completed the courses \"Successful
Telemarketing\" and \"International Sales Management.\"  He is fluent in French.",2

6,Suyama,Michael,Sales Representative,Mr.,07/02/63,10/17/93,Coventry House\r\nMiner
Rd.,London,,EC2 7JR,UK,(71) 555-7773,428,Object Type17,"Michael is a graduate of
Sussex University (MA, economics, 1983) and the University of California at Los
Angeles (MBA, marketing, 1986).  He has also taken the courses \"Multi-Cultural
Selling\" and \"Time Management for the Sales Professional.\"  He is fluent in
Japanese and can read and write French, Portuguese, and Spanish.",5

7,King,Robert,Sales Representative,Mr.,05/29/60,01/02/94,Edgeham Hollow\r\nWinchester
Way,London,,RG1 9SP,UK,(71) 555-5598,465,Object Type17,"Robert King served in the
Peace Corps and traveled extensively before completing his degree in English at the
University of Michigan in 1992, the year he joined the company.  After completing a
course entitled \"Selling in Europe,\" he was transferred to the London office in
March 1993.",5

8,Callahan,Laura,Inside Sales Coordinator,Ms.,01/09/58,03/05/94,4726 - 11th Ave.
N.E.,Seattle,WA,98105,USA,(206) 555-1189,2344,Object Type17,Laura received a BA in
psychology from the University of Washington.  She has also completed a course in
business French.  She reads and writes French.,2

9,Dodsworth,Anne,Sales Representative,Ms.,01/27/66,11/15/94,7 Houndstooth
Rd.,London,,WG2 7LT,UK,(71) 555-4444,452,Object Type17,Anne has a BA degree in English
from St. Lawrence College.  She is fluent in French and German.,5
```

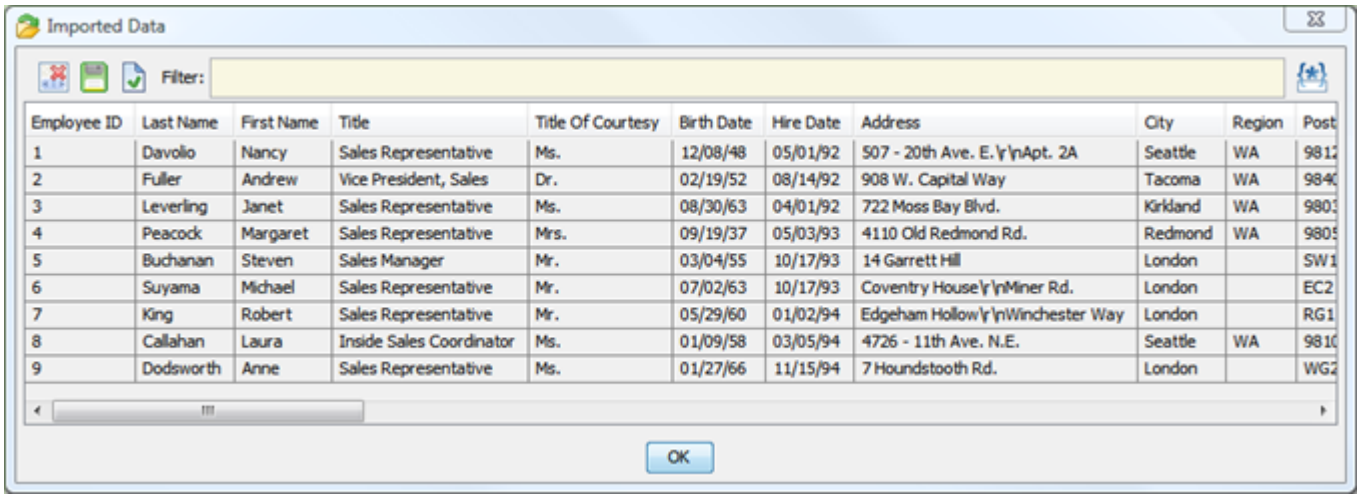
- Кликните правой кнопкой мыши по корню узла **Владельцы карт** (или любому узлу **Владельцы** карт в Вашей иерархии, если у вас есть подразделения, отделы и пр.).

- Щёлкните **Импорт**.
- В диалоге **Выбрать файл для импорта**, щёлкните **Выбрать** и используя навигатор по файловой системе укажите файл. В нашем случае это `employees.csv`.
- Щёлкните **OK**.
- Вы увидите диалог **Опции для импорта CSV**. Этот диалог имеет несколько опций:

1. **Разделитель полей**: Это - символ-разделитель для полей. По умолчанию это точка с запятой (;), но в нашем случае это запятая (,). Так что если Вы следуете нашим инструкциям, то измените его.
2. **Использовать разделитель текста**: Разделитель текста используется в начале и конце длинных строк, например "like this". Обычно это символ двойных кавычек. Эта опция отключена по умолчанию, но нам она нужна. Включите её.
3. **Разделитель текста**: Это непосредственно символ-разделитель. Можно оставить тот, что указан по умолчанию. В нашем файле используются как раз двойные кавычки.
4. **Символ комментария**: Для комментирования внутри файла. Текст, который AtomMind будет игнорировать при импорте. В нашем файле нет комментариев, но эту опцию можно оставить как есть.
5. **Режим экранирования**: Что, если бы Вам захотелось вставить в текст символ, который используется в качестве разделителя (например, двойные кавычки)? Есть два традиционных подхода к обработке этой ситуации. Некоторые программы используют обратную косую черту перед таким символом, при этом текст выглядит так: \". Другие программы просто повторяют двойные кавычки дважды вот так: "". Эта опция позволяет Вам выбрать один из этих двух способов экранирования. Для нашего примера нам необходимо изменить значение этой опции. Вы берите **Использовать обратную косую черту**.
6. **Заголовок**: Иногда первая запись в файле используется для именования полей или их описания. Иногда это просто отдельная запись и иногда она даже содержит мусор, который лучше не принимать во внимание. Эта настройка позволяет Вам выбрать один из четырёх режимов. Для нашего примера выберите **Описание полей**.

Смотрите приложение Опции импорта/экспорта CSV 2161.

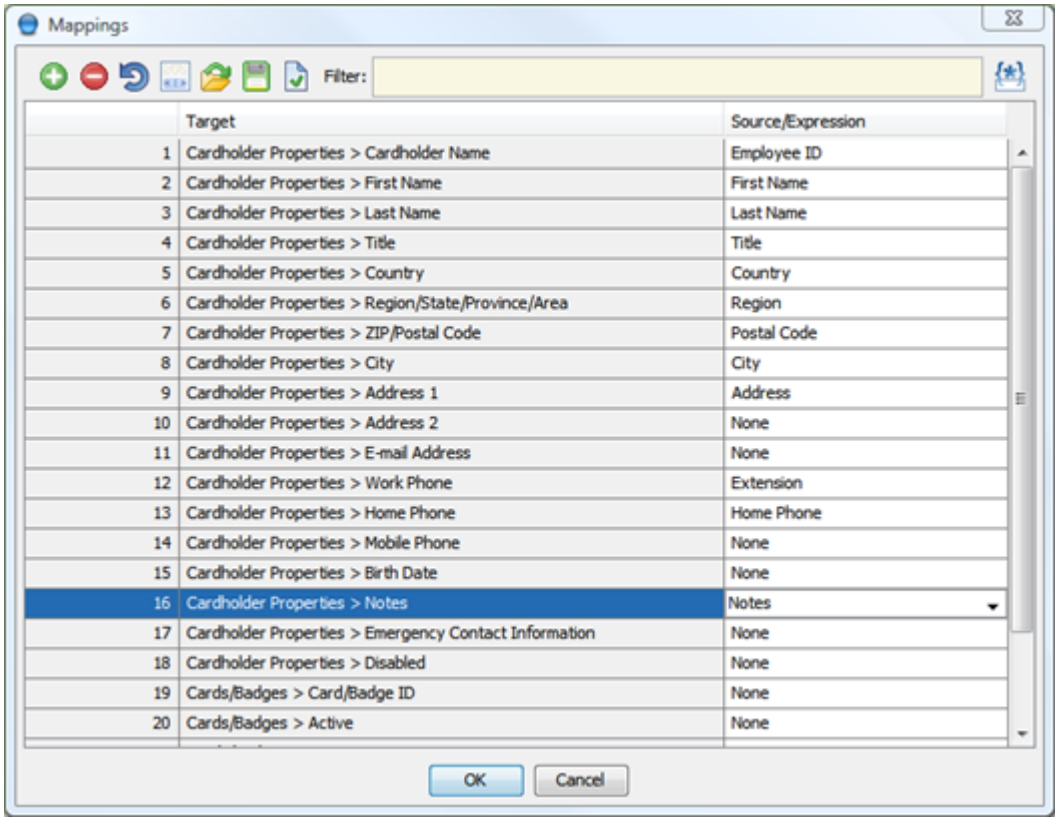
- Щёлкните **OK.**
- Если Вы всё правильно сделали, Вы увидите следующий диалог:



Employee ID	Last Name	First Name	Title	Title Of Courtesy	Birth Date	Hire Date	Address	City	Region	Post
1	Davolio	Nancy	Sales Representative	Ms.	12/08/48	05/01/92	507 - 20th Ave. E.\r\nApt. 2A	Seattle	WA	9812
2	Fuller	Andrew	Vice President, Sales	Dr.	02/19/52	08/14/92	908 W. Capital Way	Tacoma	WA	9840
3	Leverling	Janet	Sales Representative	Ms.	08/30/63	04/01/92	722 Moss Bay Blvd.	Kirkland	WA	9803
4	Peacock	Margaret	Sales Representative	Mrs.	09/19/37	05/03/93	4110 Old Redmond Rd.	Redmond	WA	9805
5	Buchanan	Steven	Sales Manager	Mr.	03/04/55	10/17/93	14 Garrett Hill	London		SW1
6	Suyama	Michael	Sales Representative	Mr.	07/02/63	10/17/93	Coventry House\r\nMiner Rd.	London		EC2
7	King	Robert	Sales Representative	Mr.	05/29/60	01/02/94	Edgeham Hollow\r\nWinchester Way	London		RG1
8	Callahan	Laura	Inside Sales Coordinator	Ms.	01/09/58	03/05/94	4726 - 11th Ave. N.E.	Seattle	WA	9810
9	Dodsworth	Anne	Sales Representative	Ms.	01/27/66	11/15/94	7 Houndstooth Rd.	London		WG2

Обратите внимание, насколько хорошо он разбил поля. Он также указывает имена полей в заголовках ячеек.

- Нашим следующим шагом будет поставить соответствия этим полям имена полей AtomMind. Щёлкните **OK**.



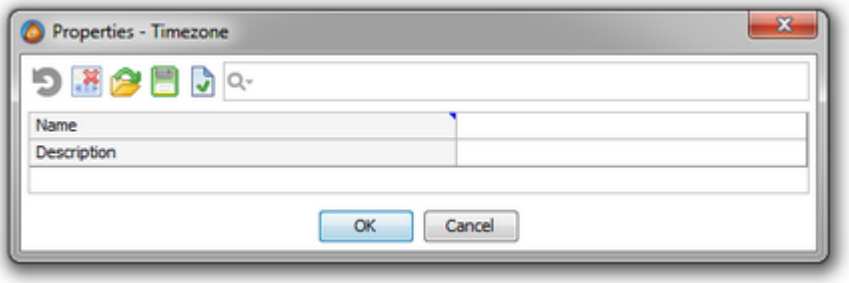
- Диалог **расставления соответствий** показывает каждое поле в записи о Владельце Карты в AtomMind. Раскройте выпадающее меню Источник/Выражение сразу за каждым из полей и выберите имя поля из Вашего CSV. Например, **Имя** так и останется **Именем** (Ваш CSV содержит определение этого поля), в то время как **Имя владельца карты** должно соответствовать **Идентификатору работника** (Employee ID). Если Вы не можете найти соответствующее поле, оставьте его как **None**.
- Выставьте соответствия, Которые Вас устроят и нажмите **OK.** AtomMind создаст новый контекст для каждого из владельцев карт. Импорт закончен.

19.3.7 Управление временными зонами

Для создания новой временной зоны вам понадобится:

- дважды щёлкнуть по узлу корня **Временные зоны**

Диалоговое окно для добавления новой временной зоны выглядит так:

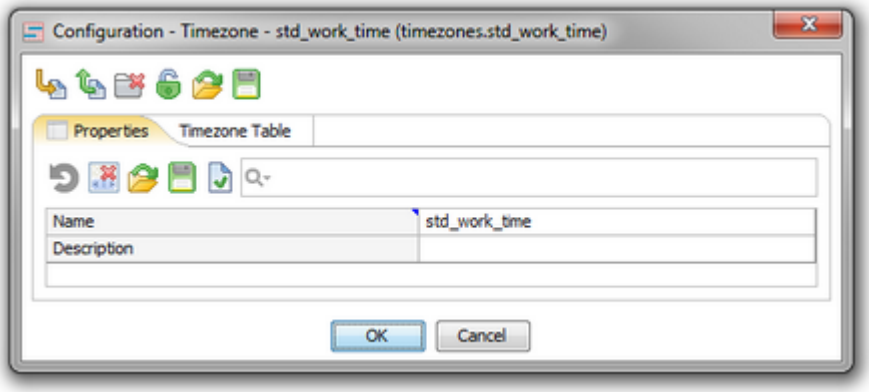


Все детали не требуют объяснений. Здесь нужно сделать всего несколько замечаний:

- **Имя –** это *имя контекста* временной зоны в системе. Поэтому, требуется подтверждение для названия контекстов 42 AtomMind. Звучит, конечно, немного сложно, но обычно это означает, что Вам не стоит использовать пробелы в этом поле, и его значение должно быть уникальным.

Редактирование временных зон

Когда вы делаете двойной щелчок по временной зоне (или щёлкните правой кнопкой мыши и выберите **Редактировать временную зону**), вы видите следующее диалоговое окно:



Заметьте, что здесь появляется новая вкладка. При добавлении временной зоны, вам доступна только первая вкладка (свойства временной зоны). Теперь у вас есть дополнительная вкладка – **Таблица временной зоны**.

Таблица временной зоны: Здесь вы вводите информацию об элементе временной зоны. Больше информации в Управление элементами временной зоны 2017.

Выбор нескольких временных зон

Вы можете выбрать несколько временных зон (используя Shift+клик для сплошной выборки и Ctrl+клик для выбора нескольких временных зон в разных местах системного дерева). Если вы затем кликнете правой кнопкой мыши по выбранным элементам, вы сможете редактировать их все (или удалить их и т.д.).

19.3.8 Управление элементами временной зоны

В AtomMind каждая временная зона может иметь более одного элемента.

Создание элемента временной зоны

- Кликните правой кнопкой мыши по временной зоне, для которой вы хотите добавить элемент и выберите **Конфигурировать временную зону**. Можете просто сделать двойной щелчок по временной зоне.
- Кликните по вкладке **Таблица временной зоны**.
- Нажмите кнопку **Добавить ряд (⊕)** и заполните данные поля:

Начиная со времени: Это время, начиная с которого временная зона должна быть активной.

До времени: Это время, до которого временная зона активна.

Начиная с года: Это год, начиная с которого временная зона должна быть активной.

До года: Это год, до которого временная зона активна.

Месяцы: Это список месяцев. Вы можете выбрать любые месяцы.

Дни месяца: Это список дней месяца. Вы можете выбрать любые дни месяцев.

Дни недели: Это список дней недели. Вы можете выбрать любые дни недели.

Комментарий: Это ваш комментарий о текущем элементе временной зоны.

Удаление элемента временной зоны

Вы можете удалить ряд, содержащий информацию об элементе временной зоны, кликнув по кнопке **Удалить выбранный ряд (⊖)**.

19.3.9 Справочник по контекстам

В этой главе говорится о контекстах, переменных, функциях, событиях и действиях, относящихся к контролю доступа.

19.3.9.1 Политики доступа

Этот контекст[41] является контейнером, содержащим все контексты политики доступа[2019] в системе.

Уникальные действия [?[1450]]

СОЗДАТЬ ПОЛИТИКУ ДОСТУПА (Действие по умолчанию[88])

Это действие используется для создания новой политики доступа. Это позволяет пользователю определять основные свойства новой политики доступа и конфигурировать его сразу же после создания.

Тип действия: Создать[105]

Права доступа: Доступно на уровне[486] доступа *Менеджера*

Общие действия [?[1450]]

Создать на основе шаблона[105], Копировать в дочерние контексты[111], Импорт[108], Экспорт[108], Редактировать права доступа[106], Просмотр событий[109], Поиск/фильтрация[110], различные Групповые действия[101], относящиеся к дочерним контекстам.

Иконки и состояния контекста

У этого контекста нет состояний[44]. Его всегда представляет иконка  .

Дополнительная информация

Информация о контексте

Тип контекста[43]: accessPolicies

Имя контекста[42]: access_policies

Описание контекста[43]: Политики доступа

Путь контекста[42]: access_policies

Маска контекста[44]: **access_policies**

Права доступа контекста [?[44]]

Уровень	Описание
Отсутствует	Доступ не разрешён.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт политики доступа.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (Свойства) [?[61]]

У этого контекста нет общих переменных (свойств).

Общие функции [?[70]]

Общие функции: makeCopy (Сделать копию)[71], delete (Удалить)[72]

СОЗДАТЬ

Создает новую политику доступа.

Имя функции:	create
Права доступа:	Доступно на уровне 486 прав доступа *Менеджера*
Количество строк:	1
Формат 49 входных данных:	Такой же, как у переменной childInfo 2020 в контексте Политика доступа 2019.
Записи вывода:	0
Формат 50 выходных данных:	нет

Общие события [? 73]

Общие события: info (Информация) 77

19.3.9.2 Политика доступа

Этот контекст 41 предоставляет вам доступ и позволяет управлять одной из политик доступа 2010.

Уникальные действия [? 1450]

НАСТРОИТЬ (Действие по умолчанию 88)

Это действие используется для редактирования свойств политики доступа.

Изменение поля **Имя** в процессе обработки приведёт к переименованию текущего контекста. Это может привести к неправильной работе других компонентов системы, которые используют имя/путь контекста в качестве первичного идентификатора.

Тип действия: Настроить 105

Общие действия [? 1450]

Удалить 106, Копировать 109, Реплицировать 110, Редактировать права доступа к контексту 106, Просмотр событий 109, Показать статус 111

Иконки и состояния контекста

У этого контекста нет состояний 44. Его всегда представляет иконка 🔑.

Дополнительная информация

Информация о контексте

Тип контекста 43: accessPolicy

Имя контекста 42: назначается пользователем

Описание контекста 43: назначается пользователем

Путь контекста 42: **access_policies.ACCESS_POLICY_NAME** для политики доступа, принадлежащей контексту политики доступа 2019 в корневом контексте

Маска контекста 44: **access_policies.*** для всех политик доступа, принадлежащих к контексту политик доступа 2019 корневого контекста

Права доступа контекста [?44]

Уровень	Описание
Отсутствует	Доступ не разрешён.
Наблюдатель	Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Конфигурация и удаление политик доступа.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (Свойства) [?61]

Общие переменные: groupMembership (членство группы)67, activeAlerts (активные тревоги)

СВОЙСТВА ПОЛИТИКИ ДОСТУПА

Смотрите описание переменной и её полей здесь2010.

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне486 прав доступа *Наблюдателя*, доступно для записи на уровне *Менеджера*.

Формат50 записи:

Имя поля	Тип поля	Примечания
name	Строка	**Имя политики доступа.** Имя этого контекста политики доступа, требующееся для ссылки на данную политику доступа из других частей системы. Оно должно соответствовать правилам именования42 контекста. Длина: 1-50 символов.
description	Строка	Описание.
defaultAction	Целое	**Тип**. 0(Разрешить) или 1(Отклонить)

ТАБЛИЦА ДОСТУПА

Смотрите описание переменной и её полей здесь2011.

Имя переменной: accessTable

Записи: 0...unlimited

Права доступа: Доступно для чтения на уровне486 прав доступа *Наблюдателя*, доступно для записи на уровне *Менеджера*.

Формат50 записи:

Имя поля	Тип поля	Примечания
accessPoint	Строка	Имя устройства Контроля доступа, используемое для текущей записи доступа.

readers	Таблица	Список считывателей из данного устройства Контроля доступа.
timezone	Строка	Имя временной зоны2037, используемое для текущей записи доступа.
action	Целое	**Тип**. 0(Разрешить) или 1(Отклонить)

Общие функции [?70]

У этого контекста нет общих функций.

Общие события [?73]

Общие события: info (Информация)77

19.3.9.3 Сотрудники

Этот контекст41 является контейнером, содержащим контексты сотрудников2063 на определённом уровне организационной иерархии или же все контексты пользователей картами вне данной организационной иерархии.

Уникальные действия[?1450]

СОЗДАТЬ СОТРУДНИКА (Действие по умолчанию88)

Это действие используется для создания нового сотрудника, что позволяет определить основные свойства нового сотрудника и конфигурировать его сразу же после создания.

Тип действия: Создать105

Права доступа: Открыт на уровне486 прав доступа *Менеджера*

Общие действия [?1450]

Создать на основе шаблона105, Копировать в дочерние контексты111, Импорт108, Экспорт108, Редактировать права доступа106, Мониторинг относительных событий109, Просмотр событий109, Поиск/фильтрация110, различные Групповые действия101, относящиеся к дочерним контекстам.

Иконки и состояния контекста

У этого контекста нет состояний44. Его всегда представляет иконка  .

Дополнительная информация

Информация о контексте

Тип контекста43: сотрудники

Имя контекста42: cardholders

Описание контекста43: Сотрудники

Путь контекста42: cardholders, organizations.ORGANIZATION_NAME.cardholders, organizations.ORGANIZATION_NAME.divisions.DIVISION_NAME.cardholders, organizations.ORGANIZATION_NAME.divisions.DIVISION_NAME.departments.DEPARTMENT_NAME.cardholders

Маска контекста44: **cardholders, organizations.*.cardholders** для всех держателей во всех организациях, **organizations.*.divisions.*.cardholders** для всех держателей во всех подразделениях всех организаций, **organizations.*.divisions.*.departments.*.cardholders** для пользователей карт всех отделов во всех подразделениях всех организаций.

Права доступа контекста [?44]

Уровень	Описание
Отсутствует	Доступ не разрешён.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт, импорт Сотрудника.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (Свойства) [? 61]

СМЕНА

Эта переменная определяется только для "корневого" узла сотрудников, которые не относятся ни к одной организации.

Имя переменной: shift

Записи: 1

Permissions: Доступно для чтения на уровне 486 права доступа *Наблюдателя,* доступно для записи на уровне прав доступа *Менеджера*

Формат 50 записи:

Имя поля	Тип поля	Примечания
shift	Строка	Смена 2051 для всех дочерних пользователей карт.

Общие функции [? 70]

Общие функции: makeCopy (сделать копию) 71, delete (удалить) 72

СОЗДАТЬ

Создает нового сотрудника.

Имя функции: create

Права доступа: Доступно на уровне 486 с правом доступа *Менеджер.*

Входные записи: 1

Формат 49 входных данных: Такой же формат как и у переменной childInfo 2064 в контексте Сотрудник 2063.

Выходные записи: 0

Формат 50 выходных данных: нет

Общие события [? 73]

Общие события: info (информация) 77

19.3.9.4 Сотрудник

Этот контекст[41] предоставляет Вам доступ к одному из сотрудников[2043] и позволяет управлять этим узлом.

Уникальные действия [?[1450]]

НАСТРОИТЬ (Действие по умолчанию[88])

Это действие используется для редактирования свойств сотрудника.

Изменение поля **Имя** в процессе обработки приведёт к переименованию текущего контекста. Это может привести к неправильной работе других компонентов системы, которые используют имя/путь контекста в качестве первичного идентификатора.

Тип действия: Конфигурировать[105]

Общие действия [?[1450]]

Удалить[106], Копировать[109], Реплицировать[110], Редактировать права доступа к контексту[106], Просмотр событий[109], Показать статус[111]

Иконки и состояния контекста

У этого контекста нет состояний[44]. Его всегда представляет иконка  .

Дополнительная информация

Информация о контексте

Тип контекста[43]: сотрудник

Имя контекста[42]: предоставляется пользователем

Описание контекста[43]: предоставляется пользователем

Путь контекста[42]: **cardholders.CARDHOLDER_NAME** для владельцев карт, принадлежащих к контексту сотрудники[2061] в корневом контексте, **organizations.ORGANIZATION_NAME.cardholders.CARDHOLDER_NAME** для владельцев карт, принадлежащих к контексту сотрудников в определённом организационном контексте, **organizations.ORGANIZATION_NAME.divisions.DIVISION_NAME.cardholders.CARDHOLDER_NAME** для владельцев карт, принадлежащих к контексту сотрудников в определённом контексте подразделения, **organizations.ORGANIZATION_NAME.divisions.DIVISION_NAME.departments.DEPARTMENT_NAME.cardholders.CARDHOLDER_NAME** для владельцев карт, принадлежащих к контексту сотрудников в определённом контексте отдела

Маска контекста[44]: **cardholders.*** для всех владельцев карт, принадлежащих к контексту сотрудники[2061] в корневом контексте, **organizations.*.cardholders.*** для всех владельцев карт, принадлежащих к контексту сотрудников в определённом организационном контексте, **organizations.*.divisions.*.cardholders.*** для всех владельцев карт, принадлежащих к контексту сотрудников в определённом контексте подразделения, **organizations.*.divisions.*.departments.*.cardholders.*** для всех владельцев карт, принадлежащих к контексту сотрудников в определённом контексте отдела

Права доступа контекста [?[44]]

Уровень	Описание
Отсутствует	Доступ не разрешён.
Наблюдатель	Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Конфигурация и удаление сотрудника.

Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (Свойства)[?61]

Общие переменные: groupMembership (членство группы)67, activeAlerts (активные тревоги)

СВОЙСТВА СОТРУДНИКА

Смотрите описание переменной и её полей здесь2043.

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне486 прав доступа *Наблюдателя,* доступно для записи на уровне *Менеджера*

Формат50 записи:

Имя поля	Тип поля	Примечания
name	Строка	**Имя сотрудника.** Имя данного контекста сотрудника, требуемое для ссылки на данного держателя карты из других частей системы. Оно должно удовлетворять правилам именования42 контекста. Длина: 1-50 знаков.
firstname	Строка	Имя
lastname	Строка	Фамилия
gender	Целое	Пол
position	Строка	Должность
country	Целое	Страна
region	Логическое	Область/Штат/Провинция/Район
zip	Строка	ZIP/Почтовый код
city	Строка	Город
address1	Строка	Адрес 1
address2	Строка	Адрес 2
email	Строка	E-mail адрес
workPhone	Строка	Рабочий телефон
homePhone	Строка	Домашний телефон
mobilePhone	Строка	Мобильный телефон
notes	Строка	Заметки
birthDate	Дата	Дата рождения

disabled	Логическое	Неактивный
emergency	Строка	Информация о контактном лице при чрезвычайных ситуациях

ИДЕНТИФИКАЦИОННЫЕ КАРТОЧКИ

Смотрите описание переменной и её полей здесь 2048.

Имя переменной: cards

Записи: 0...unlimited

Права доступа: Доступно для чтения на уровне 486 права доступа *Наблюдателя,* доступно для записи на уровне прав доступа *Менеджера*

Формат 50 записи:

Имя поля	Тип поля	Примечания
id	Строка	ID идентификационной карточки
status	Целое	Статус
activationDate	Дата	Дата активации
deactivationDate	Дата	Дата деактивации

ФОТОГРАФИЯ

Имя переменной: photo

Записи: 1

Права доступа: Доступно для чтения на уровне 486 права доступа *Наблюдателя,* доступно для записи на уровне прав доступа *Менеджера*

Формат 50 записи:

Имя поля	Тип поля	Примечания
photo	Блок данных	Использует редактор изображений.

ГОЛОСОВАЯ МЕТКА

Имя переменной: voice

Записи: 1

Права доступа: Доступно для чтения на уровне 486 права доступа *Наблюдателя,* доступно для записи на уровне прав доступа *Менеджера*

Формат 50 записи:

Имя поля	Тип поля	Примечания

voice	Блок данных	Использует редактор звука.

ОСОБАЯ РАБОЧАЯ СМЕНА

Смотрите описание переменной и её полей здесь 2048.

Имя переменной: customShift

Записи: 1

Права доступа: Доступно для чтения на уровне 486 права доступа *Наблюдателя,* доступно для записи на уровне прав доступа *Менеджера*

Формат 50 записи:

Имя поля	Тип поля	Примечания
customShift	Строка	Смена 2070 пользователя карты.

ИСКЛЮЧЕНИЯ ПОСЕЩАЕМОСТИ

Смотрите описание переменной и её полей здесь 2048.

Имя переменной: attendance

Записи: 1

Права доступа: Доступно для чтения на уровне 486 права доступа *Наблюдателя,* доступно для записи на уровне прав доступа *Менеджера*

Формат 50 записи:

Имя поля	Тип поля	Примечания
startDate	Дата	Используется редактор дат.
endDate	Дата	Используется редактор дат.
isWorkday	Логическое	
reportingTime	Дата	Используется редактор времени.
knockOffTime	Дата	Используется редактор времени.
hasLunch	Логическое	
lunchBeginHour	Дата	Используется редактор времени.
lunchEndHour	Дата	Используется редактор времени.
maximumOT	Длинное	
minimumOT	Длинное	
category	Целое	

comment	Строка	

Общие функции [?70]

У этого контекста нет общих функций.

Общие события [?73]

Общие события: info (Информация)77

19.3.9.5 Отделы

Этот контекст41 является контейнером, содержащим все контексты отдела2068 в соответствующем подразделении.

Уникальные действия [?1450]

СОЗДАТЬ ОТДЕЛ (Действие по умолчанию88)

Это действие используется для создания нового отдела2042, что позволяет определить основные свойства нового отделения и настроить его сразу же после создания.

Тип действия: Создать105

Права доступа: Доступно на уровне486 доступа *Менеджера*

Общие действия [?1450]

Создать на основе шаблона105, Копировать в дочерние контексты111, Импорт108, Экспорт108, Редактировать права доступа106, Мониторинг относительных событий109, Просмотр событий109, Поиск/фильтрация110, различные Групповые действия101, относящиеся к дочерним контекстам.

Иконки и состояния контекста

У этого контекста нет состояний44. Его всегда представляет иконка  .

Дополнительная информация

Информация о контексте

Тип контекста43: отделы

Имя контекста42: departments

Описание контекста43: Отделы

Путь контекста42: organizations.ORGANIZATION_NAME.divisions.DIVISION_NAME.departments

Маска контекста44: organizations.*.divisions.*.departments

Права доступа контекста [?44]

Уровень	Описание
Отсутствует	Доступ не разрешён.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт отделов.

Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (Свойства) [? 61]

У этого контекста нет общих переменных (свойств).

Общие функции [? 70]

Общие функции: makeCopy (сделать копию) 71, delete (удалить) 72

СОЗДАТЬ

Создать новый отдел.

Имя функции:	create
Права доступа:	Доступно на уровне 486 доступа *Менеджера*
Записи ввода:	1
Формат 49 входных данных:	Такой же формат как и у переменной childInfo 2064 в контексте Department 2068.
Записи вывода:	0
Формат 50 выходных данных:	нет

Общие события [? 73]

Общие события: info (Информация) 77

19.3.9.6 Отдел

Этот контекст 41 предоставляет Вам доступ к одному из отделов 2042 и позволяет управлять им.

Уникальные действия [? 1450]

НАСТРОИТЬ (Действие по умолчанию 88)

Это действие используется для редактирования свойств отдела.

Изменение поля **Имя** в процессе обработки приведёт к переименованию текущего контекста. Это может привести к неправильной работе других компонентов системы, которые используют имя/путь контекста в качестве первичного идентификатора.

Тип действия: Настроить 105

Общие действия [? 1450]

Удалить 106, Копировать 109, Реплицировать 110, Редактировать права доступа к контексту 106, Просмотр событий 109, Показать статус 111

Иконки и состояния контекста

У этого контекста нет состояний 44. Его всегда представляет иконка .

Дополнительная информация

Информация о контексте

Тип контекста[43]: отдел

Имя контекста[42]: предоставляется пользователем

Описание контекста[43]: предоставляется пользователем

Путь контекста[42]: organizations.ORGANIZATION_NAME.divisions.DIVISION_NAME.departments.DEPARTMENT_NAME

Маска контекста[44]: organizations.*.divisions.*.departments.*

Права доступа контекста [?[44]]

Уровень	Описание
Отсутствует	Доступ не разрешён.
Наблюдатель	Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Создание, экспорт и импорт отдела.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (Свойства) [?[61]]

Общие переменные: groupMembership (Членство группы)[67], activeAlerts (Активные тревоги)

СВОЙСТВА

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне[486] прав доступа *Наблюдателя,* доступно для записи на уровне *Менеджера*

Формат[50] записи:

Имя поля	Тип поля	Примечания
name	Строка	**Имя отдела**. Имя контекста отдела, требуемое для ссылки на данный отдел из других частей системы. Оно должно удовлетворять правилам именования[42] контекста. Длина: 1-50 знаков.
description	Строка	**Описание отдела**. Длинна: 1-100 символов.

СМЕНА

Имя переменной: customShift

Записи: 1

Права доступа: Доступно для чтения на уровне[486] права доступа *Наблюдателя,* доступно для записи на уровне прав доступа *Менеджера*

Формат[50] записи::

Имя поля	Тип поля	Примечания
customShift	Строка	Смена 2070 пользователя карты.

Общие функции [? 70]

У этого контекста нет общих функций.

Общие события [? 73]

Общие события: info (Информация) 77

19.3.9.7 Подразделения

Этот контекст 41 является контейнером, содержащим все контексты подразделения 2072 в соответствующей организации.

Уникальные действия [? 1450]

СОЗДАТЬ ПОДРАЗДЕЛЕНИЕ (Действие по умолчанию)

Это действие используется для создания нового подразделения 2042, что позволяет определить основные свойства нового подразделения и настроить его сразу же после создания.

Тип действия: Создать 105

Права доступа: Доступно на уровне 486 доступа *Менеджера*

Общие действия [? 1450]

Создать на основе шаблона 105, Копировать в дочерние контексты 111, Импорт 108, Экспорт 108, Редактировать права доступа 106, Мониторинг относительных событий 109, Просмотр событий 109, Поиск/фильтрация 110, различные Групповые действия 101, относящиеся к дочерним контекстам.

Иконки и состояния контекста

У этого контекста нет состояний 44. Его всегда представляет иконка  .

Дополнительная информация

Инфомация о контексте

Тип контекста 43: подразделения

Имя контекста 42: divisions

Описание контекста 43: Подразделения

Путь контекста 42: organizations.ORGANIZATION_NAME.divisions

Маска контекста 44: organizations.*.divisions

Права доступа контекста [? 44]

Уровень	Описание
Отсутствует	Доступ не разрешён.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.

Менеджер	Создание, экспорт и импорт подразделений.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (Свойства) [?61]

У этого контекста нет общих переменных (свойств).

Общие функции [?70]

Общие функции: makeCopy (сделать копию)71, delete (удалить)72

СОЗДАТЬ

Создает новое подразделение.

Имя функции:	create
права доступа:	Доступно на уровне486 доступа *Менеджера*
Записи ввода:	1
Формат49 входных данных:	Такой же формат как и у переменной childInfo2064 в контексте Department2068.
Записи вывода:	0
Формат50 выходных данных:	нет

Общие события [?73]

Общие события: info (Информация)77

19.3.9.8 Подразделение

Этот контекст41 предоставляет Вам доступ к одному подразделению2042 и позволяет им управлять.

Уникальные действия [?1450]

РЕДАКТИРОВАТЬ (Действие по умолчанию88)

Это действие используется для редактирования свойств подразделения.

Изменение поля **Имя** в процессе обработки приведёт к переименованию текущего контекста. Это может привести к неправильной работе других компонентов системы, которые используют имя/путь контекста в качестве первичного идентификатора.

Тип действия: Редактировать105

Общие действия [?1450]

Удалить106, Копировать109, Реплицировать110, Редактировать права доступа к контексту106, Просмотр событий109, Показать статус111

Иконки и состояния контекста

У этого контекста нет состояний44. Его всегда представляет иконка  .

Дополнительная информация

Информация о контексте

Тип контекста 43: подразделение

Имя контекста 42: предоставлено пользователем

Описание контекста 43: предоставлено пользователем

Путь контекста 42: organizations.ORGANIZATION_NAME.divisions.DIVISION_NAME

Маска контекста 44: organizations.*.divisions.*

Права доступа контекста [? 44]

Уровень	Описание
Отсутствует	Доступ не разрешён.
Наблюдатель	Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Создание и удаление подразделений.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (Свойства) [? 61]

Общие переменные: groupMembership (членство группы) 67, activeAlerts (активные тревоги)

СВОЙСТВА ПОДРАЗДЕЛЕНИЯ

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне 486 прав доступа *Наблюдателя,* доступно для записи на уровне *Менеджера*

Формат 50 записи:

Имя поля	Тип поля	Заметки
name	Строка	**Имя подразделения**. Имя контекста подразделения, требуемое для ссылки на данный отдел из других частей системы. Оно должно удовлетворять правилам именования 42 контекста. Длина: 1-50 знаков.
description	Строка	**Описание подразделения**. Длина: 1-100 знаков.

СМЕНА

Имя переменной: customShift

Записи: 1

Права доступа: Доступно для чтения на уровне 486 права доступа *Наблюдателя,* доступно для записи на уровне прав доступа *Менеджера*

Формат 50 записи:

Имя поля	Тип поля	Заметки
customShift	Строка	Смена 2051 подразделения.

Общие функции [? 70]

У этого контекста нет общих функций.

Общие события [? 73]

Общие события: info (Информация) 77

19.3.9.9 Организации

Этот контекст 41 является контейнером, который содержит все контексты организации 2075 в системе.

Уникальные действия [? 1450]

СОЗДАТЬ ОРГАНИЗАЦИЮ (Действие по умолчанию 88)

Это действие используется для создания новой организации. Оно позволяет пользователю задать основные свойства для новой организации и настроить её сразу после создания.

Тип действия: Создать 105

Права доступа: Доступно на уровне 486 с правами доступа *Менеджера.*

Общие действия [? 1450]

Создать на основе шаблона 105, Копировать в дочерние контексты 111, Импорт 108, Экспорт 108, Редактировать права доступа 106, Просмотр событий 109, Поиск/фильтрация 110, различные Групповые действия 101, относящиеся к дочерним контекстам.

Состояния и иконки контекста

У этого контекста нет состояний 44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста 43: организации

Имя контекста 42: organizations

Описание контекста 43: Организации

Путь Контекста 42: организации

Маска контекста 44: организации

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Доступ не разрешён.

Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт организации.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (Свойства) [?61]

У этого контекста нет общих переменных (свойств).

Общие функции [?70]

Общие функции: makeCopy (сделать копию)71, delete (удалить)72

СОЗДАТЬ

Создает новую организацию.

Имя функции:	create
Права доступа:	Доступно на уровне486 с правами доступа для *Менеджера.*
Записи ввода:	1
Формат49 входных данных:	Тот же самый формат, что и у переменной childInfo2076 в контексте Организация2075.
Записи вывода:	0
Формат50 выходных данных:	нет

Общие события [?73]

Общие события: info (Информация)77

19.3.9.10 Организация

Этот контекст41 предоставляет Вам доступ к одной организации2042 и позволяет им управлять.

Уникальные действия [?1450]

НАСТРОИТЬ (Действие по умолчанию88)

Это действие используется для редактирования свойств организации.

Изменение поля **Имя** в процессе обработки приведёт к переименованию текущего контекста. Это может привести к неправильной работе других компонентов системы, которые используют имя/путь контекста в качестве первичного идентификатора.

Тип действия: Настроить105

Общие действия [?1450]

Удалить106, Копировать109, Реплицировать110, Редактировать права доступа к контексту106, Просмотр событий109, Показать статус111

Иконки и состояния контекста

У этого контекста нет состояний[44]. Его всегда представляет иконка  .

Информация о контексте

Тип контекста[43]: организация

Имя контекста[42]: предоставлено пользователем

Описание контекста[43]: предоставлено пользователем

Путь контекста[42]: organizations.ORGANIZATION_NAME

Маска контекста[44]: organizations.*

Права доступа к контексту [?[44]]

Уровень	Описание
Отсутствует	Доступ не разрешён.
Наблюдатель	Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Создание, экспорт и импорт организации.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (Свойства) [?[61]]

Общие переменные: groupMembership (Членство группы)[67], activeAlerts (Активные тревоги)

СВОЙСТВА ОРГАНИЗАЦИИ

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне[486] прав доступа *Наблюдатель,* доступно для записи на уровне *Менеджера.*

Формат[49] записи:

Имя поля	Тип поля	Примечания
name	строка	**Имя организации**. Имя контекста организации, необходимое для обращения к данной организации из других частей системы. Оно должно соответствовать правилам именования[42] контекста. Длина 1-50 символов.
description	строка	**Описание организации**. Длина: 1-100 символов.

СМЕНА

Имя переменной: смена

Записи: 1

Права доступа: Доступно для чтения на уровне[486] с правами доступа для *Наблюдатель,* доступно для записи на уровне с правами доступа *Менеджера.*

Формат[49] записи:

Имя поля	Тип поля	Примечания

shift	строка	Смена 2051 организации

Общие функции [? 70]

У этого контекста нет общих функций.

Общие события [? 73]

Общие события: info (Информация) 77

19.3.9.11 Временные зоны

Этот контекст 41 является контейнером, который содержит все контексты временных зон 2037 в системе.

Уникальные действия [? 1450]

СОЗДАТЬ ВРЕМЕННУЮ ЗОНУ (Действие по умолчанию 88)

Это действие используется для создания новой временной зоны. Оно позволяет пользователю задавать основные свойства для новой временной зоны и настраивать её сразу после создания.

Тип действия: Создать 105

Права доступа: Доступно на уровне 486 с правами доступа *Менеджера.*

Общие действия [? 1450]

Создать на основе шаблона 105, Копировать в дочерние контексты 111, Импорт 108, Экспорт 108, Редактировать права доступа 106, Просмотр событий 109, Поиск/фильтрация 110, различные Групповые действия 101, относящиеся к дочерним контекстам.

Состояния и иконки контекста

У этого контекста нет состояний 44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Context Type 43: временные зоны

Имя контекста 42: timezones

Описание контекста 43: Временные зоны

Путь Контекста 42: Временные зоны

Маска контекста 44: **timezones**

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Доступ не разрешён.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт временной зоны.

Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (Свойства) [?61]

У этого контекста нет общих переменных (свойств).

Общие функции [?70]

Общие функции: makeCopy (сделать копию)71, delete (удалить)72

СОЗДАТЬ

Создает новую временную зону.

Имя функции:	create
Права доступа:	Доступно на уровне486 прав доступа *Менеджера.*
Записи ввода:	1
Формат49 входных данных:	Тот же самый формат, что у переменной childInfo2076 в контексте Временной зоны2037.
Записи вывода:	0
Формат50 выходных данных:	нет

Общие события [?73]

Общие события: info (Информация)77

19.3.9.12 Временная зона

Этот контекст41 предоставляет Вам доступ к одной временной зоне2037 и позволяет ей управлять.

Уникальные действия [?1450]

НАСТРОИТЬ (Действие по умолчанию88)

Это действие используется для редактирования свойств временной зоны.

Изменение поля **Имя** в процессе обработки приведёт к переименованию текущего контекста. Это может привести к неправильной работе других компонентов системы, которые используют имя/путь контекста в качестве первичного идентификатора.

Тип действия: Настроить105

Общие действия [?1450]

Удалить106, Копировать109, Реплицировать110, Редактировать права доступа к контексту106, Просмотр событий109, Показать статус111

Иконки и состояния контекста

У этого контекста нет состояний44. Его всегда представляет иконка  .

Дополнительная информация

Информация о контексте

Имя контекста[43]: временная зона

Имя контекста[42]: предоставлено пользователем

Описание контекста[43]: предоставлено пользователем

Путь Контекста[42]: **timezones.TIMEZONE_NAME** для временных зон, принадлежащих к контексту временные зоны[2036] корневого контекста

Маска контекста[44]: **timezones.*** для всех временных зон, принадлежащих к к контексту временные зоны[2036] корневого контекста

Права доступа к контексту [?[44]]

Уровень	Описание
Отсутствует	Доступ не разрешён.
Наблюдатель	Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Создание, экспорт и импорт временных зон.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (Свойства) [?[61]]

Общие переменные: groupMembership (членство группы)[67], activeAlerts (активные тревоги)

СВОЙСТВА ВРЕМЕННОЙ ЗОНЫ

Смотрите описание переменной и её полей здесь[2016].

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне[486] с правами доступа *Наблюдатель,* доступно для записи на уровне с правами доступа *Менеджера.*

Формат[49] записи:

Имя поля	Тип поля	Примечания
name	Строка	**Имя временной зоны.** Имя контекста временной зоны, необходимое для обращения к данной временной зоне из других частей системы. Оно должно соответствовать правилам именования[42] контекста. Длина 1-50 символов.
description	Строка	Описание данной временной зоны.

TIMEZONE TABLE

Смотрите описание переменной и её полей здесь[2016].

Имя переменной: timeTable

Записи: 0...unlimited

Права доступа: Доступно для чтения на уровне[486] с правами доступа *Наблюдатель,* доступно для записи на уровне с правами доступа *Менеджера.*

Формат 49 записи:

Имя поля	Тип поля	Примечания
fromTime	Данные	Временная зона начинается с этого времени. Используется редактор даты.
toTime	Данные	Временная зона заканчивается этим временем. Используется редактор даты.
fromYear	Целое	Временная зона начинается с этого года.
toYear	Целое	Временная зона заканчивается этим годом.
months	Таблица	Таблица содержит все месяцы. Возможен выбор данных месяцев в любой вариации.
monthDays	Таблица	Таблица содержит все дни месяца. Возможен выбор данных дней в любой вариации.
weekDays	Таблица	Таблица содержит все дни недели. Возможен выбор данных дней в любой вариации.
comment	Строка	Комментарий для данной временной зоны.

Общие функции [? 70]

У этого контекста нет общих функций.

Общие события [? 73]

Общие события: info (Информация) 77

19.4 Учет рабочего времени

В этой главе говорится о **AtomMind Time and Attendance**, мультипользовательской системе учета рабочего времени сотрудников, основанной на AtomMind.

Чтобы продолжить, перейдите к обзору 2039.

19.4.1 Обзор

AtomMind Time and Attendance - это многопользовательская система учета рабочего времени сотрудников, основанная на ядре AtomMind. Она тесно взаимосвязана с **AtomMind Access Control** и другими решениями AtomMind, разработанными для управления умным домом.

Основные возможности AtomMind Time and Attendance

- Обработка событий посещаемости от различных типов устройств учета рабочего времени
- Улучшенный сервис оповещения, журналирования и обработки событий
- Встроенные отчеты (посещение, пунктуальность, ежедневная активность, кто в здании/вне здания и т.д.)
- Отправка отчетов по E-mail
- Поддержка недельных и стандартных смен
- Настраиваемые правила для обработки времени обеда/перерыва и сверхурочного времени
- Поддержка крупных организаций со сложными многоступенчатыми иерархиями персонала
- Импорт данных о сотруднике/владельце смарт карты

- Лучшие в данной области возможности обработки данных для больших сетей различных устройств учета рабочего времени
- Ролевая модель управления доступом и контроль масштабируемости предприятия
- Интегрированные GUI-интерфейс и Редактор Отчетов
- Интеграция со сторонними системами (платежные ведомости, контроль доступа и пр.) через открытые протоколы
- Легкое внедрение и администрирование

19.4.2 Подключение устройств учета рабочего времени

Одна из лучших характеристик AtomMind Time and Attendance – это то, что система очень открыта и позволяет работать с различными типами терминалов учета рабочего времени:

- Радиоидентификация
- Магнитные карты
- Штрихкод
- Биометрия
- Клавиатура
- Счетчик времени на ПК
- Счетчик времени PDA
- Онлайн счетчик времени

Встроенные устройства AtomMind: Некоторые устройства, такие как устройство учета рабочего времени TR610 GigaTMS, были "созданы для AtomMind". У таких устройств настройки AtomMind являются частью их собственного меню конфигурации, которые оперативно выполняют подключение.

Сетевые устройства учета рабочего времени: Если у вас есть устройство учета, подключенное к сети (или сеть из нескольких устройств), которые вы бы хотели подключить к AtomMind, вам нужно лишь создать драйвер устройства 518 AtomMind для них. Это может быть компонент ПО AtomMind, который позволяет работать с данными устройствами при помощи родного приложения. Конечно, вы можете также создать (или купить) несколько различных драйверов для подключения различных устройств учета от разных производителей и объединить их в единую систему.

Подключение терминалов напрямую: Если вы хотите, чтобы сотрудники регистрировали приход на работу и уход с нее, используя простое устройство получения данных, такое как магнитная полоса или считыватель штрихкода, вы можете соединить их с помощью AtomMind 684 Agent. Это небольшое аппаратное устройство, на котором работает специальное приложение ТВЭЛ BASIC, которое может общаться с AtomMind. Оно выполняет основные действия с простым конечным узлом: регистрирует его в AtomMind Server, передает полученные данные, и т.д.

Как подключить терминал учета рабочего времени

Система работает так:

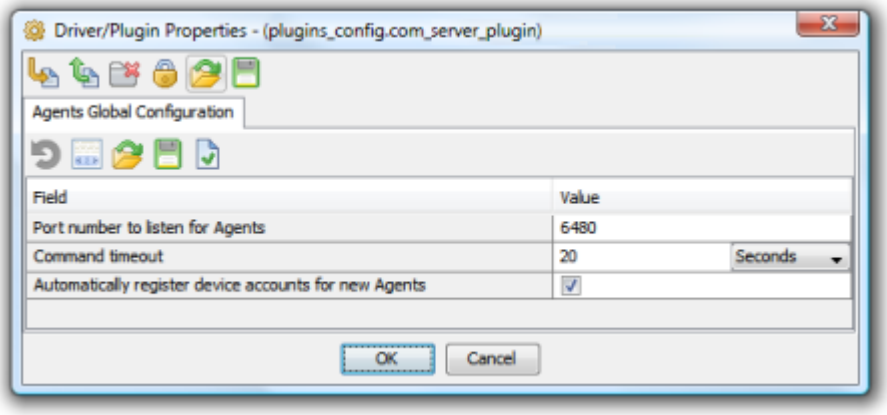
- AtomMind прослушивает входящие подключения.
- Новое устройство пытается подключиться, заявляя, что оно управляется пользователем 478 "admin" (например). Оно также говорит AtomMind, что это – устройство учета рабочего времени.
- AtomMind проверяет существование аккаунта данного пользователя.
- Если аккаунт пользователя найден, AtomMind создает узел (также называемый "аккаунт устройства") для нового устройства данного пользователя.
- Теперь пользователь может работать с устройством, получать его статусы, изменять его настройки, и т.д.

Итак, повторим вкратце все положения: подключения всегда инициируются *от* устройства *к* AtomMind Server. Это означает, что для конфигурации устройства, вам следует знать несколько вещей о вашем AtomMind Server:

- Его IP-адрес,
- Какой порт он прослушивает для поиска новых подключений,
- Убедитесь, что порт настроен правильно для организации прослушивания.
- Под каким именем устройство должно находиться в списке (должно быть пользователем, который уже существует на вашем AtomMind Server).

Для IP -адреса: Вам необходимо выяснить IP-адрес вашего AtomMind Server. Вы, возможно знаете, как это делать (`ipconfig` на Windows, `ifconfig` на Linux; сложнее, если ваше входящее подключение исходит из локального LAN). Это касается больше настроек сети, чем настроек AtomMind. В конечном итоге, вам нужно знать, к какому IP-адресу хочет подключиться ваше устройство, чтобы получить доступ к вашему AtomMind Server.

Другие детали перечислены в **Драйверы/Конфигурация плагинов > AtomMind Agent**. Обратитесь к этому узлу в вашем системном дереве и дважды кликните по нему (или щелкните правой кнопкой мыши и выберите **Редактировать драйвер/Свойства плагина.** Вы увидите следующее диалоговое окно:



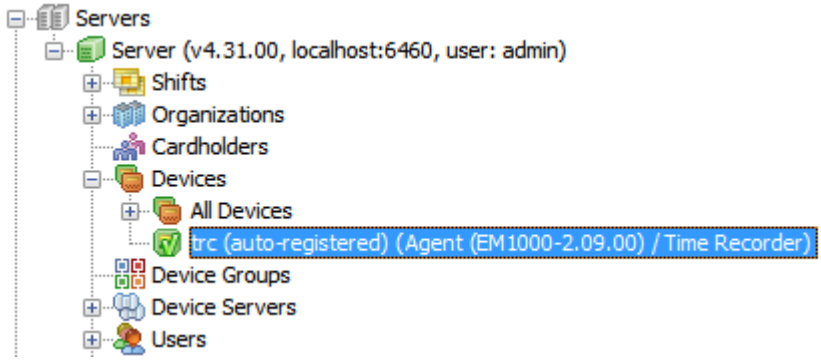
Во-первых, обратите внимание на **Номер порта для прослушивания Агентов** - по умолчаию 6480. Однако, ваш номер может быть иным. Затем, убедитесь, что проверена **Автоматическая регистрация аккаунтов устройств для новых агентов,** чтобы новые устройства могли регистрироваться в системе. В будущем, как только вы зарегистрируете все ваши устройства, вы можете отключить данную настройку, чтобы новые устройства не могли регистрироваться без вашего (системного администратора) ведома.

Еще одна деталь, которую вам нужно включить в конфигурацию вашего устройства – это **аккаунт пользователя**, под которым устройство должно быть зарегистрировано. Другими словами, AtomMind требует наличия устройства, которое сообщает ему, какой пользователь им "владеет". Если у вас небольшая инсталляция, вы можете оставить ее имя – **admin**, администратор AtomMind по умолчанию 479.

Теперь, когда вы знаете эти детали, пора применить их к вашему устройству учета рабочего времени (или конфигурации Agent). Мы не можем предоставить вам точные указания здесь, они зависят от вашего устройства. Сейчас самое время обратиться к документации для получения нужных вам советов и выяснения, как это сделать. Как только данное действие выполнено, ваше устройство должно попробовать подключиться к AtomMind Server с корректными параметрами (и сделать это успешно).

Как понять, что устройство успешно подключено

После регистрации устройства, войдите в систему AtomMind Client AtomMind с помощью имени пользователя, под которым вы зарегистрировались. Это означает, что если устройство сказало AtomMind Server, что владелец – **admin**, вы должны войти как **admin**. Затем кликните по Device в системном дереве. Вы должны увидеть что-то похожее на следующее:



Выделенный узел является вашим терминалом (ваш, конечно, будет назван иначе). Галочка рядом с ним показывает, что он подсоединен и находится в режиме синхронизации. В техническом отношении, она показывает *аккаунт* для вашего устройства.

ЧТО ОЗНАЧАЕТ ИКОНКА

Иконка рядом с вашим устройством учета рабочего времени показывает статус синхронизации. Изображение, показанное выше, означает "Синхронизировано от устройства к серверу". Простым языком, это означает, что все считанные (или отсканированные) карты были перемещены к этому моменту в AtomMind Server, и что любые отчеты, сгенерированные вами, отображают информацию, соответствующую текущему моменту.

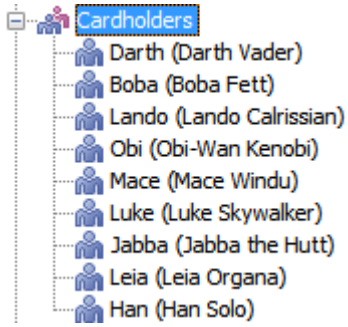
Существует несколько других статусов синхронизации, описанных в Device.

19.4.3 Настройка структуры организации

AtomMind - масштабируемое решение. Это означает, что AtomMind может использовать как компания, состоящая из десяти человек, так и крупное предприятие со штатом в 30,000 сотрудников. Таким образом, у нас есть два основных метода добавления штатных служащих, или, говоря на языке AtomMind *пользователей карт*:

Горизонтальная структура

Этот формат подходит для малых организаций. Если Вы собираетесь использовать учет рабочего времени AtomMind в компании, состоящей из 30 человек, Вы, возможно, не собираетесь разделять ее на отделы и подразделения. Вы, скорее всего, пожелаете внести своих сотрудников в список наиболее простым способом. Для этого предназначен узел **Пользователи карт**, расположенный в системном дереве. Горизонтальная структура выглядит следующим образом:



Иерархическая структура

Эта опция предназначена для более крупных организаций. Она позволяет разделить организацию на три иерархических уровня.

- **Организация ():** это самый верхний уровень. Учитывается вся корпорация целиком. Обратите внимание, что у Вас может быть несколько организаций, поэтому Вы могли бы использовать эту структуру для каждого отдельного физического места (например, для каждого из филиалов международной компании).
- **Подразделение ():** это средний уровень. Сюда может относится, например, R&D или Коммерческий отдел.
- **Отдел ():** Это нижний уровень, например, Отдел по работе с корпоративными клиентами - это отдел, входящий в подразделение "Коммерческий отдел".

Вам не нужно вдаваться в детали. Например, Вы могли бы хранить все информацию о Вашей организации на уровне подразделения. Кроме того, если Вы хотите использовать терминологию, в соответствии с Вашей текущей структурой организации, и разделяете ее на "отделы" без "подразделений", Вы можете создать единую организацию, содержащую одно подразделение, и поместить все отделы в это подразделение. Основное требование заключается в том, что иерархия фиксирована - отдел можно создать *лишь* в рамках подразделения.

Добавление организаций, подразделений и отделов

Для того, чтобы добавить новую организацию, подразделение или отдел, щелкните дважды мышкой по соответствующему узлу. Например, двойной щелчок мышью по узлу ***Организации*** добавил бы новую организацию (Вам необходимо лишь выбрать для нее **имя** и **описать** ее). После того, как Вы создали организацию, Вам следует кликнуть дважды по ней, чтобы добавить подразделение. Аналогичным образом создаются отделы.

ГРУППЫ

Вы можете создать группы подразделений, отделов и владельцев карт для использования в отчетах и выполнения различных групповых операций. Группы 751 описаны в соответствующей теме.

ТИПОВЫЕ ДЕЙСТВИЯ

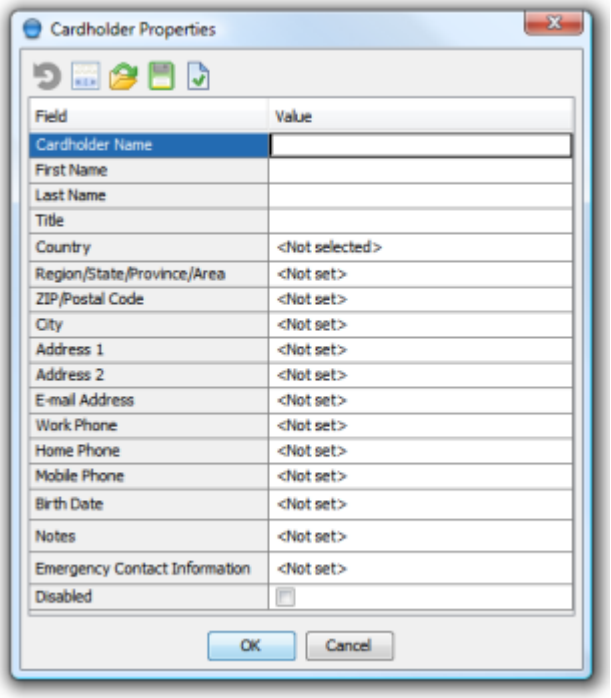
Создание, настройка и удаление контекстов (контекст 41 - это типичное название для объектов AtomMind, таких как организации, подразделения или отделы) называются *типовыми действиями.* Существует ряд других действий, которые Вы можете выполнить практически с любым контекстом в AtomMind. К таким действиям относится копирование контекста, редактирование разрешений к его доступу, мониторинг относящихся к нему событий и пр., эти действия включены в типовые действия 103. Обычно они не используются при типовом использовании системы.

19.4.4 Управление сотрудниками

Чтобы создать нового сотрудника, сделайте одно из следующих действий:

- Щёлкните дважды по корню узла **Сотрудники** или
- Щёлкните дважды по узлу **Сотрудники** интересующего Вас отдела, подразделения или организации чтобы внести информацию о новом сотруднике в этой административной единице. У Вас также есть возможность перетащить сотрудника в другое подразделение позже..

Диалог создания нового сотрудника (или редактирования уже существующего) выглядит следующим образом:

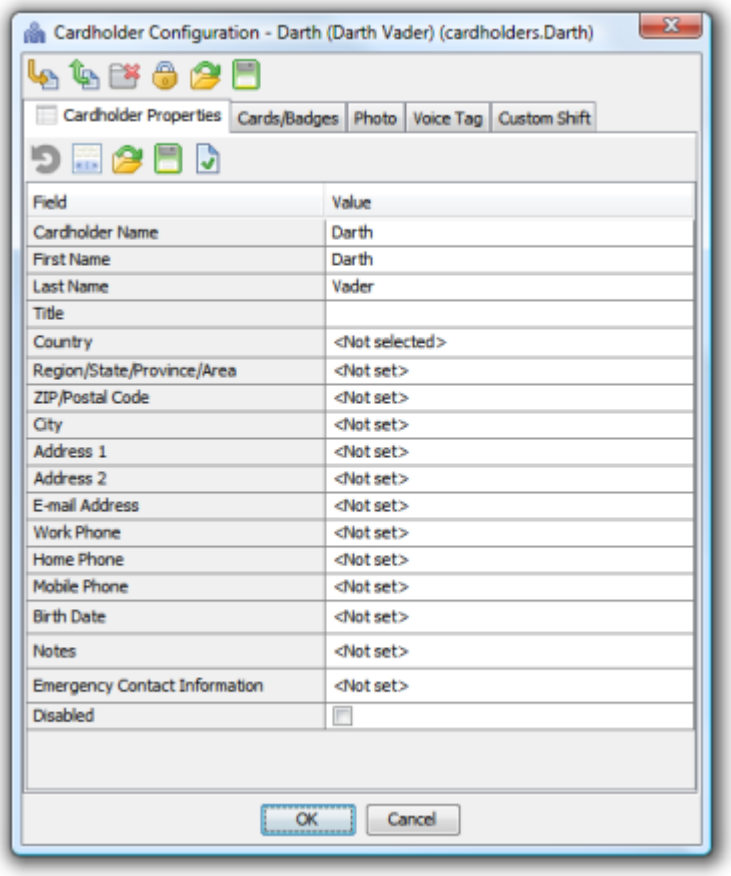


Свойства имеют привычные названия (фамилия, имя, контактный телефон и пр.). Есть только пара замечаний, на которые необходимо обратить внимание:

- **Имя** - имя *контекста* владельца в системе. Нет необходимости указывать реальное имя сотрудника (для этого существуют поля **Фамилия** и **Имя**). Это поле должно соответствовать требованиям к именованию контекстов 42 AtomMind. Звучит, конечно, немного сложно, но обычно это означает, что Вам не стоит использовать пробелы в этом поле и его значение должно быть уникальным для данного уровня иерархии (Вы не можете иметь два контекста *Joe* внутри одного и того же отдела. Одному из них следует дать имя *Joe1*).
- Одна ключевая деталь, которую нельзя указать в этом диалоге - действительный *номер карты,* с сотрудник регистрируется в системе! После добавления каждого сотрудника Вам необходимо отредактировать его свойства и указать номер его карты.
- Вы также можете импортировать 2045 список держателей карт из любой другой программной среды, поддерживающей экспорт в CSV.

Редактирование сотрудников

Когда вы дважды щёлкните по имени сотрудника (или щелкните правой кнопкой мыши и выберите **Настроить**), Вы увидите следующий диалог:



Обратите внимание на дополнительные вкладки в этом диалоге. При добавлении сотрудника Вам был доступна только первая вкладка (свойства). Теперь появилось еще четыре: **Идентификационные карточки, Фотография, Образец голоса** и **Особая рабочая смена**.

Идентификационные карточки: В этой вкладке вводят информацию о карте. Как следует из названия, каждый сотрудник может держать по нескольку карт. Подробнее об этом в Управление картами/бэджиками 2048.

Фотография: Изображение сотрудника.

Образец голоса: Здесь указывается файл с записью образца голоса сотрудника с целью проверок.

Особая рабочая смена: Позволяет Вам указать индивидуальное расписание сотрудника, если оно отличается от общепринятого в его отделе. Подробнее об этом в Управление Расписанием 2017.

Перенос сотрудника или создание по образцу уже существующего

При перетаскивании узла сотрудника в другое место дерева организации создается копия перетаскиваемого узла. После этого Вы можете отредактировать только часть информации (имя, бэджики и пр.), создав таким образом нового сотрудника по образцу уже существующего. После перетаскивания/копирования Вы можете удалить оригинальный узел, что будет эквивалентно переносу в новое местоположение.

Вы также можете реплицировать 1101 сотрудника (или любой другой контекст) или выполнить любые Общие действия 103 на нём.

Выбор множества сотрудников

Вы можете выбрать множество сотрудников (через Shift+клик для выделения соседей и Ctrl+клик для индивидуального выбора сотрудников в различных местах поддерева). Если следом за этим Вы кликните правой кнопкой мыши по выделенным записям, Вы сможете отредактировать их все (или удалить их все или пр.).

Пользовательские свойства сотрудников

Возможно, Ваша организация требует учёта некоторых свойств сотрудников, которые по умолчанию не представлены в AtomMind. Например, номер водительской лицензии или номер парковочного места или таблица данных об имуществе, за которое отвечает сотрудник (если сотрудник работает вне офиса и подключается к системе удаленно).

К счастью, Вы можете добавлять пользовательские свойства каждому сотруднику. Почитайте об этом в обучающем руководстве Добавление пользовательских свойств 1686.

19.4.4.1 Импорт сотрудников

В случае, если Вы производите миграцию с Вашей старой системы учета рабочего времени на AtomMind Time and Attendance и Ваша старая система способна выполнять экспорт списка сотрудников/владельцев карт в формате CSV, AtomMind может помочь Вам сохранить Ваше время с помощью функции импорта этих данных вместо выполнения длительного ввода данных вручную.

Покажем это на практическом примере. Ниже представлен CSV-файл, экспортированный из реально используемой программы учёта рабочего времени. Вы можете скопировать содержимое в файл `employees.csv` и проделать все самостоятельно.

```
EmployeeID,LastName,FirstName,Title,TitleOfCourtesy,BirthDate,HireDate,Address,City,Re
gion,PostalCode,Country,HomePhone,Extension,Photo,Notes,ReportsTo

1,Davolio,Nancy,Sales Representative,Ms.,12/08/48,05/01/92,507 - 20th Ave. E.\r\nApt.
2A,Seattle,WA,98122,USA,(206) 555-9857,5467,Object Type17,Education includes a BA in
psychology from Colorado State University in 1970.  She also completed "The Art of the
Cold Call."  Nancy is a member of Toastmasters International.,2

2,Fuller,Andrew,"Vice President, Sales",Dr.,02/19/52,08/14/92,908 W. Capital
Way,Tacoma,WA,98401,USA,(206) 555-9482,3457,Object Type17,"Andrew received his BTS
commercial in 1974 and a Ph.D. in international marketing from the University of
Dallas in 1981.  He is fluent in French and Italian and reads German.  He joined the
company as a sales representative, was promoted to sales manager in January 1992 and
to vice president of sales in March 1993.  Andrew is a member of the Sales Management
Roundtable, the Seattle Chamber of Commerce, and the Pacific Rim Importers
Association.",

3,Leverling,Janet,Sales Representative,Ms.,08/30/63,04/01/92,722 Moss Bay
Blvd.,Kirkland,WA,98033,USA,(206) 555-3412,3355,Object Type17,Janet has a BS degree in
chemistry from Boston College (1984).  She has also completed a certificate program in
food retailing management.  Janet was hired as a sales associate in 1991 and promoted
to sales representative in February 1992.,2

4,Peacock,Margaret,Sales Representative,Mrs.,09/19/37,05/03/93,4110 Old Redmond
Rd.,Redmond,WA,98052,USA,(206) 555-8122,5176,Object Type17,Margaret holds a BA in
English literature from Concordia College (1958) and an MA from the American Institute
of Culinary Arts (1966).  She was assigned to the London office temporarily from July
through November 1992.,2

5,Buchanan,Steven,Sales Manager,Mr.,03/04/55,10/17/93,14 Garrett Hill,London,,SW1
8JR,UK,(71) 555-4848,3453,Object Type17,"Steven Buchanan graduated from St. Andrews
University, Scotland, with a BSC degree in 1976.  Upon joining the company as a sales
representative in 1992, he spent 6 months in an orientation program at the Seattle
office and then returned to his permanent post in London.  He was promoted to sales
manager in March 1993.  Mr. Buchanan has completed the courses \"Successful
Telemarketing\" and \"International Sales Management.\"  He is fluent in French.",2

6,Suyama,Michael,Sales Representative,Mr.,07/02/63,10/17/93,Coventry House\r\nMiner
Rd.,London,,EC2 7JR,UK,(71) 555-7773,428,Object Type17,"Michael is a graduate of
Sussex University (MA, economics, 1983) and the University of California at Los
Angeles (MBA, marketing, 1986).  He has also taken the courses \"Multi-Cultural
Selling\" and \"Time Management for the Sales Professional.\"  He is fluent in
Japanese and can read and write French, Portuguese, and Spanish.",5

7,King,Robert,Sales Representative,Mr.,05/29/60,01/02/94,Edgeham Hollow\r\nWinchester
Way,London,,RG1 9SP,UK,(71) 555-5598,465,Object Type17,"Robert King served in the
Peace Corps and traveled extensively before completing his degree in English at the
University of Michigan in 1992, the year he joined the company.  After completing a
course entitled \"Selling in Europe,\" he was transferred to the London office in
March 1993.",5

8,Callahan,Laura,Inside Sales Coordinator,Ms.,01/09/58,03/05/94,4726 - 11th Ave.
N.E.,Seattle,WA,98105,USA,(206) 555-1189,2344,Object Type17,Laura received a BA in
psychology from the University of Washington.  She has also completed a course in
business French.  She reads and writes French.,2

9,Dodsworth,Anne,Sales Representative,Ms.,01/27/66,11/15/94,7 Houndstooth
Rd.,London,,WG2 7LT,UK,(71) 555-4444,452,Object Type17,Anne has a BA degree in English
from St. Lawrence College.  She is fluent in French and German.,5
```

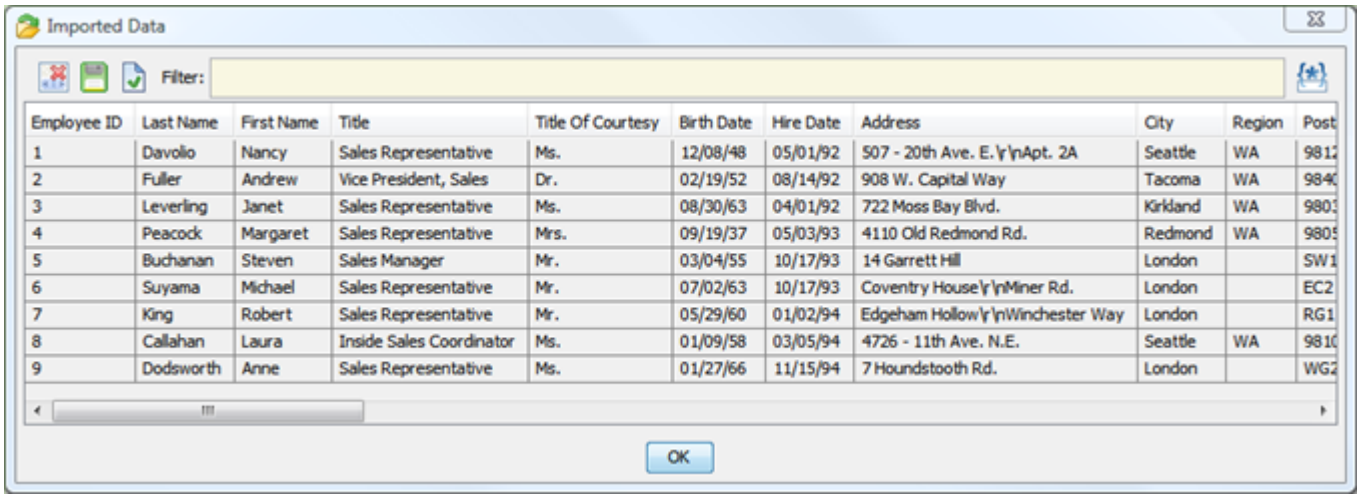
- Кликните правой кнопкой мыши по корню узла **Владельцы карт** (или любому узлу **Владельцы** карт в Вашей иерархии, если у вас есть подразделения, отделы и пр.).

- Щелкните **Импорт**.
- В диалоге **Выбрать файл для импорта**, щелкните **Выбрать** и используя навигатор по файловой системе укажите файл. В нашем случае это `employees.csv`.
- Щелкните **OK**.
- Вы увидите диалог **Опции для импорта CSV**. Этот диалог имеет несколько опций:

1. **Разделитель полей**: Это - символ-разделитель для полей. По умолчанию это точка с запятой (;), но в нашем случае это запятая (,). Так что если Вы следуете нашим инструкциям, то измените его.
2. **Использовать разделитель текста**: Разделитель текста используется в начале и конце длинных строк, например "like this". Обычно это символ двойных кавычек. Эта опция отключена по умолчанию, но нам она нужна. Включите её.
3. **Разделитель текста**: Это непосредственно символ-разделитель. Можно оставить тот, что указан по умолчанию. В нашем файле используются как раз двойные кавычки.
4. **Символ комментария**: Для комментирования внутри файла. Текст, который AtomMind будет игнорировать при импорте. В нашем файле нет комментариев, но эту опцию можно оставить как есть.
5. **Режим экранирования**: Что, если бы Вам захотелось вставить в текст символ, который используется в качестве разделителя (например, двойные кавычки)? Есть два традиционных подхода к обработке этой ситуации. Некоторые программы используют обратную косую черту перед таким символом, при этом текст выглядит так: \". Другие программы просто повторяют двойные кавычки дважды вот так: "". Эта опция позволяет Вам выбрать один из этих двух способов экранирования. Для нашего примера нам необходимо изменить значение этой опции. Вы берите **Использовать обратную косую черту**.
6. **Заголовок**: Иногда первая запись в файле используется для именования полей или их описания. Иногда это просто отдельная запись и иногда она даже содержит мусор, который лучше не принимать во внимание. Эта настройка позволяет Вам выбрать один из четырех режимов. Для нашего примера выберите **Описание полей**.

Смотрите приложение Опции импорта/экспорта CSV 2161.

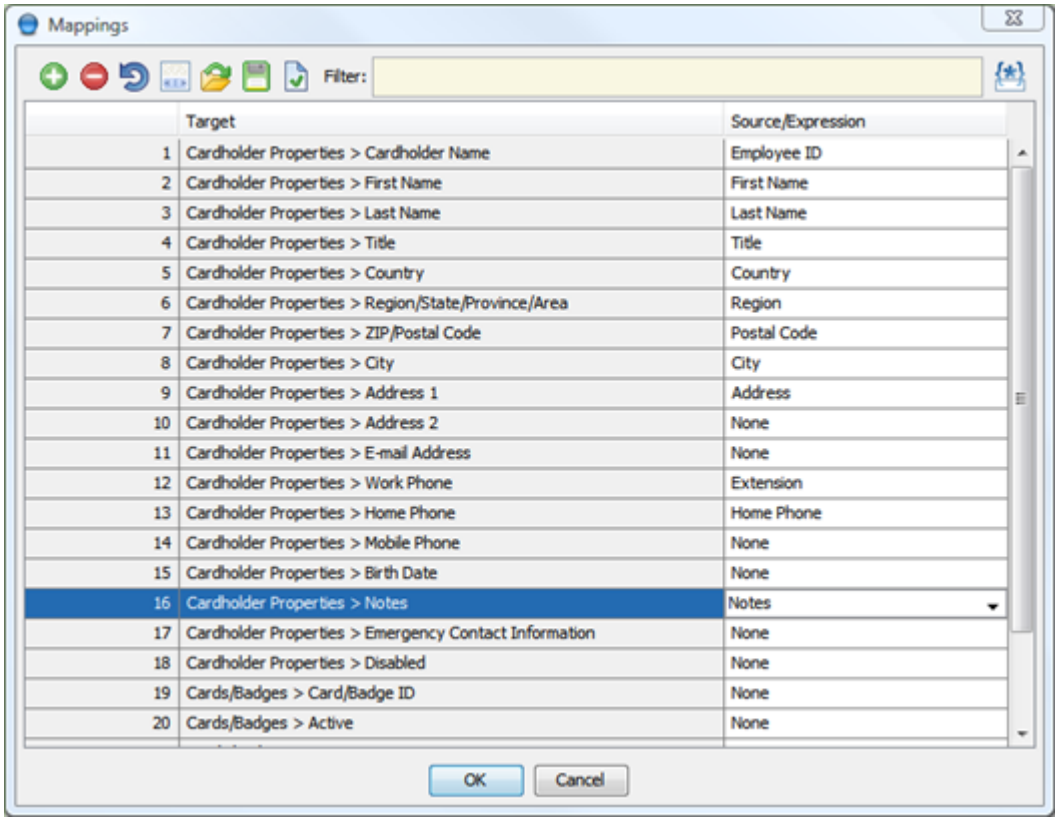
- Щелкните **OK.**
- Если Вы все правильно сделали, Вы увидите следующий диалог:



Employee ID	Last Name	First Name	Title	Title Of Courtesy	Birth Date	Hire Date	Address	City	Region	Post
1	Davolio	Nancy	Sales Representative	Ms.	12/08/48	05/01/92	507 - 20th Ave. E.\r\nApt. 2A	Seattle	WA	9812
2	Fuller	Andrew	Vice President, Sales	Dr.	02/19/52	08/14/92	908 W. Capital Way	Tacoma	WA	9840
3	Leverling	Janet	Sales Representative	Ms.	08/30/63	04/01/92	722 Moss Bay Blvd.	Kirkland	WA	9803
4	Peacock	Margaret	Sales Representative	Mrs.	09/19/37	05/03/93	4110 Old Redmond Rd.	Redmond	WA	9805
5	Buchanan	Steven	Sales Manager	Mr.	03/04/55	10/17/93	14 Garrett Hill	London		SW1
6	Suyama	Michael	Sales Representative	Mr.	07/02/63	10/17/93	Coventry House\r\nMiner Rd.	London		EC2
7	King	Robert	Sales Representative	Mr.	05/29/60	01/02/94	Edgeham Hollow\r\nWinchester Way	London		RG1
8	Callahan	Laura	Inside Sales Coordinator	Ms.	01/09/58	03/05/94	4726 - 11th Ave. N.E.	Seattle	WA	9810
9	Dodsworth	Anne	Sales Representative	Ms.	01/27/66	11/15/94	7 Houndstooth Rd.	London		WG2

Обратите внимание, насколько хорошо он разбил поля. Он также указывает имена полей в заголовках ячеек.

- Нашим следующим шагом будет поставить соответствия этим полям имена полей AtomMind. Щёлкните **OK**.



- Диалог **расставления соответствий** показывает каждое поле в записи о Владельце Карты в AtomMind. Раскройте выпадающее меню Источник/Выражение сразу за каждым из полей и выберите имя поля из Вашего CSV. Например, **Имя** так и останется **Именем** (Ваш CSV содержит определение этого поля), в то время как **Имя владельца карты** должно соответствовать **Идентификатору работника** (Employee ID). Если Вы не можете найти соответствующее поле, оставьте его как **None**.
- Выставьте соответствия, Которые Вас устроят и нажмите **OK.** AtomMind создаст новый контекст для каждого из владельцев карт. Импорт закончен.

19.4.4.2 Исключение посещений

Каждый сотрудник, его отдел, подразделение и организация могут иметь особые настройки расписания 2048. Хотя каждый конкретный сотрудник может иметь "общее" расписание, важно отслеживать индивидуальные события посещения, такие как::

- Выходные
- Дни болезни
- Отгулы
- Незапланированные рабочие дни
- И пр.

Такие индивидуальные события называются *исключение посещений*. Каждый владелец карт имеет собственную таблицу **Исключений посещений**, позволяющую операторам просматривать и менять информацию о таких особых событиях.

Свойства Исключительных посещений

Большинство полей **Исключений посещений** совпадают с такими же полями из Сдвига расписания 2049. Кроме дополнительных двух:

- Категория (больничный, отгул, выходной, рабочий день или другое)
- Комментарий

Обработка исключений посещений

Исключения посещений имеют приоритет перед сдвигами расписания. Это значит, что если у владельца карты имеется персональное исключение для конкретного дня, свойства этого исключения будут использоваться для обработки событий этого дня.

Категория и комментарий исключения будут показаны в отчетах посещений.

19.4.5 Управление электронными картами доступа

В AtomMind у каждого владельца электронного пропуска может быть больше одной карты или бейджа. Бейдж может быть как активным, так и неактивным. Если сотрудник теряет свой бейдж (или Вы переходите на другую карточную систему), можно просто добавить новый бейдж для этого человека, а прежний сделать неактивным.

У одного человека может быть несколько активных бейджей. Это означает, что есть множество способов отметить приход и уход с работы и это правильно отобразится в системе.

Создание карт

- Щелкните правой кнопкой мыши по владельцу карты, для которого Вы хотите добавить карту, и выберите **Настроить пользователя карты**. Можно также дважды щелкнуть мышью по элементу Пользователь Карты.
- Щелкните мышью по вкладке **Карты/Бейджи**.
- Щелкните мышью по кнопке **Добавить ряд (**  **)** и заполните следующие поля:

ID карты, бейджа: должны соответствовать тому, что Ваше устройство передает AtomMind Server. Значение может быть от 1 до 100 знаков длиной и не может содержать пробелы.

Статус*: Активный, Приостановлен, Утрачен, Украден или Поврежден*. Любой статус, отличный от *Активного* деактивирует карту.

Дата активации: это та дата, с которой карта считается активной. Для карты, которая считается активной, дата активации должна относится к прошедшему периоду.

Дата деактивации: это та дата, с момента которой карта перестает быть активной. Для карты, которая считается активной, дата деактивации относится в будущему периоду (или же "не установлено").

Удаление карт

Технически Вы можете удалить ряд, содержащий информацию о карте, щелкнув мышью по кнопке **Удалить выбранный ряд** (). Однако это не лучшее решение, поскольку это привело бы к тому, что все записи для этой карты исчезнут из последующих отчетов об учете времени. Поэтому будет лучше деактивировать карту (отменить флажок **Активна**).

Когда Вы просто деактивируете карту (а не удаляете ее), система будет знать, что ранее она принадлежала определенному владельцу карты и принимает это во внимание при последующих, связанных с ним отчетах.

ПОДРОБНЕЕ О ЛОГИКЕ АКТИВАЦИИ/ДЕАКТИВАЦИИ

Это может представлять некоторую сложность:

Если карта отмечена как **Активная**, система проверяет, соответствует ли ее дата активации заявленной. Если ее **дата Активации** соответствует заявленной, система посчитает карту активной.

Если карта не отмечена как **Активная**, система не посчитает ее активной и даже не будет проверять ее дату активации.

Если карта **Активная**, но ее **дата Активации** относится к будущему периоду, или же **дата Деактивации** к прошлому, система посчитает карту неактивной.

19.4.6 Управление сменами

AtomMind Time and Attendance предназначен для использования в организациях двух типов, для каждой из которых существует свой организационный порядок:

Офисный тип организации, недельная смена

Под "офисной организацией" понимается типичная организация, рабочий график которой составляет 5 дней в неделю по 9 часов, при этом основная часть сотрудников приходит на рабочее место и уходит одновременно. Вместе с тем здесь могут быть исключения (ночные сторожи, системные администраторы, работающие в ночную смену, и пр.), но у них свой собственный график работы.

Такая организация использует по крайней мере один *недельный график.* Этот тип графика позволяет просматривать всю рабочую неделю целиком и устанавливать еженедельный график работы (понедельник, вторник и т.д.).

Мы используем обычный пример, чтобы настроить Недельную смену. В наше описание мы включим подтемы для тех, кому свойственно лишь поверхностно изучать руководством - можно будет прочить их все или же лишь ту часть, которая заинтересовала.

СВОЙСТВА УЗЛА СМЕНЫ

- Двойной щелчок по узлу **Смены** открывает диалоговое окно **Свойств узла Смены** для добавления новой смены.
- Введите следующие элементы:

Имя: наименование смены. Обычные требования к наименованию контекста 42 - запрещается использовать пробелы. Введите "ОбычнаяНеделя"

Описание: описание, которое появляется в системном дереве и пр. Введите "Обычная Рабочая Неделя".

Тип: это то, где Вы выбираете лишь **Недельную смену** либо **Обычную смену**. В этом случае, например, выберите Недельная Смена.

Тип смены нельзя изменить после ее создания.

Переработка после конца рабочего дня: разрешено ли сотрудникам оставаться после конца рабочего дня, и прибавляются ли им сверхурочные?

Переработка перед началом рабочего дня: разрешено ли сотрудниками приходить раньше начала рабочего времени, и прибавляются ли им сверхурочные?

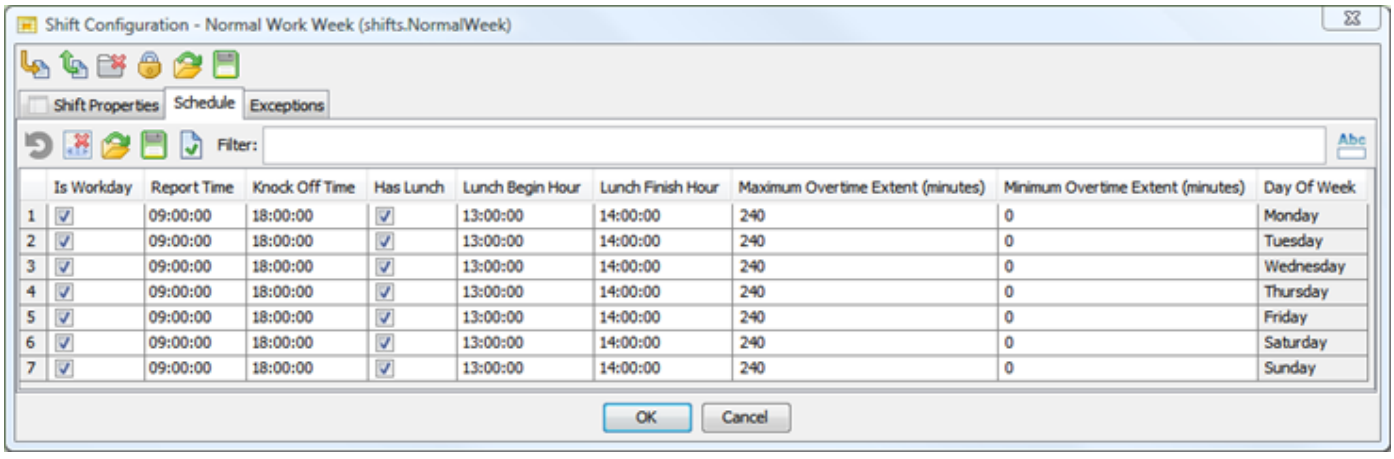
Переработка в выходные: разрешено ли сотрудникам приходить в официально нерабочие дни, и прибавляются ли им сверхурочные?

Переработка в обеденное время: разрешено ли сотрудникам сокращать время обеда или пропускать его, и прибавляются ли им сверхурочные?

- Настройте данные элементы согласно организационной политике Вашей компании и кликните **ОК.**

РАСПИСАНИЕ

- А теперь щелкните мышью дважды по созданной смене (**Обычная Рабочая Неделя**) для того, чтобы настроить для нее рабочий график.
- Кликните по вкладке **Расписание**:



	Is Workday	Report Time	Knock Off Time	Has Lunch	Lunch Begin Hour	Lunch Finish Hour	Maximum Overtime Extent (minutes)	Minimum Overtime Extent (minutes)	Day Of Week
1	☑	09:00:00	18:00:00	☑	13:00:00	14:00:00	240	0	Monday
2	☑	09:00:00	18:00:00	☑	13:00:00	14:00:00	240	0	Tuesday
3	☑	09:00:00	18:00:00	☑	13:00:00	14:00:00	240	0	Wednesday
4	☑	09:00:00	18:00:00	☑	13:00:00	14:00:00	240	0	Thursday
5	☑	09:00:00	18:00:00	☑	13:00:00	14:00:00	240	0	Friday
6	☑	09:00:00	18:00:00	☑	13:00:00	14:00:00	240	0	Saturday
7	☑	09:00:00	18:00:00	☑	13:00:00	14:00:00	240	0	Sunday

- Как Вы можете видеть, каждому дню присвоен ряд со следующими полями:

Рабочий день: должны ли сотрудники приходить в офис в этот день?

Время прихода: когда должно отмечаться время их прихода на работу?

Время ухода: когда наступает конец рабочего дня?

Обед: включен ли в график работы перерыв на обед? Если день короткий/неполный, перерыв на обед может быть не включен в рабочий график.

Начало обеда/Окончание обеда:

- **Отсутствует** - не принято определенное обеденное время, любые прерывания работы в течение дня будут обрабатываться, как перерывы.
- **Фиксированное** - в этом случае должны быть заданы **Время начала обеда** и **Время окончания обеда**. В системе существует специальный алгоритм поиска прерывания работы, чье время и продолжительность соответствуют времени/периоду обеда. Это прерывание в работе будет рассматриваться как обеденное, а все остальные прерывания в работе в течение дня будут обрабатываться как перерывы.
- **Гибкое** - в этом случае должна быть установлена **Продолжительность обеда**. Первые минуты прерванной работы регистрируются как обеденные, любое время прерывания работы, которое превышает **Продолжительность обеда**, считается перерывом.

См. Расчет перерывов 2016 для более подробного ознакомления с обработкой времени обеда/перерыва.

Начало времени обеда, Конец времени обеда: Промежутки обеденного времени, используется, когда поле **Обед** установлено на **Гибкое**.

Продолжительность обеда: Максимальная длина обеденного времени, используется, когда поле **Обед** установлено на **Гибкое**.

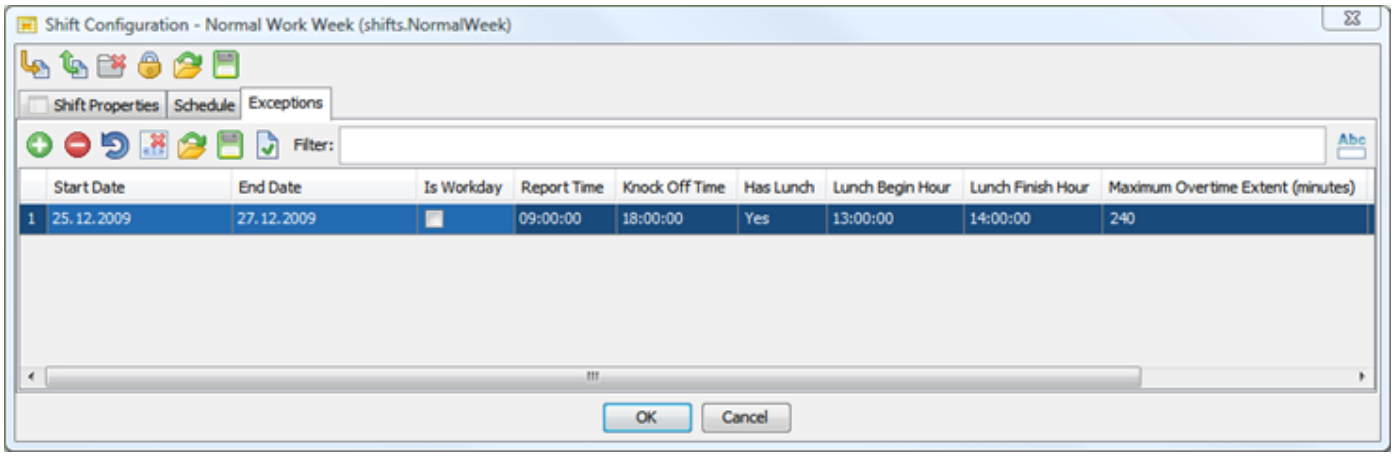
Максимальное время переработки (в минутах): какой объем внеурочного времени может прибавляться за один из этих дней?

Минимальное время переработки (в минутах): с какого момента начинается отсчет внеурочного времени после окончания рабочего дня? Например, если это 15 минут, а сотрудник уходит в 18:10, эти десять минут не будут учтены. Они не будут считаться внеурочными, несмотря на то, что это время после окончания рабочего дня, поэтому оно будет просто сброшено, а сотрудник не получит компенсацию за переработку. Это важный момент, на который следует обратить внимание потому, что неправильная настройка этого параметра может привести к нарушению соответствующего пункта трудового законодательства.

Объем внеурочного времени учитывается и ***до*** начала рабочего дня, если это разрешено (см. выше **Внеурочное время до начала работы**).

ИСКЛЮЧЕНИЯ

Вкладка **Исключения** позволяет настроить праздничные дни:



Начальная дата: когда начинаются праздничные дни?

Конечная дата: когда заканчиваются праздничные дни?

Все остальные поля аналогичны выше описанным выше.

Если **Рабочий** день не помечен праздничным *и* **Внеурочный нерабочий день** не выбран в закладке **Свойства Смены**, сотрудники, приходящие на работу в праздничные дни, не получат доплату за переработку - как если бы офис был закрыт, а они приходят в офис в эти дни. Пометьте это как исключение.

Производственный тип организации, периодическая смена

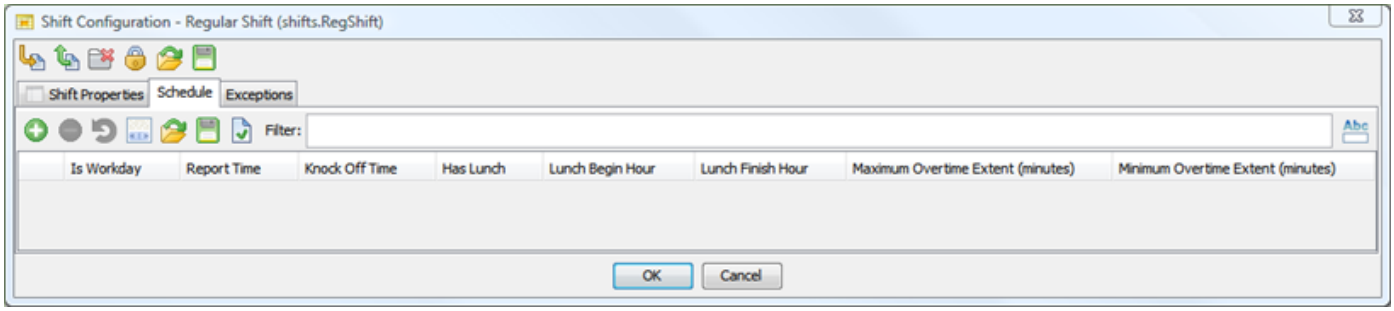
Обычные смены выстраиваются циклично, не так как типичная рабочая неделя. Например, у Вас может быть 3-дневный цикл: 2 дня рабочих, а третий выходной. Давайте настроим обычную смену:

СВОЙСТВА

- Кликните дважды по узлу **Смены** и следуйте инструкциям, приведенным для свойств смены 2049 выше. Следует обратить внимание на *одно отличие:* в элементе **Тип** нужно выбрать **Регулярная смена**.

РАСПИСАНИЕ

- Теперь кликните дважды по созданной смене, чтобы настроить для ее график.
- Кликните по вкладке **Расписание**:



Здесь разница между Недельной и Периодической Сменой становится более прозрачной. В то время как для Недельной смены есть заполненная предварительно таблица на семь дней, для Периодической Смены имеется пустая таблица.

Все свойства такие же как и для Недельной Смены (см. Управление сменами 2049 выше), но смена - это определенный цикл дней. У Вас может быть трехдневный, двухдневный цикл и т.п.

ИСКЛЮЧЕНИЯ

Элемент Исключения настраивается аналогично Недельной Смене. См. выше 2050.

НАЧАЛЬНАЯ ДАТА

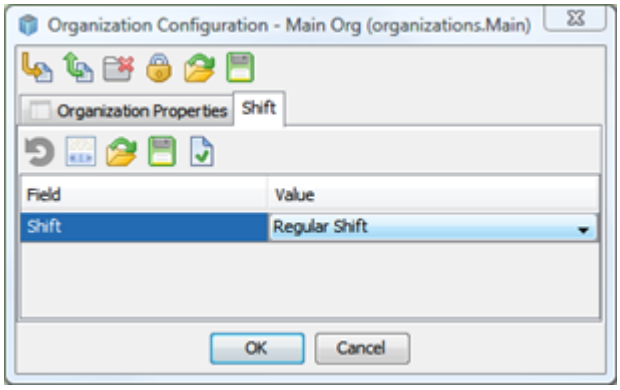
Когда Вы создаете Регулярную смену, AtomMind настраивает еще один параметр для него, отличный от Недельной смены: **Начальная дата.** Это необходимо AtomMind, чтобы позднее сообщать, в какой день обычной смены работал сотрудник из-за цикличного характера обычной смены. Например, если у Вас обычная смена, которая разделяется на трехдневный цикл, и сотрудник работает пять дней, день номер пять будет соответствовать второму дню обычной смены. Чтобы это распознать, AtomMind нужно знать дату начала периодической смены.

Вы можете видеть эту настройку и изменять ее (хотя это не рекомендуется), повторно обратившись к свойствам Периодической Смены после ее создания (но не во время самого процесса создания).

Установка смен

ДЛЯ ОРГАНИЗАЦИЙ

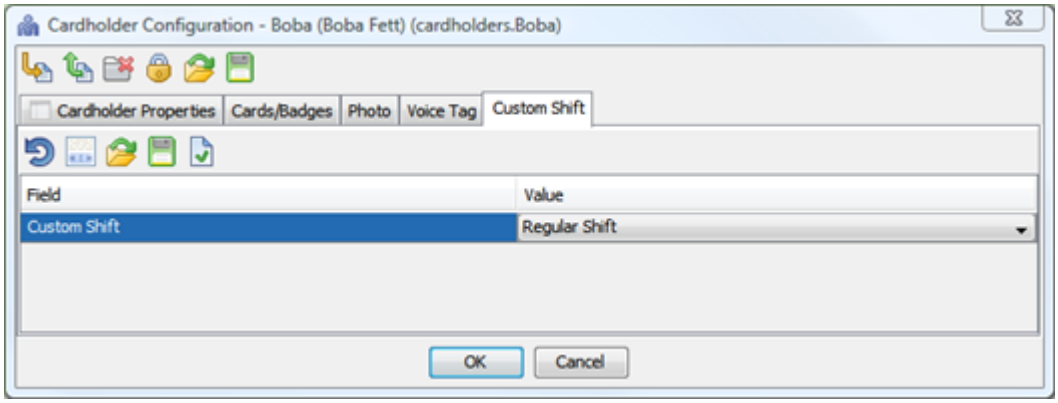
Дважды кликните по элементу Организация в дереве, кликните на вкладку **Смена** и выберите смену, которую вы хотите установить для данной организации и весь контекст, который она содержит.



ОБЫЧНЫЕ СМЕНЫ ДЛЯ ПОДРАЗДЕЛЕНИЙ, ОТДЕЛОВ И ВЛАДЕЛЬЦЕВ КАРТ

Даже если Вы назначаете смену для целой организации, Вы можете назначить различные смены для подразделений, отделов или владельцев карт, входящих в эту организацию.

Кликните дважды по элементу подразделение, отдел или сотрудник. Кликните по вкладке **Рабочая смена** и назначьте в поле **Рабочая Смена** подходящую в Вашем случае смену.



Смена по умолчанию

Существует недельная смена, которая создается автоматически, - это **Смена по умолчанию**. Ее можно редактировать, но удалять нельзя. Она представляет собой "классическую" 8-часовую рабочую неделю с понедельника по пятницу.

19.4.7 Управление событиями посещений

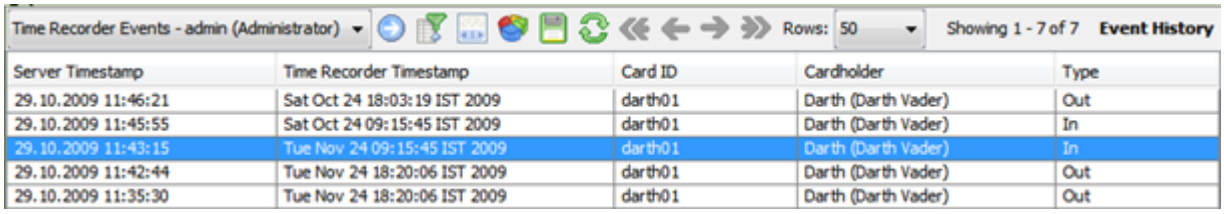
События посещения - это события 73, исходящие от терминалов учета рабочего времени. У каждого события есть три поля:

Отметка времени учетчика времени	Это то время, о котором сообщает записывающее устройство. Оно может отличаться от времени сервера (из-за разницы временных зон и пр.)
ID карты	ID карты должен быть привязан к определенному владельцу карты в системе
Тип	Может быть **вход** и **выход**.

AtomMind добавляет следующую информацию для каждого события:

Отметка времени сервера	Местное время на сервере, когда событие было получено.
Пользователь карты	Какой пользователь карты привязан к данной карте.

Всю эту информацию можно просмотреть через Журнал регистрации событий:

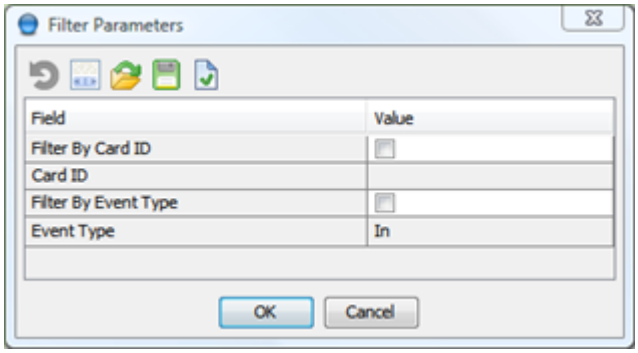


Server Timestamp	Time Recorder Timestamp	Card ID	Cardholder	Type
29.10.2009 11:46:21	Sat Oct 24 18:03:19 IST 2009	darth01	Darth (Darth Vader)	Out
29.10.2009 11:45:55	Sat Oct 24 09:15:45 IST 2009	darth01	Darth (Darth Vader)	In
29.10.2009 11:43:15	Tue Nov 24 09:15:45 IST 2009	darth01	Darth (Darth Vader)	In
29.10.2009 11:42:44	Tue Nov 24 18:20:06 IST 2009	darth01	Darth (Darth Vader)	Out
29.10.2009 11:35:30	Tue Nov 24 18:20:06 IST 2009	darth01	Darth (Darth Vader)	Out

Как просмотреть события терминала учета рабочего времени в Журнале регистрации

Если Вы не уверены в том, что знаете, что такое журнал регистрации событий, можете обратиться к главе Журнал регистрации событий 398. Объяснение, данное ниже, предполагает, что Вы уже имеете понятие о Журнале регистрации событий.

- Щелкните по окну списка **Фильтры** (расположен в верхнем левом углу на скриншоте, приведенном выше) и выберите **События терминалов учета рабочего времени** - *(ваше имя пользователя)*
- Должно появится диалоговое окно **Параметров фильтра**:



Фильтровать по ID карты: если Вы ищете события, относящиеся к одному определенному бейджу или карте. Обратите внимание, что ID *карта* и имя владельца карты не одно и то же.

ID карты: ID, который вы хотите задать в качестве критерия для фильтрации (если есть)

Фильтр по типу события: переключает фильтрацию событий **вход/выход**.

Тип события: выберите **вход** или **выход**, чтобы задать критерий для фильтрации.

- Кликните **OK**, чтобы провести фильтрацию согласно выбранным Вами параметрам. Если ни один из параметров не выбран, это приведет к тому, что будет выбран слишком широкий набор событий. Обратите внимание, что Вы получите только события от учетчиков времени, которые принадлежат Вам.

Удаление и подтверждение событий

Если кликнуть правой кнопкой мыши по событию в журнале регистрации событий, появится контекстное меню. Вы сможете:

Показать данные о событии: получить дополнительную информацию о событии, например, о ID карты и пр.

Подтверждение события: введите подтверждение (текст оператора), относящееся к данному событию.

Просмотр подтверждений: если есть.

Удалить событие: это полностью удалит событие, и, таким образом, сделает его несуществующим для всех.

Связанные действия: некоторые другие операции, которые Вы можете выполнить с контекстом, который сгенерировал событие, такие как перезагрузка. Вы можете перезагрузить терминал учета времени через контекстное меню.

19.4.8 Обработка данных посещений

В этом параграфе говорится о внутренней архитектуре системы. Здесь описывается алгоритм выполнения системных операций. Это может оказаться полезным при выявлении неисправностей, поскольку позволяет глубже понять механизм работы системы.

Основная задача этого параграфа - описание правил обработки данных о посещениях работы пользователей карт для построения отчетов.

Сюда относится:

- Обнаружение владельцев карт, у которых есть активные карты/бейджи для выбранного диапазона дат.

- Обнаружение активной смены для каждого владельца карты.

- Обнаружение записи смены, которая необходима для обработки данных о посещениях.

- Обнаружение событий посещения владельца карты и нахождение отсутствующих событий.

- Подсчет рабочего времени, перерывов/обеденных перерывов и сверхурочного времени.

КакAtomMind определяет, что карта активна

Во время обработки данных карты AtomMind требуется определить, является ли та или иная карта активной или нет.

Немного терминологии:

Дата начала означает дату начала Вашего запроса. Если Вы хотите получить отчет о посещаемости за период с 1/янв/2009 по 1/фев/2009, а тогда 1/янв/2009 будет **датой начала** (а 1/фев/2009 является, соответственно, **датой окончания**). А теперь поговорим о том, как система понимает, какие карты/бейджи следует включить в этот запрос:

1. Она получает данные карты/бейджа от контекста определенного владельца карты (которые содержат таблицы данных со всеми зарегистрированными картами/бейджами).
2. Теперь она собирает все те записи из таблицы карт/бейджей, у которых стоит флаг **Активно**.
3. Она получает значение **даты активации** для каждой записи и отбрасывает запись, если **дата окончания** раньше даты активации (это означает, что запрашивается период времени, в который карта еще не была активной).
4. Затем AtomMind считывает значение **даты деактивации** для бейджа. Оно может быть нулевым. Если нет, и дата дезактивации оказывается раньше даты начала, система не принимает эту запись (поскольку это означает, что карта/бейдж уже были деактивированы датой начала).
5. Любая запись, прошедшая эти проверки успешно, является *активной картой* для данного пользователя карты и включается в отчет.

Обнаружение активной смены

AtomMind находит активную смену для данного владельца карты, когда проверяет настройки Периодической смены 2051 для определенного владельца карты. Если они не определены, он переходит выше в Иерархию организации и проверяет регулярную смену для отдела или подразделения. Если ни одно из них не определено, AtomMind берет смену, определенную для всей организации. Последующая обработка данных включает обработку данных по временным ограничениям и политике учета внеурочного времени, заданной для данной смены.

Обнаружение соответствующих событий посещения и нормализация последовательности событий

На этом этапе процесса у нас есть полный список карт/бейджей, информация о которых необходима, а также диапазон дат (см. выше **Дата начала** и **Дата окончания**).

1. AtomMind осуществляет поиск по базе данных всех событий, находящихся между датой начала и датой окончания с заданным ID карты/бейджа.
2. AtomMind сортирует все события согласно их отметки времени, от самой ранней до самой поздней.
3. Для каждого владельца карты по данному рабочему дню AtomMind проверяет, является ли первым событием IN (событием входа). Если первое событие IN отсутствует, т.е. первое событие - OUT(событие выхода), система добавляет событие IN согласно следующей логике:

   - Если событие OUT относится к периоду до начала смены: AtomMind добавляет событие IN с отметкой времени как раз до начала события OUT.

   - Если событие OUT относится к периоду после начала смены: AtomMind добавляет событие IN в начале смены 2048 (т.е. Времени Отчета).
4. AtomMind проверяет последнее событие владельца карты за тот день для того, чтобы убедиться, что это событие OUT (событие выхода). Если последнее событие за тот рабочий день - IN (событие входа) (это означает, что событие OUT отсутствует), AtomMind добавляет событие OUT согласно следующей логике:

   - Если событие IN относится к периоду до окончания смены: AtomMind добавляет событие OUT в то же время, когда смена заканчивается (т.е. Время окончания работы).

   - Если событие IN относится к периоду после конца смены: AtomMind добавляет событие OUT с отметкой времени как раз после события IN.
5. AtomMind проверяет итоговое множество событий для того, чтобы удостовериться, что нет двух событий IN или OUT, следующих друг за другом.

Обратите внимание: правила в пунктах 3 и 4 действительны для любых двух последовательных событий IN или OUT. Т.е. если AtomMind обнаруживает два *примыкающих друг к другу* события IN, в этом случае применяется правило 4; если он обнаруживает два примыкающих друг к другу события OUT, применяется правило 3.

Обнаружение дневных правил для данной пары событий

Теперь у AtomMind есть нормализированная пара событий для обработки. Каждая пара событий IN/OUT отмечает период времени, который сотрудник провел на рабочем месте. Этот период может как входить в границы заданной смены, так и находиться вне этих границ.

AtomMind также знает, к какой смене привязан тот или иной сотрудник. Поэтому теперь ей нужно определить, какие правила существуют для определенного дня, к которому относится пара событий.

ДЛЯ НЕДЕЛЬНОЙ СМЕНЫ

Для недельной смены алгоритм очень простой.

1. Когда произошло событие?
2. Дата определена как исключение? Если да, принимает ее правила. Если нет, двигается дальше и определяет...
3. В какой день недели произошло событие?
4. Теперь мы знаем правила для переработки и ожидаемый график для этой даты (поскольку недельная смена определяет правило для каждого дня недели).

ДЛЯ ОБЫЧНОЙ СМЕНЫ

Обычные смены всегда цикличны, поэтому определить, на какой день недели цикла выпадает пара событий, оказывается несколько сложнее. Давайте рассмотрим пример этой ситуации.

1. Допустим, у нас трехдневная периодическая смена, дата начала 2051 которой 21 июля 2009.
2. Пара событий, с которыми мы работаем, приходится на 17 октября 2009.
3. Определена ли эта дата как исключение? Если да, принимаем ее правила. Если нет, двигаемся дальше.
4. AtomMind подсчитывает разницу между датой начала и датой события. Это составляет 88 дней.
5. Теперь AtomMind делит разницу дней (88) на количество дней в смене (3) и подсчитывает остаток. В этом случае он составляет **1**.
6. Теперь AtomMind берет остаток и прибавляет его к **1** потому, что если число ряда для первого дня смены 1, а не 0 (поэтому, если остаток равен 0, например, правильный день - это день 1 смены). В нашем случае, результат будет **2**. Таким образом, применяются правила для второго дня трехдневного цикла.

Подсчет рабочего времени и переработки

1. Если время входит в рамки графика для определенного дня смены, оно считается нормальным рабочим временем. Это самый простой вариант.
2. Когда время выходит за границы графика дня, AtomMind использует комплекс алгоритмов для того, чтобы решить, считать это время как переработку, обычное рабочее время или сбросить его. Существует несколько возможных случаев:

   - Сотрудник находился на рабочем месте до начала смены.
   - Сотрудник находился на рабочем месте во время обеденного перерыва.
   - Сотрудник продолжал работать после окончания смены.
   - Сотрудник находился на рабочем месте в официально нерабочий день.

3. У каждого из этих случаев есть соответствующая опция в свойствах смены 2048. Если эта опция включена, AtomMind проверяет длительность переработки.

4. Теперь AtomMind подсчитывает общую длительность сверхурочного времени и проверяет, превышает ли она Минимальное время переработки. Если она меньше, чем Минимальное время переработки, переработке будет присвоено нулевое значение.

5. Теперь система суммирует все время по рабочему графику с переработкой и сравнивает результат с назначенной длительностью смены. Любое время, превышающее назначенное время для смены теперь считается переработкой - *только* то время, которое превышает итоговое рабочее время относительно итоговой длительности смены. Следовательно, задержка на рабочем месте на 30 минут " покрывает" 30-минутное опоздание и не учитывается как переработка.

ПРИМЕР

Допустим, у сотрудника Джоя рабочий график с 9:00 до 17:00. Джои опаздывает на работу в понедельник, приходит в 9:05. Он продолжает работать до 17:25, чтобы наверстать пропущенное время. В этом случае, Джои не присваивается сверхурочное время потому, что итоговая длительность его рабочего дня все равно составляет 8 часов.

У вас может появиться **проблематичный сценарий**: допустим, Вы устанавливаете **Минимальное время переработки** в 30 минут (не очень хорошая идея). В этом случае, если Джои приходит на работу в 9:25 и остается там до 17:25, система **не будет считать эти 25 минут**, потому что это меньше Минимального времени переработки**.** Это выглядело было так, что Джои просто пропустил 25 минут рабочего времени и не восполнил его (ему не оплатили те 25 минут, которые он провел после рабочего дня). Поэтому не следует устанавливать **Минимальное время переработки** больше 5 минут (или Минимальное время переработки на очень короткий промежуток времени).

Подсчет перерывов

Чтобы подсчитать время перерыва, AtomMind собирает все последовательности событий OUT-IN, укладывающиеся в рамки заданной смены. Он суммирует эти периоды, чтобы получить общее число всех перерывов.

Некоторые отчеты специально отражают время перерыва на обед (потому что время, когда сотрудник делает перерыв на обед устанавливается в свойствах Смены[2048]). Чтобы определить, какая пара событий OUT-IN в рамках смены обозначает перерыв на **обед**, AtomMind рассматривает значение настройки **Обед** рабочей смены[2048] :

ОТСУТСТВУЕТ

- Все пары событий вход-выход рассматриваются как перерывы.

ФИКСИРОВАННОЕ

1. Если есть событие OUT между **Часом начала обеда** и **Часом конца обеда**, AtomMind принимает это на начало перерыва на обед (и следующее за ним событие IN, как окончание перерыва на обед).

2. Если такого события нет, AtomMind примет последнее событие OUT, которое предшествует часу обеда.

Если сотрудник решает пропустить обед и пообедать позднее (после того, как час обеда уже закончился) AtomMind посчитает это период перерывом, но не отметит в отчетах как "обеденный перерыв".

ГИБКОЕ

Взяв за занчение **Продолжительности обеда** N минут, первые N минут прерываний в работе в течение дня будут рассматриваться как обеденное время, остальные - условное время перерыва.

Обработка времени обеда и перерывов

Суммируется **время всех перерывов на обед** с итоговым временем перерывов. Это означает, итоговое время перерывов за определенный день будет всегда включать время, потраченное на обеденный перерыв.

Если Вы не устанавливаете в свойствах смены **Переработку в обеденный перерыв**, сотрудники не *смогут работать во время обеденного перерыва.* Они могли бы остаться на рабочих местах и продолжить работу, но система не посчитает затраченное время рабочим или переработкой. Поэтому, если Джои решает поработать с 13:00-14:00 (в установленное обеденное время), а затем решит сделать перерыв с 14:30-15:30, система вычтет из его рабочего дня *два часа*: обеденное время (потому, что система не разрешала работать во время обеденного времени) и тот перерыв, который он решил сделать самостоятельно. Поэтому так важно настроить систему правильно!

19.4.9 Отчеты о рабочем времени

Составление отчетов

Вы можете составить отчет для одного пользователя карты или группы пользователей.

Для **одного пользователя карт**: щелкните правой кнопкой мыши по элементу пользователь карты и выберите тип отчета, который необходимо создать.

Чтобы составить **отчет для группы**: щелкните правой кнопкой мыши по группе (организации, подразделению, отделу) и выберите, какой тип отчета необходимо создать. Отчет, в итоге, будет включать информацию обо всех владельцах карт в этом контексте, как можно увидеть ниже:

Содержание колонок для отчетов не нуждаются в объяснении, особенно после прочтения главы Обработка данных посещений[2016].

Отчет о посещаемости

Этот отчет, содержащий ежедневные данные, демонстрирует, когда сотрудники приходят на работу, покидают рабочее место, и как долго они работают (без учета перерывов).

Attendance

Period: 12.10.09 - 16.10.09

Employee	Card ID	In Time	Out Time	Work Time
12.10.09				
Boba Fett	001	08:51	17:01	7:13
Darth Vader	002	08:51	17:22	7:31
Han Solo	003	09:12	17:02	6:52
Jabba the Hutt	004	09:04	17:23	7:11
Lando Calrissian	005	09:12	17:03	6:45
Leia Organa	006	00:00	00:00	0:00
Luke Skywalker	007	00:00	00:00	0:00
Mace Windu	008	00:00	00:00	0:00
Obi-Wan Kenobi		00:00	00:00	0:00
13.10.09				
Boba Fett	001	08:55	17:11	7:09
Darth Vader	002	09:03	17:17	7:10
Han Solo	003	09:06	17:12	7:12
Jabba the Hutt	004	08:56	17:03	7:08
Lando Calrissian	005	09:00	16:59	6:50
Leia Organa	006	00:00	00:00	0:00
Luke Skywalker	007	00:00	00:00	0:00

Отчет об активности за день

Этот отчет более подробен и включает смену, к которой привязан каждый из сотрудников, время его прихода, длительность его обеденного перерыва, время ухода с работы и пр. Поле **Комментарии** создается автоматически.

Daily Activity

Period: 12.10.09 - 16.10.09

Employee	Card ID	Shift	In Time	Lunch Time	Break Time	Out Time	First In To Last Out	Work Time	Over Time	Comments
12.10.09										
Boba Fett	001	Default Shift	08:51	0:57	0:00	17:01	8:10	7:13	0:13	
Darth Vader	002	Default Shift	08:51	1:00	0:00	17:22	8:31	7:31	0:31	Late Out
Han Solo	003	Default Shift	09:12	0:58	0:00	17:02	7:50	6:52	0:00	
Jabba the Hutt	004	Default Shift	09:04	1:08	0:00	17:23	8:19	7:11	0:11	Late Out
Lando Calrissian	005	Default Shift	09:12	1:06	0:00	17:03	7:51	6:45	0:00	
Leia Organa	006	Default Shift	00:00	0:00	0:00	00:00	0:00	0:00	0:00	Absent
Luke Skywalker	007	Default Shift	00:00	0:00	0:00	00:00	0:00	0:00	0:00	Absent
Mace Windu	008	Default Shift	00:00	0:00	0:00	00:00	0:00	0:00	0:00	Absent
Obi-Wan Kenobi		Default Shift	00:00	0:00	0:00	00:00	0:00	0:00	0:00	Absent
13.10.09										
Boba Fett	001	Default Shift	08:55	1:07	0:00	17:11	8:16	7:09	0:09	
Darth Vader	002	Default Shift	09:03	1:04	0:00	17:17	8:14	7:10	0:10	Late Out
Han Solo	003	Default Shift	09:06	0:54	0:00	17:12	8:06	7:12	0:12	
Jabba the Hutt	004	Default Shift	08:56	0:59	0:00	17:03	8:07	7:08	0:08	
Lando Calrissian	005	Default Shift	09:00	1:09	0:00	16:59	7:59	6:50	0:00	
Leia Organa	006	Default Shift	00:00	0:00	0:00	00:00	0:00	0:00	0:00	Absent
Luke Skywalker	007	Default Shift	00:00	0:00	0:00	00:00	0:00	0:00	0:00	Absent

Отчет о ежедневном использовании рабочего времени

Этот отчет позволяет сравнить установленное рабочее время с действительным, а также выявить возможные различия (например, Люк Скайуокер даже не показывается в офисе (см. график ниже), или Дарт Вейдер работает сверхурочно).

Daily Utilization

Date: 16.10.09

Employee	Card ID	Shift	Schedule	Scheduled Time	In Time	Out Time	Work Time	Scheduled vs Actual	Comments
Boba Fett	001	Default Shift	9.00 - 17.00	7:00	09:07	16:57	6:48	(12)	
Darth Vader	002	Default Shift	9.00 - 17.00	7:00	08:51	17:30	7:40	40	Late Out
Han Solo	003	Default Shift	9.00 - 17.00	7:00	08:56	17:07	7:14	14	
Jabba the Hutt	004	Default Shift	9.00 - 17.00	7:00	08:57	17:00	7:05	5	
Lando Calrissian	005	Default Shift	9.00 - 17.00	7:00	08:55	17:25	7:29	29	Late Out
Leia Organa	006	Default Shift	9.00 - 17.00	7:00	00:00	00:00	0:00	(420)	Absent
Luke Skywalker	007	Default Shift	9.00 - 17.00	7:00	00:00	00:00	0:00	(420)	Absent
Mace Windu	008	Default Shift	9.00 - 17.00	7:00	00:00	00:00	0:00	(420)	Absent

Отчет о событиях терминала учета рабочего времени

Этот отчет используется, главным образом, для поиска и устранения неполадок. Поскольку AtomMind использует внутреннюю логику 2016 для компенсации любых отсутствующих событий (т.е. случаев, когда сотрудники не отмечали время прихода или ухода с работы), этот отчет позволяет выделить те события, которые были получены из терминала, без сопутствующей логики AtomMind.

Time Recorder Events

Period: 12.10.09 - 16.10.09

Time	Name	Event Type
12.10.09 08:51	Boba Fett	in
12.10.09 12:04	Boba Fett	out
12.10.09 13:01	Boba Fett	in
12.10.09 17:01	Boba Fett	out
13.10.09 08:55	Boba Fett	in
13.10.09 11:58	Boba Fett	out
13.10.09 13:05	Boba Fett	in
13.10.09 17:11	Boba Fett	out
14.10.09 08:58	Boba Fett	in
14.10.09 12:01	Boba Fett	out
14.10.09 12:57	Boba Fett	in
14.10.09 17:00	Boba Fett	out
15.10.09 09:11	Boba Fett	in
15.10.09 11:57	Boba Fett	out
15.10.09 13:02	Boba Fett	in

Time	Name	Event Type
13.10.09 17:17	Darth Vader	out
14.10.09 09:02	Darth Vader	in
14.10.09 12:03	Darth Vader	out
14.10.09 13:02	Darth Vader	in
14.10.09 17:27	Darth Vader	out
15.10.09 09:03	Darth Vader	in
15.10.09 11:58	Darth Vader	out
15.10.09 13:03	Darth Vader	in
15.10.09 17:20	Darth Vader	out
16.10.09 08:51	Darth Vader	in
16.10.09 11:56	Darth Vader	out
16.10.09 12:55	Darth Vader	in
16.10.09 17:30	Darth Vader	out
12.10.09 09:12	Han Solo	in
12.10.09 12:02	Han Solo	out

Карта контроля времени

Этот отчет, в котором данные сгруппированы для каждого отдельного владельца карты (в отличие от отчета о Ежедневной Активности). Он содержит ту же самую информацию о каждом владельце карты, но в более удобном для просмотра.

Timecard

Period: 12.10.09 - 16.10.09

Date/Time	In Time	Lunch Time	Break Time	Out Time	First In To Last Out	Work Time	Over Time	Comments
Boba Fett (Card ID: 001, Shift: Default Shift)								
12.10.09	08:51	0:57	0:00	17:01	8:10	7:13	0:13	
13.10.09	08:55	1:07	0:00	17:11	8:16	7:09	0:09	
14.10.09	08:58	0:56	0:00	17:00	8:02	7:06	0:06	
15.10.09	09:11	1:05	0:00	17:17	8:06	7:01	0:01	Late Out
16.10.09	09:07	1:02	0:00	16:57	7:50	6:48	0:00	
Darth Vader (Card ID: 002, Shift: Default Shift)								
12.10.09	08:51	1:00	0:00	17:22	8:31	7:31	0:31	Late Out
13.10.09	09:03	1:04	0:00	17:17	8:14	7:10	0:10	Late Out
14.10.09	09:02	0:59	0:00	17:27	8:25	7:26	0:26	Late Out
15.10.09	09:03	1:05	0:00	17:20	8:17	7:12	0:12	Late Out
16.10.09	08:51	0:59	0:00	17:30	8:39	7:40	0:40	Late Out
Han Solo (Card ID: 003, Shift: Default Shift)								
12.10.09	09:12	0:58	0:00	17:02	7:50	6:52	0:00	
13.10.09	09:06	0:54	0:00	17:12	8:06	7:12	0:12	
14.10.09	09:07	0:52	0:00	17:02	7:55	7:03	0:03	
15.10.09	08:54	0:59	0:00	17:30	8:36	7:37	0:37	Late Out
16.10.09	08:56	0:57	0:00	17:07	8:11	7:14	0:14	

Кто внутри/Кто снаружи

Эти отчеты позволяют видеть, кто из сотрудников находится на рабочем месте в настоящий момент, а кто из сотрудников отсутствует.

19.4.9.1 Комментарии к отчету

Некоторые отчеты (такие как Ежедневная Активность) содержат поле Комментарии. Это поле содержит заранее подготовленные сообщения, которые предназначены помогать администраторам помечать определенные обстоятельства. Ниже приводится список всех сообщений и сопутствующих им обстоятельств.

Событие вход пропущено	Одно событие **вход** или более добавлены внутренней логикой AtomMind потому, что они не были получены из Терминала Учета Рабочего Времени.
Событие выход пропущено	Одно событие **выход** или более добавлены внутренней логикой AtomMind потому, что они не были получены их терминала Учета Рабочего Времени.
Опоздание	Первое событие **вход** владельца карты позднее установленного по графику начала рабочего дня на 15 минут или более.
Ранний приход	Первое событие **вход** владельца карты раньше установленного по графику начала рабочего дня на 15 минут или более.
Поздний уход	Последнее событие **выход** владельца карты позднее установленного по графику конца рабочего дня на 15 минут или более.
Ранний уход	Последнее событие **выход** владельца карты ранее установленного по графику конца рабочего дня на 15 минут или более.
Короткий обед	Обеденный перерыв короче установленного по графику на 15 минут или более.
Длинный обед	Обеденный перерыв длиннее установленного по графику на 15 минут или более.
Отсутствует	События владельца карты на этот день отсутствуют.

19.4.10 Справочник по контекстам

В этой главе говорится о контекстах, переменных, функциях, событиях и действиях, относящихся к учету рабочего времени.

19.4.10.1 Посещаемость

Этот системный контекст, который предоставляет доступ к различным данным, относящимся ко всей операции Учета рабочего времени. Он не указывается в видимой части контекстного дерева.

Дополнительная информация

Информация о контексте

Тип контекста[43]: посещаемость

Имя контекста[42]: attendance

Описание контекста[43]: посещаемость

Путь Контекста[42]: посещаемость

Маска контекста[44]: посещаемость

Права доступа контекста [?[44]]

Уровень	Описание
Отсутствует	Доступ не разрешен.
Наблюдатель	Все операции.
Оператор	Те же, что у Наблюдателя.
Менеджер	Те же, что у Наблюдателя.
Инженер	Те же, что у Наблюдателя.
Администратор	Те же, что у Наблюдателя.

Общие переменные (Свойства) [?[61]]

У этого контекста не общих переменных (свойств).

Общие функции [?[70]]

У этого контекста нет общих функций.

Общие события [?[73]]

Общие события: (Информация)[77]

ПОСЕЩАЕМОСТЬ

Событие **посещаемость** генерируется, когда устройство Учета Времени, связанное с AtomMind Server, производит событие *In* или *Out.*. Оно предоставляет общий формат данных о посещаемости, в то время как другие типы учета времени сообщают события посещаемости в другом формате.

Это аппаратно-независимое событие. Каждый терминал учета рабочего времени или драйвер должен конвертировать данные о посещаемости, передаваемые с оборудования в события **посещаемости**. Эти события обрабатывает AtomMind Time and Attendance, чтобы создать различные отчеты. В то же время, эти необработанные события могут быть доступны для сторонних программ составления отчетов, расчетных ведомостей, а также для другого программного обеспечения через AtomMind Server API[134] или Веб-сервисы[141].

Возможно также собирать все события **посещаемости** в отдельную таблицу базы данных, используя при этом устройство записи событий[199]. Сторонние системы могут иметь доступ к этой таблице напрямую для того, чтобы получить данные о посещаемости для обработки.

Имя события: attendance

Права доступа: Доступно на уровне 486 прав доступа к контексту

Период действия: 10 лет

Записи: 1

Формат 49 записи:

Имя поля	Тип поля	Примечания
timestamp	Дата	Отметка времени регистрации события, проставленная терминалом (следует отличать от серверной отметки времени, т.е. времени, когда событие было получено сервером)
cardId	Строка	Идентификатор карты.
cardholder	Строка	Тот, кому принадлежит карта (путь к контексту владельца карты)
device	Строка	Контекстный путь контекста учетчика времени, откуда отправляется это событие.
type	Строка	Тип события: '**in**' или '**out**'

19.4.10.2 Сотрудники

Этот контекст 41 является контейнером, содержащим контексты сотрудников 2063 на определенном уровне организационной иерархии или же все контексты пользователей картами вне данной организационной иерархии.

Уникальные действия [?1450]

У контекста Сотрудники доступно несколько действий по демонстрации отчетов 933 для просмотра отчетов по всем дочерним сотрудникам:

- **Посещаемость** 2056
- **Активность за день** 2057
- **Использование рабочего времени** 2057
- **События терминалов учета рабочего времени** 2058
- **Карта контроля времени** 2058
- **Кто внутри** 2059
- **Кто снаружи** 2059

СОЗДАТЬ (Действие по умолчанию 88)

Это действие используется для создания нового сотрудника. Оно позволяет пользователю выбрать основные свойства для нового сотрудникв и настроить его сразу же после создания.

Тип действия: Создать 105

Права доступа: Доступно на уровне 486 права доступа *Администратор*

Общие действия [?1450]

Создать на основе шаблона 105, Копировать в дочерние контексты 111, Импорт 108, Экспорт 108, Редактировать права доступа 106, Просмотр событий 109, Поиск/фильтрация 110, различные Групповые действия 101, относящиеся к дочерним контекстам.

Иконки и состояния контекста

У этого контекста нет состояний[44]. Его всегда представляет иконка  .

Дополнительная информация

Информация о контексте

Тип контекста[43]: пользователи карт

Имя контекста[42]: cardholders

Описание контекста[43]: Сотрудники

Путь контекста[42]: cardholders, organizations.ORGANIZATION_NAME.cardholders, organizations.ORGANIZATION_NAME.divisions.DIVISION_NAME.cardholders, organizations.ORGANIZATION_NAME.divisions.DIVISION_NAME.departments.DEPARTMENT_NAME.cardholders

Маска контекста[44]: **cardholders, organizations.*.cardholders** для всех держателей во всех организациях, **organizations.*.divisions.*.cardholders** для всех держателей во всех подразделениях всех организаций, **organizations.*.divisions.*.departments.*.cardholders** для пользователей карт всех отделов во всех подразделениях всех организаций.

Права доступа к контексту [?[44]]

Уровень	Описание
Отсутствует	Доступ не разрешен.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт владельца карты.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (Свойства) [?[61]]

СМЕНА

Эта переменная определяется только для "корневого" узла сотрудники, который содержит сотрудников, не принадлежащих к организации.

Имя переменной: shift

Записи: 1

Права доступа: Доступно для чтения на уровне[486] права доступа *Наблюдателя,* доступно для записи на уровне прав доступа *Менеджера*

Формат[49] записи:

Имя поля	Тип поля	Примечания
shift	Строка	Смена[205] для всех дочерних пользователей карт.

Общие функции [?[70]]

Общие функции: makeCopy (сделать копию)[71], delete (удалить)[72]

СОЗДАТЬ

Создает нового сотрудника.

Имя функции:	create
Права доступа:	Доступно на уровне 486 с правом доступа *Администратор.*
Входные записи:	1
Формат 49 ввода:	Такой же формат как и у переменной childInfo 2064 в контексте Cardholder 2063.
Выходные записи:	0
Формат 49 вывода:	нет

Общие события [? 73]

Общие события: info (информация) 77

19.4.10.3 Сотрудник

Этот контекст 41 предоставляет Вам доступ к одному из сотрудников 2043 и позволяет управлять этим узлом.

Уникальные действия [? 1450]

У контекста Сотрудник есть несколько действий показать отчет 933, которые необходимы для просмотра отчетов о данном сотруднике:

- **Посещаемость** 2056
- **Активность за день** 2057
- **Использование рабочего времени** 2057
- **События терминалов учета рабочего времени** 2058
- **Карта контроля времени** 2058
- **Кто внутри** 2059
- **Кто снаружи**

НАСТРОИТЬ (Значение по умолчанию 88)

Это действие используется для редактирования свойств сотрудника.

Смена поля **Имени** во время этой операции вызовет переименование текущего контекста. Это может привести к неправильному функционированию других компонентов системы, использующих имя/путь контекста как главный идентификатор.

Тип действия:	Настроить 105

Общие действия [? 1450]

Удалить 108, Создать копию 109, Реплицировать 110, Редактировать права доступа 106, Просмотр событий 109, Показать статус 111

Иконки и состояния контекста

У этого контекста нет состояний 44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста 43: сотрудник

Имя контекста 42: предоставляется пользователем

Описание контекста 43: предоставляется пользователем

Путь контекста 42: **cardholders.**ИМЯ_СОТРУДНИКА для сотрудников, которым принадлежит контекст сотрудники 2061 в корневом контексте, организациях. **ORGANIZATION_NAME.cardholders.**ИМЯ_СОТРУДНИКА для сотрудников, которым принадлежит контекст сотрудников в определенном контексте организации в организациях.**ORGANIZATION_NAME.divisions.DIVISION_NAME.cardholders.**ИМЯ_СОТРУДНИКА для сотрудников, которым принадлежит контекст в определенном контексте подразделений в организациях. **DIVISION_NAME.departments.DEPARTMENT_NAME.cardholders.**ИМЯ_СОТРУДНИКА для сотрудников, которым принадлежит контекст сотрудников в определенном контексте отдела.

Маска контекста 44: **cardholders.*** для всех сотрудников, которым принадлежит контекст сотрудников 2061 в корневом контексте в организациях**.cardholders.*** для всех сотрудников, которым принадлежит контекст сотрудников во всех контекстах организации, в организациях.***.divisions.*.cardholders.*** для всех сотрудников, которым принадлежит контекста сотрудников во всех контекстах подразделения, в организациях. ***.cardholders.***для всех сотрудников, которым принадлежит контекст во всех контекстах отдела.

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Доступ не разрешен.
Наблюдатель	Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Конфигурирование и удаление владельца карты.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (Свойства) [? 61]

Общие переменные: groupMembership (групповое членство) 67, activeAlerts (активные тревоги)

СВОЙСТВА

См. описание переменной и ее поля здесь 2043.

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне 486 прав доступа *Наблюдателя*, доступно для внесение записи на уровне прав доступа *Менеджера*

Формат 49 записи:

Имя поля	Тип поля	Примечания
name	строка	**Имя пользователя картой**. Имя контекста этого пользователя, необходимое для обращения к данному пользователю картой из других частей системы. Оно должно удовлетворять соглашениям по именованию 42 контекста. Длина: 1-50 знаков.
firstname	строка	имя
lastname	строка	фамилия
gender	целое число	пол
position	строка	должность

country	целое число	страна
region	логическое значение	регион/штат/провинция/область
zip	строка	ZIP/почтовый код
city	строка	город
address1	строка	адрес № 1
address2	строка	адрес № 2
email	строка	E-mail
workPhone	строка	рабочий телефон
homePhone	строка	домашний телефон
mobilePhone	строка	мобильный телефон
notes	строка	примечания
birthDate	дата	дата рождения
disabled	логическое значение	недееспособный
emergency	строка	Контактные данные экстренной помощи

ИДЕНТИФИКАЦИОННЫЕ КАРТОЧКИ

Описание переменных и их полей см. здесь 2048.

Имя переменной: cards

Записи: 0...до бесконечности

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдатель,* доступно для записи на уровне с правами доступа *Менеджера.*

Формат 49 записи:

Имя поля	Тип поля	Примечания
id	строка	Card/Badge ID
status	целое число	статус
activationDate	дата	Дата активации
deactivationDate	дата	Дата деактивации

ФОТОГРАФИЯ

Имя Переменной: photo

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдатель,* доступно для записи на уровне с правами доступа *Менеджера.*

Формат 49 записи:

Имя поля	Тип поля	Примечания
photo	блок данных	использует редактор изображений

ГОЛОСОВАЯ МЕТКА

Имя Переменной: voice

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа *Наблюдатель,* доступно для записи на уровне с правами доступа *Менеджера.*

Формат 49 записи:

Имя поля	Тип поля	Примечания
voice	блок данных	Использует редактор звука

ОСОБАЯ РАБОЧАЯ СМЕНА

См. описание переменной и ее полей здесь 2051.

Имя переменной: customShift

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа *Наблюдатель,* доступно для записи на уровне с правами доступа *Менеджера.*

Формат 49 записи:

Имя поля	Тип поля	Примечания
customShift	строка	Смена 2048 пользователя карты

ИСКЛЮЧЕНИЯ ПОСЕЩАЕМОСТИ

См. описание переменной и ее поля здесь 2047.

Имя Переменной: attendance

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа *Наблюдатель,* доступно для записи на уровне с правами доступа *Менеджера.*

Формат 49 записи:

Имя поля	Тип поля	Примечания

startDate	дата	Используется редактор дат.
endDate	дата	Используется редактор дат.
isWorkday	логическое значение	
reportingTime	дата	Используется редактор времени.
knockOffTime	дата	Используется редактор времени.
hasLunch	логическое значение	
lunchBeginHour	дата	Используется редактор времени.
lunchEndHour	дата	Используется редактор времени.
maximumOT	длинное	
minimumOT	длинное	
category	целое число	
comment	строка	

Общие функции [? 70]

У этого контекста отсутствуют общие функции.

Общие события [? 73]

Общие события: info (информация) 77

19.4.10.4 Отделы

Этот контекст 41 является контейнером, который содержит все контексты отдела 2068 в соответствующем подразделении.

Уникальные действия [? 1450]

СОЗДАТЬ (Значение по умолчанию 88)

Это значение используется для создания нового отдела 2042. Оно позволяет пользователю задать основные свойства нового отделения и настроить его сразу же после создания.

Тип действия: Создать 105

Права доступа: Доступно лишь на уровне 486 с правами доступа *Менеджера.*

Общие действия [? 1450]

Создать на основе шаблона 105, Копировать в дочерние контексты 111, Импорт 108, Экспорт 108, Редактировать права доступа 106, Просмотр событий 109, Поиск/фильтрация 110, различные Групповые действия 101, относящиеся к дочерним контекстам.

Состояния и иконки контекста

У этого контекста нет состояний 44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста 43: отделы

Имя контекста 42: departments

Описание контекста 43: отделы

Путь к контексту 42: organizations.ORGANIZATION_NAME.divisions.DIVISION_NAME.departments

Маска контекста 44: organizations.*.divisions.*.departments

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Доступ не разрешен.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт отделов.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (Свойства) [? 61]

У этого контекста нет общих переменных (свойств).

Общие функции [? 70]

Общие функции: makeCopy (сделать копию) 71, delete (удалить) 72

СОЗДАТЬ

Создает новый отдел.

Имя функции: create

Права доступа: Доступно лишь на уровне 486 прав доступа *Менеджера*

Записи ввода: 1

Формат 49 ввода: Такой же, как и формат переменной childInfo 2070 в контексте Отдел 2068.

Записи вывода: 0

Формат 49 вывода: нет

Общие события [? 73]

Общие события: info (информация) 77

19.4.10.5 Отдел

Этот контекст 41 предоставляет Вам доступ к одному отделу 2042 и возможность им управлять.

Уникальные действия [? 1450]

Контекст Отдел имеет несколько действий показать отчет 933 для просмотра отчетов на узле сотрудники 2063, которые относятся к этому отделу:

- **Посещаемость** 2056
- **Активность за день** 2057
- **Использование рабочего времени** 2057
- **События терминалов учета рабочего времени** 2058
- **Карта контроля времени** 2058
- **Кто внутри** 2059
- **Кто снаружи**

НАСТРОИТЬ (действие по умолчанию 88)

Это действие используется для редактирования свойств узла Отдел.

Смена поля **Имени** во время этой операции вызовет переименование текущего контекста. Это может привести к неправильному функционированию других компонентов системы, использующих имя/путь контекста как главный идентификатор.

Тип действия: Настроить 105

Общие действия [? 1450]

Удалить 106, Создать копию 109, Реплицировать 110, Редактировать права доступа 106, Просмотр событий 109, Показать статус 111

Состояния и иконки контекста

У этого контекста нет состояний 44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста 43: отдел

Имя контекста 42: предоставляется пользователем

Описание контекста 43: предоставляется пользователем

Путь к контексту 42: organizations.ORGANIZATION_NAME.divisions.DIVISION_NAME.departments.DEPARTMENT_NAME

Маска контекста 44: organizations.*.divisions.*.departments.*

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Доступ не разрешен.
Наблюдатель	Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Создание, экспорт и импорт отделов.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (Свойства) [? 61]

Общие переменные: groupMembership (членство в группе)67, activeAlerts (активные тревоги)

СВОЙСТВА

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне486 с правами доступа *Наблюдатель*, доступно для записи на уровне с правами доступа *Менеджера.*

Формат49 записи:

Имя поля	Тип поля	Примечания
name	Строка	**Наименование отдела**. Имя этого контекста отдела, необходимое, чтобы обращаться к этому отделу из других частей системы. Оно должно удовлетворять соглашению о присвоении имен42 контекста**.** Длина: 1-50 знаков.
description	строка	**Описание отдела**. Длина: 1-100 знаков.

ОСОБАЯ РАБОЧАЯ СМЕНА

Имя переменной: customShift

Записи: 1

Права доступа: Доступно для чтения на уровне486 с правами доступа *Наблюдатель*, доступно для записи на уровне с правами доступа *Менеджер.*

Формат49 записи:

Имя поля	Тип поля	Примечания
customShift	строка	Смена205 отдела

Общие функции [?70]

У этого контекста нет общих функций.

Общие события [?73]

Общие события: info (информация)77

19.4.10.6 Подразделения

Этот контекст41 является контейнером, который содержит все контексты подразделений2072 соответствующей организации.

Уникальные действия [?1450]

СОЗДАТЬ (Действие по умолчанию88)

Это действие используется для создания нового подразделения2042. Оно позволяет пользователю задавать основные свойства для нового подразделения и настраивать его сразу после создания.

Тип действия: Создать 105

Права доступа: Доступно на уровне 486 прав доступа для *Менеджера.*

Общие действия [? 1450]

Создать на основе шаблона 105, Копировать в дочерние контексты 111, Импорт 108, Экспорт 108, Редактировать права доступа 106, Просмотр событий 109, Поиск/фильтрация 110, различные Групповые действия 101, относящиеся к дочерним контекстам.

Состояния и иконки контекста

У этого контекста нет состояний 44. Его всегда представляет иконка  .

Дополнительная информация

Информация о контексте

Тип контекста 43: подразделения

Имя контекста 42: divisions

Описание контекста 43: подразделения

Путь контекста 42: organizations.ORGANIZATION_NAME.divisions

Маска контекста 44: organizations.*.divisions

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Доступ не разрешен.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт подразделений.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (Свойства) [? 61]

У этого контекста нет общих переменных (свойств).

Общие функции [? 70]

Общие функции: makeCopy (сделать копию) 71, delete (удалить) 72

СОЗДАТЬ

Создает новое подразделение.

Имя функции: create

Права доступа: Доступно на уровне 486 с правами доступа *Менеджера*

Записи ввода: 1

Формат 49 ввода: Такой же как и формат переменной childInfo 2073 в контексте подразделения 2073.

Запись вывода: 0

Формат 49 вывода: отсутствует

Общие события [? 73]

Общие события: info (информация) 77

19.4.10.7 Подразделение

Этот контекст 41 предоставляет Вам доступ к одному подразделению 2042 и позволяет им управлять.

Уникальные действия [? 1450]

У контекста подразделения есть несколько действий показать отчет 933 для просмотра отчетов на узле сотрудники 2063, которые принадлежат этому подразделению:

- **Посещаемость** 2056
- **Активность за день** 2057
- **Использование рабочего времени** 2057
- **События терминалов учета рабочего времени** 2058
- **Карта контроля времени** 2058
- **Кто внутри** 2059
- **Кто снаружи**

НАСТРОИТЬ (Действие по умолчанию 88)

Это действие используется для редактирования свойств подразделения.

Смена поля **Имени** во время этой операции вызовет переименование текущего контекста. Это может привести к неправильному функционированию других компонентов системы, использующих имя/путь контекста как главный идентификатор.

Тип действия: Настроить 105

Общие действия [? 1450]

Удалить 106, Создать копию 109, Реплицировать 110, Редактировать права доступа 106, Просмотр событий 109, Показать статус 111

Статусы и иконки контекста

У этого контекста нет статусов 44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста 43: подразделение

Имя контекста 42: предоставляется пользователем

Описание контекста 43: предоставляется пользователем

Путь к контексту 42: organizations.ORGANIZATION_NAME.divisions.DIVISION_NAME

Маска контекста 44: organizations.*.divisions.*

Права доступа к контексту [? 44]

Уровень	Описание

Отсутствует	Доступ не разрешен.
Наблюдатель	Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Создание и удаление подразделений.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [?61]

Общие переменные: groupMembership (членство группы)67, activeAlerts (активные тревоги)

СВОЙСТВА

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне486 с правами доступа *Наблюдатель*, доступно для записи на уровне с правами доступа *Менеджера*.

Формат49 записи:

Имя поля	тип поля	Примечания
name	строка	Имя подразделения. Имя этого контекста подразделения, необходимое для обращения к этому подразделению из других частей системы. Оно должно соответствовать соглашению о присвоении имен42. Длина: 1-50 знаков.
description	строка	Описание подразделения. Длина: 1-100 знаков.

ОСОБАЯ РАБОЧАЯ СМЕНА

Имя переменной: customShift

Записи: 1

Права доступа: Доступно для чтения на уровне486 с правами доступа *Наблюдатель*, доступно для записи на уровне с правами доступа *Менеджера*.

Формат49 записи:

Имя поля	Тип поля	Примечания
customShift	строка	Смена2051 подразделения

Общие функции [?70]

У этого контекста нет общих функций.

Общие события [?73]

Общие события: info (информация)77

19.4.10.8 Организации

Этот контекст[41] является контейнером, который содержит все контексты организации[2075] в системе.

Уникальные действия [?[1450]]

У контекста организаций есть несколько действий показать отчет[933], которые демонстрируют отчет, содержащий только отчеты по сотрудникам[2063], которые принадлежат любому из дочерних узлов свой организации:

- **Посещаемость**[2056]
- **Отчет о ежедневной активности**[2057]
- **Отчет о ежедневном использовании**[2057]
- **События учетчика времени**[2058]
- **Тайм-карта**[2058]
- **Кто пришел**[2059]
- **Кто отсутствует**[2059]

СОЗДАТЬ (Действие по умолчанию[88])

Это действие используется для создания новой организации. Оно позволяет пользователю задать основные свойства для новой организации инастроить ее сразу после создания.

Тип действия: Создать[105]

Права доступа: Доступно на уровне[486] с правами доступа *Менеджера.*

Общие действия [?[1450]]

Создать на основе шаблона[105], Копировать в дочерние контексты[111], Импорт[108], Экспорт[108], Редактировать права доступа[106], Просмотр событий[109], Поиск/фильтрация[110], различные Групповые действия[101], относящиеся к дочерним контекстам.

Состояния и иконки контекста

У этого контекста нет состояний[44]. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста[43]: организации

Имя контекста[42]: organizations

Описание контекста[43]: организации

Путь к контексту[42]: организации

Маска контекста[44]: организации

Права доступа к контексту [?[44]]

Уровень	Описание
Отсутствует	Доступ не разрешен.
Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт организаций.

Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (Свойства) [?61]

У этого контекста нет общих переменных (свойств).

Общие функции [?70]

Общие функции: makeCopy (сделать копию)71, delete (удалить)72

СОЗДАТЬ

Создает новую организацию.

Имя функции:	create
Права доступа:	Доступно на уровне486 с правами доступа для *Менеджера.*
Записи ввода:	1
Формат49 ввода:	Тот же самый формат, что и у переменной childInfo2076 в контексте Организация2075.
Записи вывода:	0
Формат49 вывода:	нет

Общие события [?73]

Общие события: info (информация)77

19.4.10.9 Организация

Этот контекст41 предоставляет Вам доступ к одной организации2042 и позволяет ею управлять.

Уникальные действия [?1450]

У контекста Организация есть несколько действий показать отчет933 для просмотра отчетов на узле сотрудники2063, которые принадлежат этой организации:

- **Посещаемость**2056
- **Активность за день**2057
- **Использование рабочего времени**2057
- **События терминалов учета рабочего времени**2058
- **Карта контроля времени**2058
- **Кто внутри**2059
- **Кто снаружи**

НАСТРОИТЬ (Значение по умолчанию88)

Это действие используется для редактирования свойств Организации.

Смена поля **Имени** во время этой операции вызовет переименование текущего контекста. Это может привести к неправильному функционированию других компонентов системы, использующих имя/путь контекста как главный идентификатор.

Тип действия:	Настроить105

Общие действия [?1450]

Удалить 106, Создать копию 109, Реплицировать 110, Редактировать права доступа 106, Просмотр событий 109, Показать статус 111

Состояния и иконки контекста

У этого контекста нет состояний 44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста 43: организация

Имя контекста 42: предоставляется пользователем

Описание контекста 43: предоставляется пользователем

Путь к контексту 42: organizations.ORGANIZATION_NAME

Маска контекста 44: organizations.*

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Доступ не разрешен.
Наблюдатель	Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Конфигурация и удаление организаций.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [? 61]

Общие переменные: groupMembership (членство группы) 67, activeAlerts (активные тревоги)

СВОЙСТВА ОРГАНИЗАЦИИ

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа *Наблюдатель,* доступно для записи на уровне с правами доступа *Менеджера.*

Формат 49 записи:

Имя поля	Тип поля	Примечания
name	строка	**Имя организации**. Имя контекста организации, необходимое для обращения к данной организации из других частей системы. Оно должно соответствовать соглашениям о присвоении имени 42 контексту. Длина 1-50 символов.
description	строка	**Описание организации**. Длина: 1-100 символов.

СМЕНА

Имя переменной: смена

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа для *Наблюдатель,* доступно для записи на уровне с правами доступа *Менеджера.*

Формат 49 записи:

Имя поля	Тип поля	Примечания
shift	строка	Смена 2051 организации

Общие функции [? 70]

У этого контекста нет общих функций.

Общие события [? 73]

Общие события: info (информация) 77

19.4.10.10 Рабочие смены

Этот контекст 41 является контейнером, который содержит контексты 2078 всех смен в системе.

Уникальные действия [? 1450]

СОЗДАТЬ (Действие по умолчанию 88)

Это действие используется для создания новой смены. Оно позволяет пользователю задавать основные свойства для новой смены и настраивать ее сразу после создания.

Тип действия: создать 105

Права доступа: Доступно на уровне 486 с правами доступа *Менеджера*.

Общие действия [? 1450]

Создать на основе шаблона 105, Копировать в дочерние контексты 111, Импорт 108, Экспорт 108, Редактировать права доступа 106, Просмотр событий 109, Поиск/фильтрация 110, различные Групповые действия 101, относящиеся к дочерним контекстам.

Состояния и иконки контекста

У этого контекста нет состояний 44. Он всегда представлен иконкой  .

Дополнительная информация

Информация контекста

Тип контекста 43: смены

Имя контекста 42: shifts

Описание контекста 43: смены

Путь к контексту 42: shifts

Маска контекста 44: shifts

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Доступ не разрешен.

Наблюдатель	Мониторинг основных событий.
Оператор	Те же, что у Наблюдателя.
Менеджер	Создание, экспорт и импорт организаций.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Открытые переменные (свойства) [?61]

У этого контекста нет общих переменных (свойств).

Открытые функции [?70]

Открытые функции: makeCopy (сделать копию)71, delete (удалить)72

СОЗДАТЬ

Создает новую смену.

Имя функции: create

Права доступа: Доступно на уровне486 с правами доступа *Менеджер.*

Записи ввода: 1

формат49 ввода: Такой же как и формат для переменной childInfo2079 контекста смена2078.

Записи вывода: 0

Формат49 вывода: нет

Общие события [?73]

Общие события: info (информация)77

19.4.10.11 Рабочая смена

Этот контекст41 предоставляет Вам доступ к одному узлу смена2048 и позволяет управлять им.

Уникальные действия [?1450]

НАСТРОИТЬ (Действие по умолчанию88)

Это действие используется для редактирования свойств смены.

Смена поля **Имени** во время этой операции вызовет переименование текущего контекста. Это может привести к неправильному функционированию других компонентов системы, использующих имя/путь контекста как главный идентификатор.

Тип действия: настроить105

Общие действия [?1450]

Удалить106, Создать копию109, Реплицировать110, Редактировать права доступа106, Просмотр событий109, Показать статус111

Состояния и иконки контекста

У этого контекста нет состояний44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста 43: смена

Имя контекста 42: предоставляется пользователем

Описание контекста 43: предоставляется пользователем

Путь к контексту 42: shifts.SHIFT_NAME

Маска контекста 44: shifts.*

Права доступа к контексту [? 44]

Уровень	Описание
Отсутствует	Доступ не разрешен.
Наблюдатель	Мониторинг основных событий. Просмотр статуса.
Оператор	Просмотр конфигурации.
Менеджер	Конфигурация и удаление организаций.
Инженер	Те же, что у Менеджера.
Администратор	Те же, что у Менеджера.

Общие переменные (свойства) [? 61]

Общие переменные: groupMembership (членство группы) 67, activeAlerts (активные тревоги)

СВОЙСТВА

См. описание переменной и ее полей здесь 2049.

Имя переменной: childInfo

Записи: 1

Права доступа: Доступно для чтения на уровне 486 с правами доступа *Наблюдатель*, доступно для записи на уровне с правами доступа *Менеджера*.

Формат 49 записи:

Имя поля	Тип поля	Примечания
name	строка	**Имя смены**. Имя контекста данной смены, необходимое для обращения к ней из разных частей системы. Оно должно соответствовать соглашениям по наименованию 42 контекста. Длина: 1-50 символов.
description	строка	**Описание смены**. Длина: 1-100 символов.
type	целое число	**Тип**. 0(недельная) или 1(обычная)
startDate	дата	**Дата начала**. Только для обычного типа смены.
autoOut	логическое значение	Автоматический выход.

otAfterEnd	логическое значение	Сверхурочное время после окончания рабочего дня.
otBeforeBegin	логическое значение	Сверхурочное время до начала рабочего дня.
otDayOff	логическое значение	Сверхурочное время во время нерабочего дня.
otDuringLunch	логическое значение	Сверхурочное время во время обеденного перерыва.

РАСПИСАНИЕ

См. описание переменной и ее полей здесь2049. У этой переменной есть несколько отличий для смен недельного или обычного типа.

Имя переменной: schedule

Записи: **7** для недельной смены, **0... не ограничено** для обычной смены.

Права доступа: Доступно для чтения на уровне486 с правами доступа *Наблюдатель*, доступно для записи на уровне с правами доступа *Менеджера*.

Формат49 записи:

Имя поля	Тип поля	Примечания
dayOfWeek	строка	**День недели**. У обычной смены это поле отсутствует.
isWorkday	логическое значение	
reportingTime	дата	используется только редактор времени.
knockOffTime	дата	используется только редактор времени.
hasLunch	логическое значение	
lunchBeginHour	дата	используется только редактор времени.
lunchEndHour	дата	используется только редактор времени.
maximumOT	целое число	
minimumOT	целое число	

ИСКЛЮЧЕНИЯ

См. описание переменной и ее полей здесь2050.

Имя переменной: exceptions

Записи: 0...не ограничено

Права доступа: Доступно для чтения на уровне486 с правами доступа *Наблюдатель*, доступно для записи на уровне с правами доступа *Менеджера*

Формат 49 записи:

Имя поля	Тип поля	Примечания
startDate	дата	Используется только редактор дат.
endDate	дата	Используется только редактор дат.
isWorkday	логическое значение	
reportingTime	дата	Используется только редактор времени.
knockOffTime	дата	Используется только редактор времени.
hasLunch	логическое значение	
lunchBeginHour	дата	Используется только редактор времени.
lunchEndHour	дата	Используется только редактор времени.
maximumOT	целое число	
minimumOT	целое число	

Общие функции [? 70]

У этого контекста нет общих функций.

Общие события [? 73]

Общие события: info (информация) 77

19.5 Управление серверами устройств

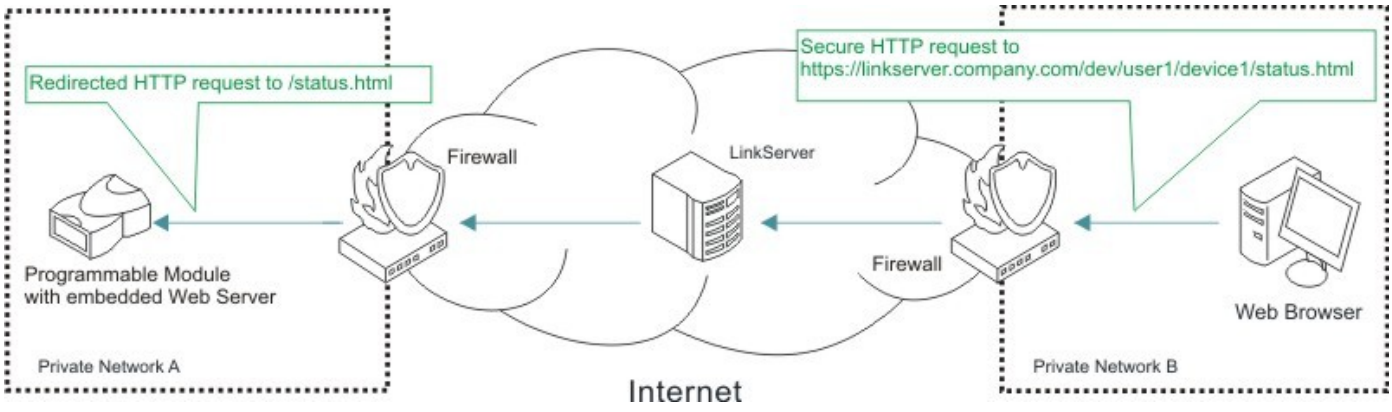
В этом разделе описывается управление Серверами устройств производства ТВЭЛ.

19.5.1 Сервис HTTP-прокси

Предположим, у Вас сотни Серверов Устройств со встроенными веб-серверами, каждый из которых предоставляет доступ к нескольким веб-страничкам. Очень просто получить доступ к каждому из них, если они все имеют реальные статические IP-адреса. Однако, в реальной жизни большинство Серверов Устройств устанавливаются в приватных сетях под защитой файерволов, а реальные статические IP-адреса стоят денег. В этой ситуации нет способа получить доступ к Серверам Устройств напрямую. Сервис HTTP Proxy решает эту проблему предоставляя единый способ доступа ко всем встраиваемым веб-серверам через AtomMind Server. Это работает так:

- Сервер Устройств со встроенным веб-сервером подключается к AtomMind Server'у. Его профиль 2087 должен быть настроен на использование режима HTTP Proxy.
- Если необходимо увидеть веб-страничку конкретного Сервера Устройств. Он запускает веб-браузер и и открывает адрес вида http://server.bigcompany.com/dev/admin/dev1/data.html
- HTTP-запрос отправляется браузером на AtomMind Server и обрабатывается им.
- AtomMind Server перенаправляет HTTP-запрос к подключенному в AtomMind Server Серверу Устройств.

- Сервер Устройств обрабатывает запрос и возвращает страницу *data.html* в AtomMind Server.
- AtomMind Server перенаправляет эту страницу в браузер клиента.



Более подробно эта технология описана в главе HTTP Proxy 2095 раздела Драйверы Устройств 518.

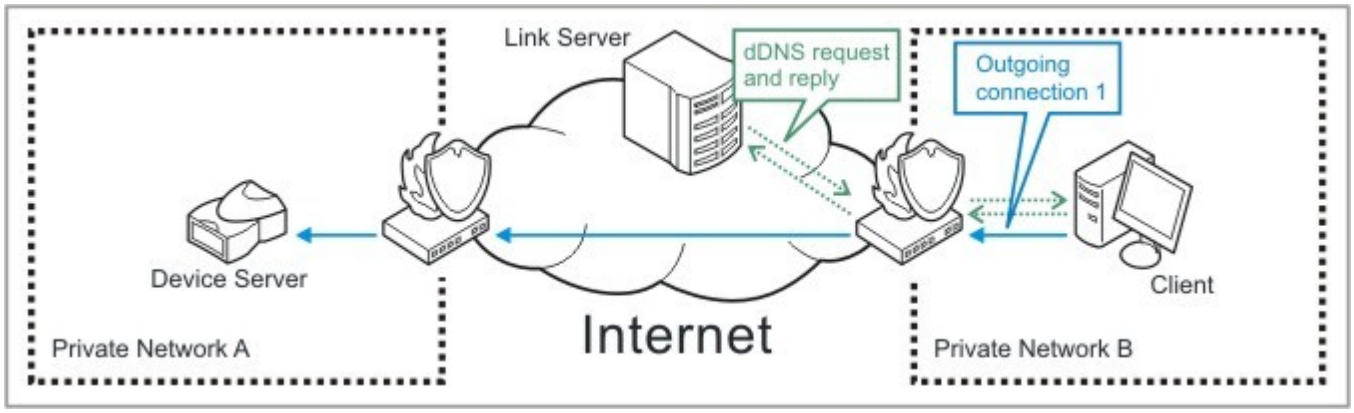
19.5.2 Сервис динамического DNS

Обратная сторона использования AtomMind Server'а в том, что он заведомо медленнее прямого соединения, поскольку данные должны проходить через AtomMind Server.

Это не критично для систем, с малым объемом трафика на каждом из узлов. Хотя иногда Вам может захотеться создать прямое подключение к устройству, но IP-адрес этого устройства может меняться со временем (как это происходит с большинством ADSL-подключений).

Вам необходимо решить эту проблему - т.е. нужен способ подключаться к статическому адресу и знать, что конкретный адрес принадлежит конкретному устройству,

Вот зачем появился *сервис динамического DNS (dDNS)*. С этим сервисом каждый ваш Сервер Устройств получает DNS-имя вида `dev1.abccorp.dev.srv1.com` (в этом примере доменное имя Вашего сервера `srv1.com`). Вы всегда можете подключаться к Вашему устройству по его хост-имени. URL устройства остается прежним, в то время как IP-адрес Сервера Устройств меняется.



Как только подключение произведено напрямую к Вашему Серверу Устройств, он становится Вам доступен (если его защищает файервол, то файервол должен быть верно настроен). Настройка файервола сложнее, чем настройка *Сервиса Связи* 2084, но Вы выиграете в скорости передачи данных.

В режиме динамического DNS Сервер Устройств подключается к AtomMind Server'у только при загрузке, после получения IP-адреса по DHCP. После регистрации в DNS он закрывает соединение с AtomMind Server'ом и работает **в точности** как "обычный" Сервер Устройств без всякой привязке к AtomMind Server'у или другим частям AtomMind. В отличии от режима Link Service 2084, не поддерживается соединение между Сервером Устройств и AtomMind Server'ом и данные не передаются через сервер.

Реализация

На стороне Сервера Устройств dDNS включается настройкой **регистрация в dDNS (DD)**. Когда она установлена в 1 (включено), Сервер Устройств регистрируется в *dDNS* сразу после включения. Во время регистрации AtomMind Server создает две записи в DNS: внешний IP-адрес Сервера Устройств и второй - "внутренний".

Пример двух записей в DNS для некоторого Сервера Устройств:

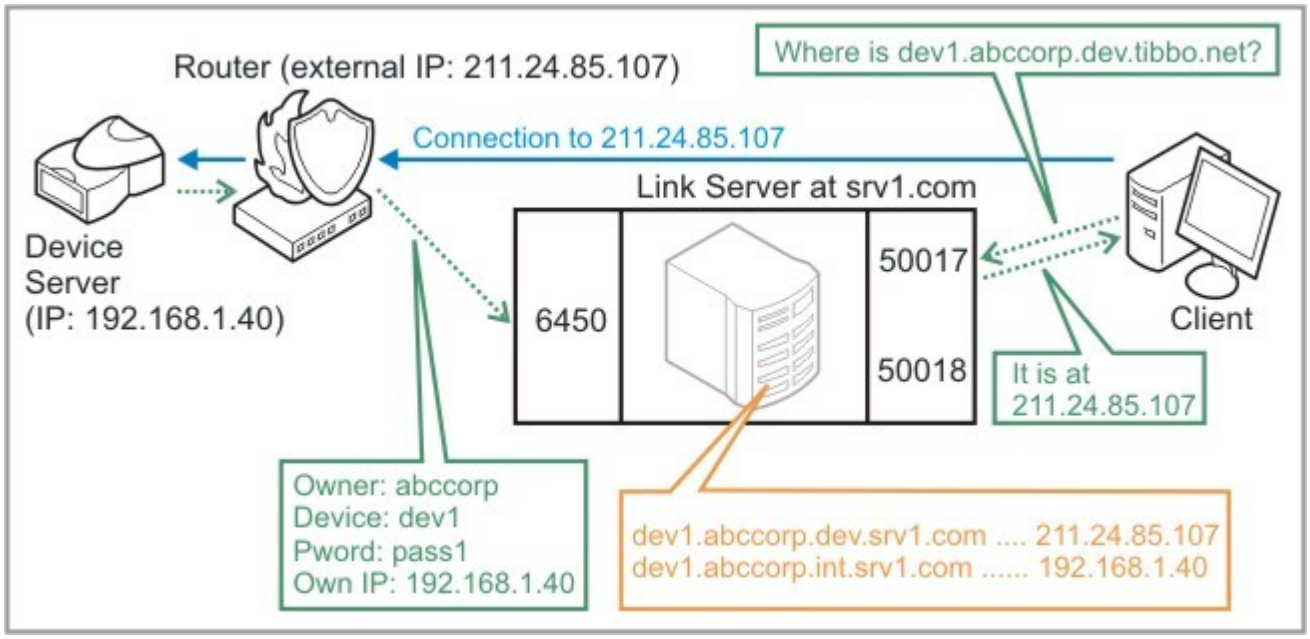
Для внешнего IP: `dev1.abccorp.dev.srv1.com`

Для внутреннего IP: `dev1.abccorp.int.srv1.com`

`Dev1` - это *имя устройства,* известное из настройки **Имя устройства (DN)** Сервера Устройств.

`Abccorp` - *имя владельца*, известное из настройки **Имя владельца (ON)** Сервера Устройств.

Получившееся в результате хост-имя не отличается от тех имен или URL-адресов, которые Вы уже когда-либо использовали. Введите это имя в любой программе, которая способна подключиться к Вашему Серверу Устройств и это имя будет автоматически преобразовано в текущий IP-адрес этого Сервера Устройств! Этот функция основана на стандартном протоколе DNS и не требует никаких особых драйверов или специального программного обеспечения. Сервера Устройств регистрируются на внешнем сервере DNS таком как BIND в *nix или Windows DNS Server.



Только зарегистрированные в AtomMind Server'e Сервера Устройств способны подключаться к сервису dDNS. Каждый Сервер Устройств идентифицируется на AtomMind Server'e своим **Именем Устройства (DN)**, **Именем Владельца (ON**) и **Паролем (PW)**.

Совместимость с DNS-серверами

Сервис Динамического DNS совместим со всеми DNS-серверами, поддерживающими динамические обновления, включая BINS, Windows 2003 Server и прочие.

Разница между внешними и внутренними адресами

Разница между этими двумя типами адресов довольно проста. Обычно Сервер Устройств подключается в сеть через маршрутизатор (или файервол), который зачастую "маскирует" IP-адрес устройства. Таким образом Сервер Устройств может находиться в сегменте с адресами вида 192.168.2.100 ("внутренний" IP), но для всего остального мира его адрес будет другим. Это "внешний" IP устройства. Итак, резюмируем:

- "Внутренний" IP-адрес - это настоящий IP-адрес Сервера Устройств, как 192.168.1.40. Он используется только в "своем" сегменте сети.
- "Внешний" IP-адрес - это IP-адрес маршрутизатора, настроенного пересылать данные между внешней и внутренней сетями. (Еще раз, маршрутизатор должен быть настроен для этого, это не стандартная автоматическая процедура, используемая везде и всегда и не требующая конфигурации. Эта настройка производится дополнительно к настройке dDNS.

Чтобы получить доступ к Серверу Устройств из внешнего сегмента сети Вам необходим внешний IP-адрес. Если Вы соединяетесь из того же самого сегмента сети, Вам нужно знать "внутренний" IP-адрес. Вот почему AtomMind Server создает обе записи.

19.5.3 Сервис связи

В нормальных условиях WAN-сети связь между узлами (хостами) может быть затруднена несколькими обстоятельствами:

Недостатком статических IP-адресов

IP-адреса во многих сетях (особенно в Интернет) дефицитны. Это заставляет использовать несколько технологий для их экономии. Некоторые из технологий могут включать:

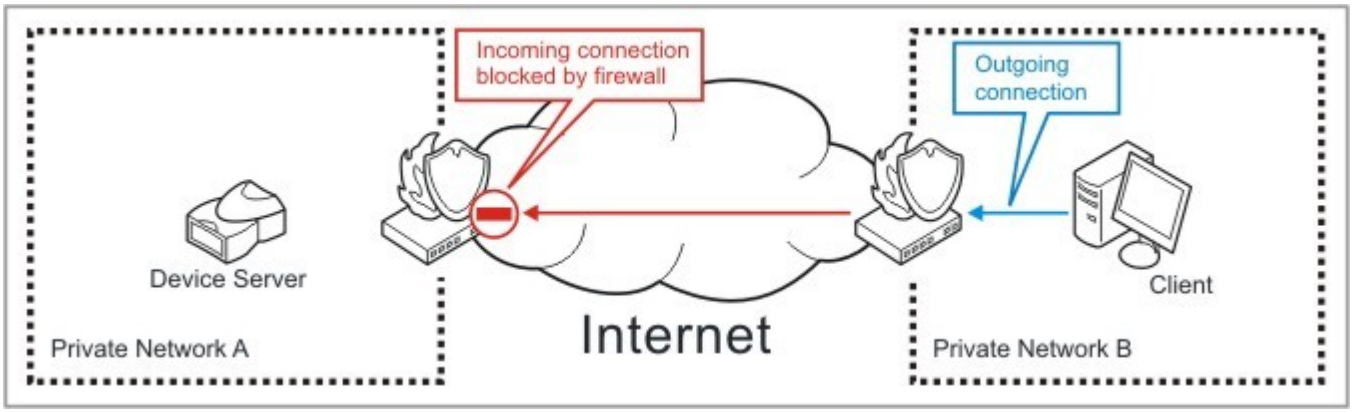
- *Динамические* IP-алреса, когда IP-адреса хостов меняются время от времени,
- *NAT* (Network Address Translation), когда несколько хостов используют один "внешний" IP-адрес на маршрутизаторе для связи с остальным миром.

Эти технологии позволяют каждому хосту создавать исходящие подключения (например, для открытия веб-сайта). Однако, может быть сложно (или вообще невозможно) создать подключение к такому хосту - Вы даже не можете узнать его IP-адрес (он динамический) или не знать его внешний адрес.

Единственным стандартным решением этой проблемы была бы привязка статического IP-адреса к каждому хосту, к которому Вам надо подключаться. Эти статические IP-адреса прежде должны быть получены. Например, Вам их надо выкупить у Вашего ISP. Один IP-адрес будет стоить не дорого, но когда системе необходимо взаимодействовать с большим количеством таких узлов, это может стоить значительных денег. Даже когда сеть частная и IP-адреса бесплатны, следует учитывать стоимость работы по администрированию и выделению таких адресов.

Файерволами, блокирующими входящие пакеты

Даже если предположить, что кто-то получил достаточное количество статических IP-адресов, всё еще имеется другая проблема: Вам необходимо *подключиться* к Серверам Устройств. Это значит пройти файерволы. Обычно файерволы ограничивают входящий траффик, так что Вам прийдется выделять диапазоны портов и пр. Это требует дополнительной работы и снижает безопасность сети (больше IP-адресов и портов доступно извне - больше риск).



Возможным решением было бы настроить все Серверы Устройств на исходящие соединения и заставить их соединяться с конкретным хостом. Это позволило бы использовать только один статический адрес - для этого хоста. К несчастью это означает, что только один этот хост будет способен взаимодействовать с Вашими Серверами Устройств. Как видите, это решение не идеально.

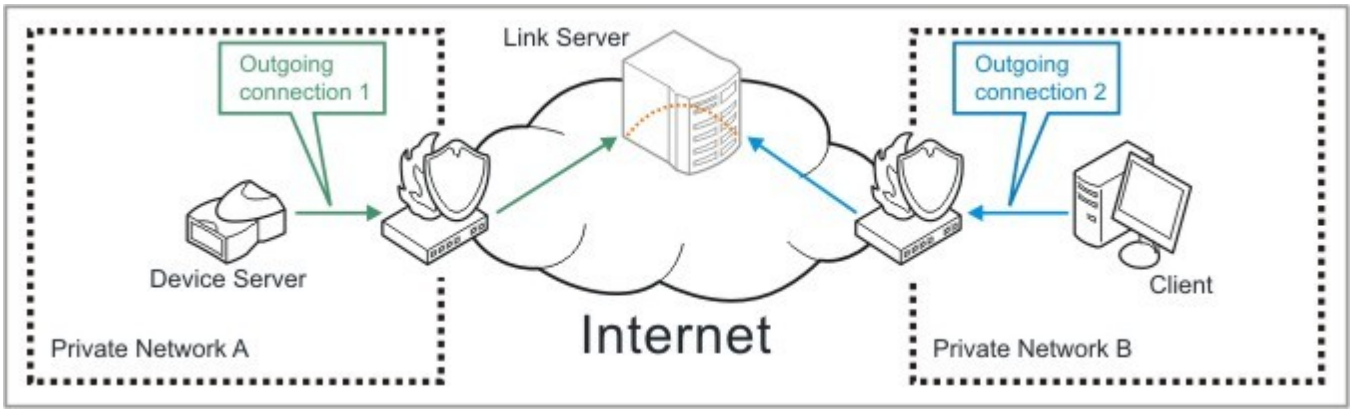
Так какой же способ соединить Сервера Устройств и их "клиентские" хосты, когда они находятся в разных сетях и не имеют статических адресов наилучший? Он называется Link Service.

Сервис связи

Для сетевого хоста проще к кому-то подключиться, чем принять соединение. К несчастью, обычно одна из сторон должна все-таки принять входящее соединение.

AtomMind предлагает обойти эту проблему, позволяя всем сторонам взаимодействовать через промежуточный сервис - Link Service так, что они все открывают исходящие соединения.

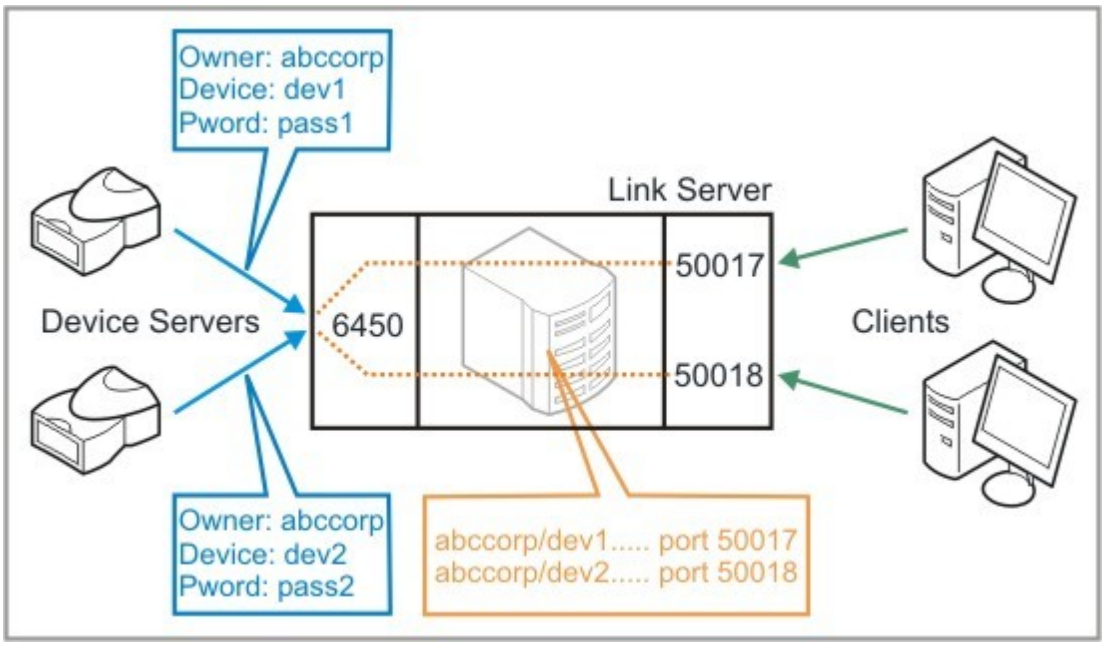
Когда Вы и Ваш друг используют ICQ или MSN, вы оба подключаетесь к центральному серверу. Никто из вас обычно не имеет статического IP-адреса, но сервер имеет. Обе стороны создают исходящее соединение и не нужно перенастраивать файерволы и иметь статические IP.



Link Service - точно такое же решение, только сделанное специально для Серверов Устройств ТВЭЛ! Ваши Сервера Устройств подключаются к Link Service (создают исходящее соединение). Хосты, желающие взаимодействовать с Серверами Устройств также подключаются к AtomMind Server'у. Обе стороны "встречаются" на AtomMind Server'е и могут взаимодействовать друг с другом.

Сервис связи: Детали реализации

Каждый Сервер Устройств, которому необходимо использовать *Link Service* должен быть сначала зарегистрирован на AtomMind Server'е, т.е. иметь профиль Сервера Устройств 2087. AtomMind Server распознаёт Серверы Устройств комбинацией данных следующих настроек: **Имя устройства (DN)**, **Имя владельца (ON)** и **Пароль (PW)**. Каждому зарегистрированному Серверу Устройств выделен уникальный номер порта при регистрации. Это тот порт, к которому должен подключаться клиент для взаимодействия с конкретным Сервером Устройств.



Сервер Устройств входит в AtomMind Server по имени устройства, владельца и паролю. Обычно СУ настроен это делать сразу после загрузки. У AtomMind Server'а имеется стандартный порт для входа - 6450 по умолчанию. Поскольку Серверы Устройств уникально идентифицируются своим именем, именем владельца и паролем, одного порта для входа всем достаточно.

"Клиентский" хост, желающий взаимодействовать с конкретным Сервером Устройств создает соединение к конкретному порту AtomMind Server'а, который был выделен Серверу Устройств в процессе регистрации. Связь создается и обе стороны могут обмениваться данными.

Клиентский хост не нуждается в особой процедуре входа для создания подключения к Серверу Устройств через AtomMind Server. Вместо подключения к IP-адресу и порту Сервера Устройств напрямую он подключается к IP-адресу и порту AtomMind Server'а, привязанному к конкретному устройству - разница только в этом.

Сервис связи реализован через драйвер устройства 518 прозрачного проброса данных (Сервис связи) 2093. Подробности в этой главе.

19.5.4 Серверы устройств, аккаунты серверов устройств, акка

Ниже мы рассмотрим разницу между оборудованием Серверов Устройств, их аккаунтами и серверами внешними Устройств. Эти термины широко используются в этой документации и начиная с этого момента важно понимать разницу между ними.

Снова, AtomMind оперирует тремя типами сущностей:

- Оборудованием Серверов Устройств
- Профилями Серверов Устройств 2087
- Профилями внешних Серверов Устройств 2096

Первый термин ссылается на оборудование. Вторые два - на логические части AtomMind Server.

Важно понимать, что единица оборудования может обрабатываться AtomMind'ом как обычный Сервер Устройств (т.е. Сервер Устройств с профилем) и *также* как внешний (т.е. как внешний Сервер Устройств с профилем) одновременно. Оборудование Сервера Устройств может быть настроено для работы с AtomMind Server как обычный Сервер Устройств и входить при включении, но если AtomMind Server обнаружит его при широковещательном опросе 2096 внешних Серверов Устройств, профиль внешнего Сервера Устройств будет для него создан атоматически.

Аппаратные Сервера Устройств

Аппартные Сервера Устройств в большинстве страниц документации называются просто Серверами Устройств. Это аппартное устройство, связывающее Устройства 497 и AtomMind Server. Аппартаный Сервер Устройств может быть физически реализован в качестве отдельного устройства или встроен в Устройство и расположен на его печатной плате (PCB). Когда Сервер Устройств представляет собой отдельную аппартную часть, он обычно соединен с Устройством посредством RS-232, ZigBee или WiFi. Когда Сервер Устройств встроен, связь между схемой Сервера Устройств и основной платой Устройства осуществляется через последовательный интерфейс.

В Агенте 684, Серверы Устройств и Устройства унифицированы физически и логически. В любом случае, логика Сервера Устройств реализована пользовательским приложением, запущенным на программируемой интегральной микросхеме. Другая часть того же приложения реализует логику самого Устройства.

Когда аппартный Сервер Устройств подключается к AtomMind Server'у и пытается войти , сервер проверяет, существует ли соответствующий профиль Сервера Устройств и не заблокирован ли он.

Профили Серверов Устройств

Профиль Сервера Устройств - это особый контекст 41 AtomMind Server'а, содержащий информацию об одном аппартном Сервере Устройств, который может подключаться к AtomMind Server'у. Этот профиль определяет пароль, который должен использоваться аппартным Сервером Устройств для входа, драйвер устройства 518 обрабатывающий данные от устройства и некоторые другие настройки.

Технически возможно создать профиль, не связанный ни с каким аппартным Сервером Устройств. Такие профиль используются для тестирования, отладки и пр.

В отличие от профилей Внешних Серверов Устройств, обычные профили Серверов Устройств используются для "приёма" подключений с аппартных Серверов Устройств. Каждый аппартный Сервер Устройств, который будет соединяться с AtomMind Server'ом и обмениваться с ним данными должен иметь соответствующий профиль Сервера Устройств.

Профили внешних Серверов Устройств

Основное назначение **профилей внешних Серверов Устройств** - позволить AtomMind Server'у иметь доступ к Серверам Устройств, которые не настроены самостоятельно подключаться к AtomMind Server'у. В большинстве случаев этого достаточно для начальной конфигурации аппаратного Сервера Устройств, который только что был подключен в сеть и включен.

Разница типов этих профилей в том, что профиль Внешнего Сервера Устройств используется AtomMind Server'ом для активного соединения с любым из Серверов Устройств, а обычные профили Серверов Устройств используются для приема соединений от "известных" аппаратных Серверов Устройств.

Каждый профиль Внешнего Сервера Устройств содержит информацию об IP- и MAC-адресах соответствующего Сервера Устройств, предпочтительный способ связи для доступа к нему и некоторую другую информацию, необходимую для подключения.

В процессе ежедневного использования AtomMind Server'а Внешние Сервера Устройств используются для начальной конфигурации аппаратуры Серверов Устройств (чтобы они подключались к AtomMind Server'у при включении). Когда аппаратура Сервера Устройств правильно настроена и подключена к AtomMind Server'у, будет использоваться обычный профили Сервера Устройств.

Для простоты профили Внешних Серверов Устройств называются "Внешние Сервера Устройств" в этой документации.

19.5.5 Серверы Устройств

Сервер устройств - это устройство, работающее как "мост" между одним или несколькими Устройствами и AtomMind Server'ом. Он имеет несколько важных предназначений:

- Подключение к сети одного или нескольких Устройств
- Позволяет AtomMind Server'у обнаруживать Устройства
- Подключение и вход AtomMind Server при включении и передаче данных между AtomMind Server'ом и Устройством

В AtomMind каждое Устройство может быть логически подключено к AtomMind Server'у через Сервер Устройств, а Сервер Устройств может быть реализован несколькими разными способами::

- Как отдельное устройство. В этом случае Устройство может быть подключено по серийному порту, WiFi или ZigBee.
- Как встроенный в Устройство ethernet-модуль, подключенный к имеющимся в устройстве компонентам.
- Как часть интегральной микросхемы (IC) Устройства. В этом случае логика Сервера Устройств реализуется посредством специального приложения Агент 684, запущенного на этой IC.

Важно помнить, что не смотря на то, что Сервер Устройств и Устройство может быть физически объединены, AtomMind всё равно считает их отдельными частями.

Для Серверов Устройств подключенных и зашедших в AtomMind Server Вам необходимо создавать *Учётную запись*. Учётная запись Сервера Устройств - это постоянно хранящаяся запись в базе данных 692 AtomMind Server'а, говорящая что-то вроде: Сервер Устройств с именем "*device1*" (то же, что и настройка **Имя Устройства** в Сервере Устройств) принадлежит пользователю AtomMind Server'а "*user1*" (соответствует настройке Сервера Устройств **Имя Владельца**) и ему позволено входить на AtomMind Server с паролем "*pass1*".

В этой статье часто используется два термина: *аппаратура* Сервера Устройств и профиль Сервера Устройств. Разница между ними объяснена в отдельной главе 2086.

Профиль Сервера Устройств также включает в себя иные настройки, определяющие как AtomMind Server будет относиться к данным, получаемым из Сервера Устройств и какого типа данные (т.е. слать ли команды или только "чистые" данные) он посылает в Сервер Устройств. Эти настройки объясняются позже на странице Свойства Сервера Устройств 2089.

Профиль Сервера Устройств может соответствовать, а может и не соответствовать имеющемуся в наличии оборудованию. Вы можете даже создать профиль Сервера Устройств, которого на самом деле нет.

Если несколько Серверов Устройств захотят войти от одной и той же учётной записи, только один из них останется подключенным. Каждый аппаратный Сервер Устройств должен иметь отдельный профиль.

Профиль Сервера Устройств всегда принадлежит пользователям AtomMind Server'а и зарегистрирован под каким-либо из пользовательских аккаунтов 478. Каждый пользователь имеет собственные профили Серверов Устройств.

По умолчанию каждый пользователь способен владеть профилями Серверов Устройств. Пользователи с достаточными правами 477 (включая администратора по умолчанию) также могут управлять профилями Серверов Устройств других пользователей.

Все операции с Серверами Устройств подвержены воздействию глобальной настройки AtomMind Server'а Максимальное время ожидания выполнения операций с Сервером Устройств.

КАК СЕРВЕР УСТРОЙСТВ ВЗАИМОДЕЙСТВУЕТ С ATOMMIND SERVER'ОМ

После включения правильно настроенный на работу с AtomMind Сервер Устройств инициирует соединение с AtomMind Server'ом и выполняет вход 2088. Когда вход выполнен, AtomMind Server передает управление Драйверу Устройства 518, настроенному для этого профиля. Драйвер определяет, что за данные посылаются в Сервер Устройств и как обрабатывать данные из устройства.

Когда Сервер Устройств залогинен в AtomMind Server, Вы можете:

- Менять его внутренние настройки 2090.
- Идентифицировать 2091 или перегрузить 2091 его.
- Отслеживать поток данных 2091 в или из Сервера Устройств
- Контролировать, настраивать или следить за Устройствами 497 подключенными к нему.

СОЗДАНИЕ УЧЁТНОЙ ЗАПИСИ СЕРВЕРА УСТРОЙСТВ

Есть три способа создания учётных записей Серверов Устройств:

- **Регистрация вручную**. Этот метод подходит, когда Сервер Устройств вручную 2099 настроен для работы с AtomMind Server'ом и автоматическая регистрация 2089 Сервера Устройств отключена. В этом случае аппаратный Сервер Устройств не сможет зайти в AtomMind Server, пока не будет создан профиль при помощи действия Новый Сервер Устройств 2108 контекста Серверы Устройств. Настройка **Имя Владельца** Сервера Устройств должна совпадать с именем владельца пользовательского аккаунта, настройка **Имя Устройства** должна совпадать с именем аккаунта и пароль, указанный в настройках Сервера Устройств должен совпадать с паролем из настроек аккаунта. Сетевые настройки Сервера Устройств должны быть установлены так, чтобы он соединялся с AtomMind Server'ом при включении. Тогда он успешно подключится.
- **Автоматическая регистрация**. Если Сервер Устройств был настроен вручную 2099 для подключения к AtomMind Server'y и автоматическая регистрация 2089 Сервера Устройств включена и имя существующего пользователя 478 совпадает с **Именем Владельца** из настроек Сервера Устройств, новый профиль Сервера Устройств будет создан автоматически как только Сервер Устройств попытается войти на AtomMind Server. Более подробно см. последовательность входа 2088.
- **Регистрация с использованием процедуры "Подключить Внешний Сервер Устройств к AtomMind Server'y"**. Когда внешний Сервер Устройств (который не был зарегистрирован) настраивается на соединение с AtomMind Server'ом при включении питания самим AtomMind Server'ом, соответствующий профиль Сервера Устройств создается автоматически при этой операции. Подробнее в этом разделе 2097. **Эта процедура является рекомендуемой**. При этом есть уверенность, что свойства 2089 профиля Сервера Устройств совпадают с настройками 2090 аппаратного Сервера Устройств.

Администрирование Серверов Устройств

Для администрирования Серверов Устройств используются два контекста: Один - общий контекст Серверы Устройств 2108, работающий как контейнер всех профилей Серверов Устройств, принадлежащих определенному пользователю. Второй - контекст Сервер Устройств 2102, соответствующий единственному профилю Сервера Устройств.

Еще один дополнительный контекст может быть использован для управления Серверами Устройств: **Все Серверы Устройств,** находящийся в корневом 1559 контексте. Этот контекст предоставляет доступ ко всем Серверам Устройств в системе и групповые действия 101 с ними. Он видим только для текущего пользователя с уровнем доступа 477 администратора в корневом контексте.

Последовательность входа

Как Сервер Устройств входит в AtomMind Server:

1. При включении аппаратное обеспечение Сервера Устройств создает TCP-соединение к AtomMind Server'y. Адрес и порт AtomMind Server'a определяется в настройках **IP назначения** и **Порт назначения** Сервера Устройств. Если при подключении возникает какая-то проблема, Сервер Устройств сигнализирует об ошибке 2101 светодиодами.
2. Когда соединение создано, Сервер Устройств начинает обмениваться командами с AtomMind Server'ом.
3. AtomMind Server проверяет, совместима ли с ним прошивка Сервера Устройств. Если нет - соединение разрывается.
4. AtomMind Server проверяет настройку **Имя Владельца** Сервера Устройств. Если профиль пользователя с таким именем не существует, соединение разрывается.
5. Аналогичная проверка делается для **Имени Устройства.** Если профиля Сервера Устройств с таким именем не существует под пользовательским аккаунтом, определённом на предыдущем шаге, связь прерывается. Однако, если включены обе настройки **Позволена автоматическая регистрация Серверов Устройств** в пользовательских настройках плагина Серверы Устройств 2093 и **Авторегистрация на AtomMind Server'e** в настройках аппаратного Сервера Устройств, создается новый профиль для Сервера Устройств. Подробнее см. Автоматическая регистрация профилей Серверов Устройств 2089.
6. Если профиль Сервера Устройств существует, но заблокирован, соединение прерывается.
7. Если пароль из настроек Сервера Устройств не совпадает с паролем из профиля Сервера Устройств, соединение прерывается.

8. Сервер Устройств регистрируется в DNS 2082, если эта опция включена в свойствах профиля 2089.
9. Если настройка **Драйвер Устройства** установлена в **Автоопределение**, AtomMind Server спрашивает у сервера устройств о предпочтительном драйвере устрройства 518. Когда сервер устройств предоставляет тип драйвера, он записывается в профиль сервера устройств для использования при последующих входах. Если сервер устройств не способен предоставить информацию о типе предпочтительного драйвера, для него будет использован драйвер Динамического DNS 2094.
10. Драйвер Устройства 518 определённый в настройках Сервера устройств начинает обрабатывать данные из Сервера Устройств и Устройств, подключенных к нему.

Автоматическая регистрация профилей Серверов Устройств

Эта возможность позволяет Серверам Устройств, неизвестным AtomMind Server'у (т.е. у которых нет соответствующего профиля Сервера Устройств) автоматически зарегистрироваться на сервере.

Чтобы это работало, должно выполняться несколько условий:

1. Сервер Устройств должен быть правильно настроен 2099 для подключения к AtomMind Server'у при включении питания,
2. Настройка **Авторегистрация на** AtomMind Server**'е (AR)** Сервера Устройств должна быть включена,
3. Параметр **Позволить Авторегистрацию Сервера Устройств** должен быть включен в конфигурации плагина Серверов Устройств 2093 или его имя должно соответствовать **Имени Владельца** Сервера Устройств.

Как только условия выполнены и Сервер Устройств пытается войти на AtomMind Server, когда он ещё не зарегистрирован (т.е. нет профиля Сервера Устройств), AtomMind Server создает профиль с настройками по умолчанию для него. После автоматической регистрации аппаратный Сервер Устройств отключается от сервера и подключается заново в течении нескольких секунд. Теперь у него есть профиль и он может войти.

Подробнее см. последовательность входа 2088.

Авторегистрация должна быть отключена из соображений безопасности в реально используемой системе. Советуем включать её лишь на короткое время для ускорения развертывания AtomMind.

Свойства профиля Сервера Устройств

Эти свойства доступны через действие Редактировать свойства Сервера Устройств 2102 в любом контексте Сервера Устройств.

Имя Владельца	Имя пользователя-владельца Сервера Устройств. Соответствует настройке **Имя Владельца** Сервера Устройств. Свойство только для чтения.
Имя Сервера Устройств	Имя профиля Сервера Устройств (также имя контекста Сервера Устройств 2102). Соответствует настройке **Имя Устройства** в Сервере Устройств. Только для чтения.
Пароль	Пароль профиля. Сервер Устройств должен использовать этот пароль в последовательности входа 2088.
Описание	Комментарий об устройстве.
Регистрировать в DNS	Определяет, должен ли Сервер Устройств быть зарегистрирован в DNS 2082 во время логина на AtomMind Server.
Заблокирован	Серверу Устройств запрещён вход, если его профиль заблокирован.
Разрешены команды в потоке данных для клиента	Определяет, может ли драйвер устройства 518, связанный с этим профилем посылать команды к Серверу Устройств. Большинство драйверов устройств сами не шлют команды, но некоторые драйверы пересылают команды, полученные от сторонних приложений, управляющих аппаратурой Сервера Устройств. Чтобы позволить передавать такие команды драйверу эту опцию необходимо включить. Слова "в потоке" означают, что команды встраиваются в поток данных TCP от AtomMind Server'а к Серверу Устройств. В AtomMind Server'е все команды "в потоке" - это лишь устаревшее обозначение со времён, когда Сервера Устройств обычно управлялись по UDP (отдельно от потока данных TCP, "вне потока"). Эта опция должна быть отключена в большинстве случаев, если Вы точно не знаете, должна ли она быть использована с Вашим устройством. Если драйвер устройств посылает или перенаправляет команды в потоке данных для Сервера Устройств в то время, как эта опция отключена в самом

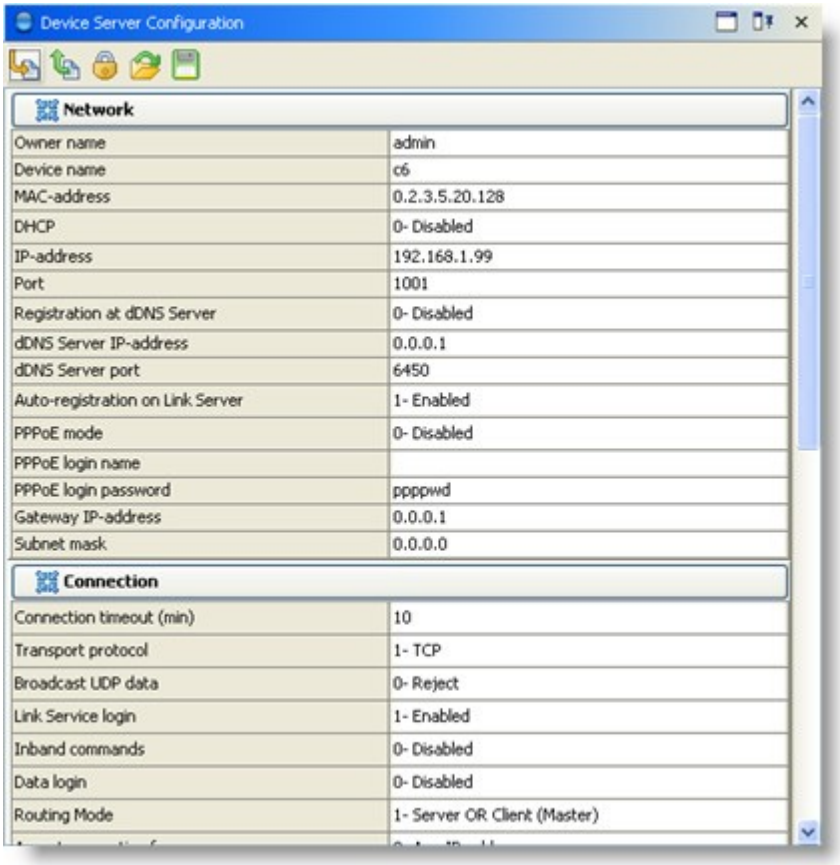
	сервере устройств, команды не будут "поняты" им. В этом случае поток данных будет поврежден.
Временная зона	Временная зона Сервера Устройств и всех подключенных в него Устройств 497. Подробнее об этом можно прочесть в Работаем с временными зонами 911.
Драйвер устройства	Определяет, что за драйвер устройства 518 используется AtomMind Server'ом для работы с Сервером Устройств. Значение **Автоопределение** заставляет AtomMind Server спрашивать сервер устройств о предпочтительном драйвере при попытке входа. Автоматически определённый тип драйвера будет сохранён в свойствах профиля и использован при последующих входах этого устройства.

Редактирование настроек оборудования Сервера Устройств

Вы можете менять настройки оборудования Сервера Устройств, пока он поддключен к AtomMind Server'у. Эта возможность сначала была реализована в программе *DS Manager*. Настройки Сервера Устройств доступны из действия Настроить Сервер Устройств 2102 в контексте Сервер Устройств 2102. При сохранении настроек система спросит пользователя о необходимости перезагрузки Сервера Устройств.

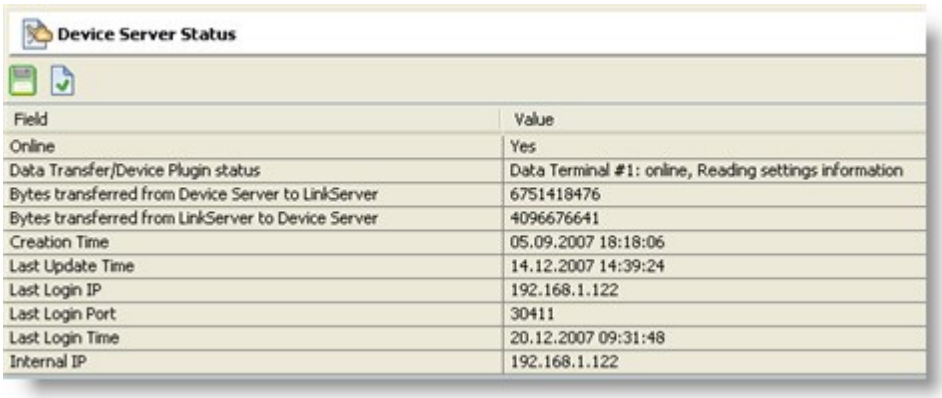
Изменение некоторых настроек (таких, как настройки сети) Сервера Устройств могут не позволить ему подключиться к AtomMind Server'у после перезагрузки. В этом случае устройство будет полностью недоступно для AtomMind, особенно, если оно располагается в удалённой сети, защищенной файерволом или иными методами. Будьте осторожны при изменении настроек Сервера Устройств.

Вот как выглядит диалог настроек Сервера Устройств в Клиенте:



Статус Сервера Устройств

Вы можете использовать действие Просмотреть статус Сервера Устройств 2103 из любого контекста Сервера Устройств чтобы увидеть, что с ним происходит:



Статус профиля включает следующее:

Он-лайн	Показывает, он-лайн ли Сервер Устройств (подключен ли и вошел ли он в AtomMind Server).
Статус драйвера устройства	Статус драйвера устройства 518. Выдается драйвером и зависит от логики работы драйвера.
Байт переданных от Сервера Устройств к AtomMind Server'у	Количество данных, посланных к Серверу Устройств
Байт переданных от AtomMind Server'а к Серверу Устройств	Количество данных, Переданных на Сервер Устройств
Время создания	Только для чтения. Показывает, когда профиль был создан.
Время последнего обновления	Только для чтения. Показывает, когда последний раз менялись свойства профиля.
Последний IP входа	IP-адрес, с которого произошёл вход. Это "внешний" адрес, т.е. адрес, с которого Сервер Устройств видится AtomMind Server'ом.
Последний порт входа	Номер порта, с которого последний раз входил Сервер Устройств.
Время последнего входа	Время последнего входа Сервера Устройств.
Внутренний IP	Внутренний адрес, установленный в Сервере Устройств, когда он последний раз входил. Это адрес Сервера Устройств в его локальной сети. Настройка *IP-адрес* в Сервере Устройств. Он может отличаться от "внешнего" адреса (последний IP входа, выше).

Другие операции с Сервером Устройств

Две другие операции могут быть выполнены на Серверах Устройств:

- Перезагрузка
- Жужжание

Обе они доступны через любой контекст Сервера Устройств 2102.

ИДЕНТИФИКАЦИЯ

Идентификация позволяет Вам визуально обнаружить Сервер Устройств по его профилю. Оно заставляет оборудование проиграть шаблон быстрого мигания красными и зелеными статусными LED'ами. Это простой способ идентифицировать устройство по его аккаунту в AtomMind Server'е.

ПЕРЕЗАГРУЗКА

Эта операция перезагружает Сервер Устройств. Перезагрузка может потребоваться для активации некоторых настроек Сервера Устройств после их изменения.

Отслеживание передачи данных

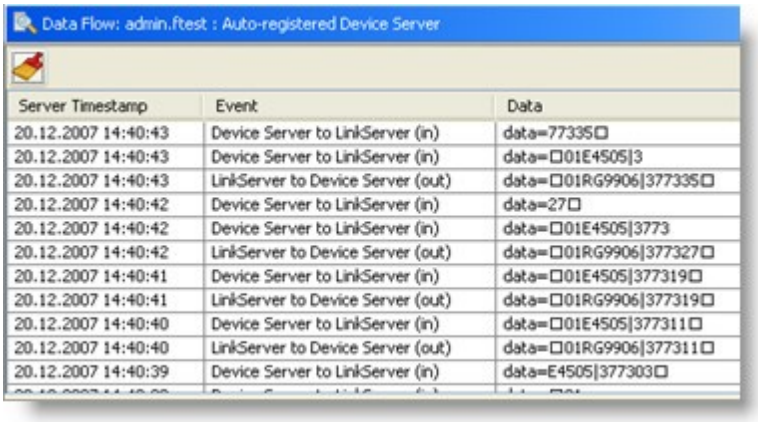
Это позволяет Вам отслеживать передачу сырых данных переданных или полученных от Сервера Устройств в режиме реального времени и просматривать журнал ранее переданных данных.

Вы можете использовать действие Просмотр передачи данных 2103 любого контекста Сервера Устройств. Это действие запускает 96 журнал событий, отслеживающий два типа событий:

- Сервер Устройств в AtomMind Server (in)
- AtomMind Server в Сервер Устройств (out)

Можно просматривать журнал обоих типов событий в режиме реального времени. Откройте отдельное событие для просмотра 90 всех его данных, если для их отображения недостаточно места в просмотрщике Журнала Событий.

Вот как выглядит отслеживание передачи данных в Клиенте:



19.5.6 Плагины серверов устройств

Плагины серверов устройств - разновидность плагинов 207, определяющих, как AtomMind Server взаимодействует с потоком данных, получаемым с сервера устройств. Обычно это означает взаимодействие с аппаратными устройствами, подключенными к AtomMind через эти сервера устройств.

Уровни настроек

Каждый плагин серверов устройств имеет три уровня настроек:

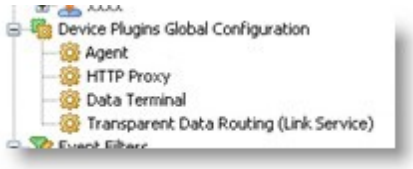
- **Глобальные настройки**. Те, что влияют на работу плагина в целом. Глобальные настройки могут редактироваться только пользователями с необходимым уровнем прав.
- **Пользовательские настройки**. Они влияют на работу плагина только в том случае, если связанный с ним сервер устройств принадлежит конкретному пользовательскому профилю 478. Когда плагин обрабатывает данные от некоторого сервера устройств, он использует *пользовательские настройки*, хранящиеся в профиле владельца.
- **Настройки уровня сервера устройств**. Эти настройки влияют на поведение драйвера, связанного с конкретным профилем сервера устройств 2087. Когда драйвер обрабатывает данные с конкретного сервера устройств, он использует *настройки сервера устройств*, хранящиеся в профиле сервера устройств.

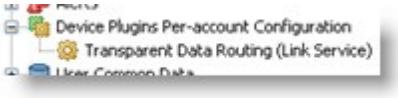
Большинство плагинов серверов устройств не использует все три уровня настроек.

Новые профили серверов устройств используют настройки плагинов серверов устройств, определённые в глобальной конфигурации настроек плагинов по умолчанию для серверов устройств 179. Тип плагина может быть позже изменен редактированием настроек профиля сервера устройств.

Администрирование плагинов серверов устройств

Для администрирования плагинов серверов устройств используются два типа контекстов: первый - общий контекст Конфигурации драйверов/плагинов 1505, обрабатываемый как контейнер. Второй - контекст Конфигурации драйверов/плагинов 1506, хранящий конфигурацию единственного плагина.





Они находятся в двух местах:

- В Корневом 1559 контексте, где они содержат глобальные настройки
- В Пользовательском 1608 контексте, где они содержат настройки уровня пользователя

Если плагин не содержит настроек на одном из этих уровней, соответствующий конфигурационный контекст не будет создан (не будет присутствовать в Системном Дереве).

Можно получить доступ к настройкам **Уровня Device Server** драйвера при помощи действия Сконфигурировать драйвер 2103 Device контекста Серверы устройств 2102.

Плагины серверов устройств

Плагин серверов устройств - это плагин 207, отвечающий за общее взаимодействие с сервером устройств и обработку данных. Он взаимодействует с конкретными специфичными для сервера устройств плагинами, отвечающими за обработку данных для конкретного сервера устройств.

Плагин серверов устройств имеет следующие глобальные настройки:

- **Номер порта прослушивания для соединений с серверами устройств (от 0 до неактивированного)**. Эта опция определяет номер порта для входов в сервер устройства, т.е. порт, с которым устройство должно установить соединение. Если установлен на ноль, небезопасные соединения с серверами устройств запрещаются.
- **Драйвер сервера устройства по умолчанию**. Эти опции определяют, какой драйвер устройства 518 используется для обработки данных вновь созданным контекстом сервера устройства. Это идентификатор драйвера по умолчанию, не его имя. Эта опция обычно редактируется с помощью Редактора виджетов (такого как AtomMind Client), включающего все необходимые идентификаторы. Информация о поиске необходимого идентификатора драйвера для ручного ввода в файл конфигурации выходит за пределы данной документации. У этой опции по умолчанию стоит драйвер Динамический DNS 2094. Если опция настроена на **Автоматическое определение**, сервер выставит **Драйверы устройств** в настройках вновь созданного аккаунта сервера устройств 2087 на автоопределение. Реальный используемый драйвер будет предоставлен Сервером устройств во время его первого входа в систему и будут сохраняться в настройках учетной записи Сервера устройств.
- **Время ожидания обычных команд**. Эта опция определяет время ожидания для обычных операций серверов устройств (вход, конфигурация внешних устройств и т.д.)
- **Время ожидания для внутриполосных команд (секунды)**. Эта опция определяет время ожидания для внутриполосных команд, отправляемых серверу устройств, пока оно подключено к серверу. Это относится ко всем серверам устройств, сконфигурированным через обычные учетные записи серверов устройств и к внешним устройствам 2095, сконфигурированным при помощи внутриполосных команд TCP.
- **Время ожидания для широковещательных команд (секунды)**. Эта опция определяет время ожидания для команд, настроенных на широковещательный режим, например, во время автоматического обнаружения внешних серверов устройств 2096.

Плагин серверов устройств имеет следующую пользовательскую настройку:

- **Включить авторегистрацию серверов устройств.** Определяет, должны ли профили серверов устройств 2087 создаваться автоматически для каждого сервера устройств, пытающегося войти. И включена ли при этом *настройка автоматической регистрации в* AtomMind Server*'е*. См. главу Авторегистрация 2089 серверов устройств.

19.5.6.1 Прозрачная маршрутизация данных (Link Service)

Плагин *Прозрачная маршрутизация данных* реализует функционал Сервиса связи 2084. Когда этот драйвер включен в свойствах профиля 2089 данного сервера устройств, как только этот сервер устройств подключен, AtomMind Serverначинает прослушивать TCP-соединения на выделенном "клиентском порту", указанному в настройках драйвера. Если некоторое оборудование (например, другой сервер устройств), или программа (какая-то программа на ПК) подключается к этому порту, AtomMind Server начинает работать как посредник: Он прозрачно передает данные между этим клиентом и устройством, подключенным к серверу устройств. Он перенаправляет все данные, полученные с сервера устройств (с оборудования, "спрятанного" за сервером устройств) клиенту. Все данные, *получаемые* от клиента отправляются на сервер устройств (и сервер устройств передаёт их "на другую сторону", т.е. спрятанному за ним устройству).

Подробнее см. Сервис связи 2084.

Информация о плагине

Идентификатор плагина: com.tibbo.linkserver.plugin.device.transparentlink

Глобальные настройки

- **Минимальный номер порта для серверов устройств.** По умолчанию 50000.
- **Максимальный номер порта для серверов устройств.** По умолчанию 60000.

Эти две опции определяют диапазон номеров портов, которые могут быть выделены для подключений Клиентов в настройках драйвера серверов устройств. Этот диапазон может быть переназначен настройками пользователя.

- **Список номеров портов, запрещенных к использованию с серверами устройств, разделенных запятыми**. Это исключения из диапазонов, указанных выше. Эти порты не могут быть присвоены клиентам ни автоматически, ни вручную.

Пользовательские настройки

- **Минимальный номер порта для серверов устройств под этим пользовательским профилем.**
- **Максимальный номер порта для серверов устройств под этим пользовательским профилем.**

Эти настройки переназначают диапазон клиентских портов, указанный в глобальных настройках.

Настройки уровня Сервера Устройств

- **Номер порта для клиентских соединений (нуль для автоматического назначения)**. По умолчанию нуль. Этот порт используется клиентами для соединения и оправки данных конкретному серверу устройств и устройству. Драйвер ожидает на этом порту только после того, как связанный с ним сервер устройств подключился и вошел в AtomMind Server. Только один клиент может подключаться к одному и тому же порту одновременно, прочие соединения будут отвергнуты. Если эта настройка установлена в нуль, драйвер найдет свободный порт (не связанный с каким-либо другим сервером устройств) из диапазона портов, указанных в глобальной или пользовательской конфигурации. Автоназначение порта производится во время первого входа сервера устройств, при всех последующих входах данный сервер устройств будет получать выделенный ему ранее порт, поскольку меняется внутренняя настройка драйвера. Вы можете узнать, какой порт в настоящий момент выделен серверу устройств, взглянув 1506 в настройки драйвера после входа сервера устройств.
- **Контроль сервера устройств**. Возможные значения: **Нет** и **линия DTR сигнализирует о статусе Клиентского соединения**. Во втором случае линия DTR сервера устройств возбуждается посылкой команды в потоке данных к серверу устройств, когда новый клиент подключается к клиентскому порту сервера устройств. Линия DTS сбрасывается при отключении клиента
- **Контроль клиента**. Возможные значения **Нет** и **Сброс клиента при пульсации DSR**. Во втором случае клиентское подключение завершается, если получены два следующих один за другим извещения об изменении состояния линии DSR от сервера устройств за короткий период времени.

19.5.6.2 Динамический DNS

Плагин *Динамический DNS* включается для создаваемых профилей Серверов Устройств 2087, как указано в значении по умолчанию глобальной конфигурации Плагина Серверов Устройств 179. Этот драйвер просто отбрасывает любые данные, полученные от Сервера Устройств и никогда не пытается ничего послать к нему. Он называется плагином Динамического DNS, потому что позволяет Серверу Устройств войти на AtomMind Server, зарегистрироваться в DNS 2082 и затем отключиться. Он также может использоваться для тестирования подключения Сервера Устройств.

Несмотря на имя этого драйвера, функционал Динамического DNS встроен в ядро AtomMind Server'a. Драйвер динамического DNS просто помогает Серверам Устройств оставаться подключенными после входа. Он работает как "заглушка", которая ничего не делает с подключенным Сервером Устройств. Другими словами, он поддерживает соединение активным без всякой передачи данных. Пока соединение активно, пользователь может проверить состояние Сервера Устройств, настроить, перезагрузить или идентифицировать его и т.д. Подробнее см. главу Сервера Устройств 2087.

Чтобы быть уверенным в том, что сервер устройств будет регестрироваться в DNS при входе на AtomMind Server, проверьте, что:

- Динамический DNS включен и верно настроен в глобальных настройках 1425 AtomMind Server'a
- Настройка **Зарегистрироваться в DNS** включена в настройках аккаунта сервера устройств 2089

См. Сервис динамического DNS 2082 для подробной информации.

Информация о драйвере

Идентификатор плагина 518: com.tibbo.linkserver.plugin.device.stub

Глобальные настройки

Не определены.

Пользовательские настройки

Не определены.

Настройки уровня Сервера Устройств

Не определены.

19.5.6.3 HTTP-прокси

Плагин *HTTP-прокси* позволяет Вам получить доступ к встроенному в сервер устройств веб-серверу, если сервер устройств подключен к AtomMind Server'у и находится в частной сети или у него нет статического IP-адреса. Доступ осуществляется по HTTP.

При запуске AtomMind Server'а этот драйвер устанавливает особое веб-приложение во встроенный в AtomMind Server веб-сервер. Это приложение превращает AtomMind Server в HTTP-прокси сервер, предоставляющий доступ к серверам устройств, подключенным к нему, способным работать с HTTP-прокси и имеющим собственные встроенные веб-страницы.

НОВЫЙ ТЕРМИН: *Прокси-сервер* - это программа, позволяющая клиентам создавать непрямое соединение к другим узлам в сети. Клиент подключается к прокси-серверу, запрашивает соединение, файл или другой ресурс, доступный на другом сервере, а прокси-сервер предоставляет этот ресурс с удалённого узла..

Более подробную информацию о сервисе HTTP Proxy можно найти в главе Сервис HTTP Proxy 2081 в разделе Основы Системы 38.

Когда драйвер запущен, следующие URL могут использоваться для доступа к странице, предоставляемой встроенным в сервер устройств веб-сервером, когда сервер устройств подключен к AtomMind Server'у:

`https://server_address/dev/device_server_owner/device_server_name/page_path/page_name`

`http://server_address/dev/device_server_owner/device_server_name/page_path/page_name`

Первый пример использует защищённое HTTP-соединение Между веб-браузером и AtomMind Server'ом. Во втором примере используется незащищённое соединение.

`server_address` - IP AtomMind Server'а или один из его DNS-адресов, определённых в Псевдонимах Хост-Имен. Если используется DNS-адрес, он должен быть настроен в DNS (т.е. Вы должны настроить Ваш сервер DNS и убедиться, что Вы можете подключиться к AtomMind Server'у по его адресу в DNS).

`device_server_owner` - имя профиля пользователя 478, под которым зарегистрирован данный сервер устройств

`device_server_name` - имя профиля сервера устройств 2087

`page_path` и `page_name` должны указывать на страницу встроенного в сервер устройств веб-сервера.

Пример:

`https://ls.nuclear-sub.com/dev/admin/thermometer/gui/view_temperature.html`

Этот URL предоставляет доступ к странице `/gui/view_temperature.html` в сервер устройств `admin.thermometer` Веб-сервера, когда он подключен к AtomMind Server'у. Предполагается, что AtomMind Server доступен по адресу `ls.nuclear-sub.com`. Запрашиваемая страница будет передаваться по шифрованному соединению, даже если соединение между AtomMind Server'ом и сервером устройств нешифрованное.

Ошибка *HTTP Internal Server Error (Error 500)* показывается веб-приложением HTTP Proxy в следующих случаях:

- Пользователь указал несуществующий URL
- Сервер Устройств с таким URL не существует
- Сервер Устройств отключен
- Сервер Устройств не работает с драйвером HTTP Proxy

Подробнее см. Сервис HTTP Proxy 2081.

Информация о драйвере

Идентификатор плагина 518 **драйвера:**	com.tibbo.linkserver.plugin.device.httpproxy

Глобальные настройки

Включить приложение HTTP Proxy. Включает/отключает доступ к HTTP Proxy.

Пользовательские настройки

Не определены.

Настройки уровня Сервера Устройств

Не определены.

19.5.7 Серверы внешних устройств

Серверы Внешних Устройств, или Аккаунты *Серверов внешних Устройств*, позволяют AtomMind Server'у получить доступ к аппаратным Серверам Устройств, которые не способны самостоятельно инициировать соединение с сервером. Эта возможность часто используется для настройки новых Серверов Устройств, которые ещё не настроены входить в AtomMind Server автоматически при включении. Аккаунты внешних Серверов Устройств также могут быть полезны при решении проблем подключения к AtomMind Server'у.

Разница между аппаратным Сервером Устройств, аккаунтом Сервера Устройств и аккаунтом внешнего Сервера Устройств объясняется здесь 2086.

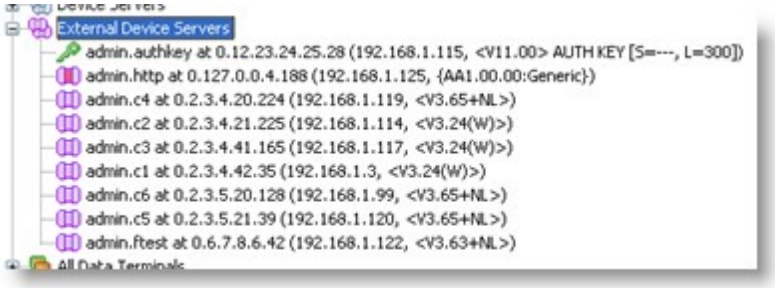
Аккаунты Серверов внешних Устройств используются для хранения параметров, необходимых AtomMind Server'у, чтобы подключиться к аппаратному Серверу Устройств по его IP- или MAC-адресу. Если AtomMind Server не может получить доступ к аппаратному Серверу Устройств (находящемуся в удалённой локальной сети за файерволом), то нет возможности получить к нему доступ через аккаунт Сервера внешнего Устройств. Есть несколько способов обойти это и заставить аппаратный Сервер Устройств подключаться к AtomMind Server'у при загрузке.:

- Подключите его на время в сегмент локальной сети, к которому AtomMind Server имеет доступ и подключитесь к нему как к Серверу Устройств. Как только Сервер Устройств будет правильно настроен и будет способен подключаться к AtomMind Server'у при загрузке, Вы можете вернуть его в его исходную сеть и инициировать подключение к AtomMind Server'у.
- Временно перенастройте файервол, маршрутизатор или другое сетевое оборудование, позволив AtomMind Server'у получить доступ к Серверу Устройств. Как только аппаратура Сервера Устройств будет настроена и будет готово к инициализации соединения с AtomMind Server'ом, настройки маршрутизатора можно вернуть в исходное состояние.
- Используя программу *DS Manager* на ПК в том же сегменте сети, в котором находится аппаратный Сервер Устройств (или из которой программ сможет получить доступ). Программа DS Manager входит в программный комплект DS Toolkit компании Tibbo.

Основная идея аккаунтов Серверов Внешних Устройств в том, чтобы использовать их только для доступа и настройки новых Серверов Устройств. Эта возможность не должна использоваться для работы с уже настроенным оборудованием.

Администрирование Серверов внешних устройств

Для администрирования Серверов Устройств используются два контекста: Первый - общий контекст Сервера внешних устройств 2115 для действий, связанных со всеми Серверами внешних устройств. Второй - контекст Сервер внешнего устройства 2112, соответствующий единственному аккаунту Сервера Устройства.



Обнаружение Серверов Устройств

Обнаружение Серверов Устройств - это процесс поиска всех аппаратных Серверов Устройств в *сегменте локальной сети* и автоматическое создание аккаунтов Внешних Серверов Устройств для них.

Сегмент локальной сети означает часть сети, в организованную при помощи свитчей (без маршрутизаторов, мостов, файерволов и пр.) между ПК и всеми устройствами в этом сегменте.

Аппаратные Серверы Устройств, находящиеся за маршрутизатором (т.е. у Вас есть ПК с запущенным на нем AtomMind Server'ом, которому чтобы отправить свои пакеты к Серверу Устройств требуется преодолеть маршрутизатор) не могут быть обнаружены автоматически AtomMind Server'ом, поскольку широковещательные UDP-датаграммы, предназначенные для обнаружения устройств, не могут быть переданы через маршрутизатор. Однако иногда бывает возможно 2098 использовать аккаунт Сервера внешнего устройств для подключения подобных Серверов Устройств, если они доступны по IP-адресу.

Каждый обнаруженный Сервер Устройств уникально идентифицируется его MAC-адресом, который отличается для каждого произведённого Сервера Устройств. Обнаружение определит все локальные Серверы Устройств, даже если некоторые из них обладают совпадающими или неправильными IP-адресами. Верно настроенный IP-адрес не требуется, чтобы AtomMind Server был способен получить доступ к Серверу Устройств в режиме обнаружения.

В процессе обнаружения AtomMind Server создает новый контекст Сервера вешнего устройства 2112 (аккаунт) для каждого найденного Сервера Устройств. Все контексты Серверов Устройств, созданные предыдущим процессом обнаружения, удаляются перед созданием новых контекстов. Нет способа создать аккаунт Сервера Устройств, кроме запуска процесса автоматического обнаружения.

Если автоматическое обнаружение отключено и не было запущено вручную, контекст Серверов Устройств 2115 не содержит аккаунтов Серверов Устройств.

АВТООБНАРУЖЕНИЕ СЕРВЕРОВ УСТРОЙСТВ

AtomMind Server может обнаруживать Сервера Устройств автоматически. Автообнаружение выполняется планируемой задачей Обнаружить и подключить серверы внешних устройств 2101. Возможно отключить автообнаружение, отключив эту задачу или поменять расписание этой задачи редактированием ее триггеров. Если опция **пытаться автоматически подключить серверы устройств к AtomMind Server'у** в этой задаче включена, AtomMind Server пытается автоматически подключить 2098 каждый обнаруженный сервер устройств.

Подключение серверов внешних устройств к AtomMind Server'у

Подключение Серверов Устройств к AtomMind Server'у - это процесс перенастройки аппаратного Сервера Устройств на немедленное подключение к AtomMind Server'у после загрузки. Есть несколько способов это сделать:

1. Подключение уже обнаруженного 2096 Сервера Устройств через его аккаунт 2112.
2. Автоподключение всех обнаруженных Серверов Устройств к AtomMind Server'у без взаимодействия с пользователем.
3. Ручная настройка обнаруженных Серверов Устройств, которые не смогли успешно подключиться двумя предыдущими функциями.
4. Ручное подключение Сервера Устройств, который не был обнаружен процессом обнаружения, но к которому может быть получен доступ с AtomMind Server'а.

ПОДКЛЮЧЕНИЕ РАНЕЕ ОБНАРУЖЕННОГО СЕРВЕРА УСТРОЙСТВ К ATOMMIND SERVER'У

Эта процедура инициируется действием Подключить Сервера Устройств 2112 к AtomMind Server контекста Сервер внешних устройств 2112.

Пользователь должен указать 90 несколько параметров перед началом процесса подключения:

- **Тип подключения.** "Постоянное (Работа с AtomMind Server)" или "только динамический DNS (сразу отключиться после регистрации в DNS)". Подробнее см. Режимы Подключения 2098.
- **Пароль.** Требуется, если в настройках аппаратного Сервера Устройств определён пароль. Этот пароль будет использован AtomMind Server'ом для доступа к аппаратному Серверу Устройств.
- **Имя владельца.** Имя пользователя 478 AtomMind Server'а, который будет владеть (или уже владеет) аккаунтом Сервера Устройств 2087, которое будет использоваться для аутентификации на аппаратном Сервере Устройств в процессе его входа в AtomMind Server. Настройка "Имя владельца" аппаратного Сервера Устройств будет установлена в это значение.
- **Имя сервера устройств.** Имя аккаунта Сервера Устройств 2087. Настройка **Имя устройства** аппаратного Сервера Устройств будет установлена в это значение.
- **Выполнить принудительное подключение, даже если Сервер Устройств уже настроен для работы с AtomMind Server'ом или его прошивка несовместима.** По умолчанию процедура подключения прервётся с сообщением об ошибке, если Сервер Устройств уже настроен для подключения к AtomMind Server'у или имеет устаревшую прошивку (которая не полностью поддерживает автоматическое конфигурирование для AtomMind Server). Эта опция отключает проверку значений настроек Сервера Устройств и версии его прошивки. Принудительное подключение будет успешно завершено, но Сервер Устройств может не быть способен

подключиться к AtomMind Server'у или войти. Если Сервер Устройств не вошёл в AtomMind Server в течении нескольких секунд после окончания процедуры принудительного подключения, продолжайте настройку в ручном режиме 2099.

Последовательность подключения описана здесь 2098.

АВТОПОДКЛЮЧЕНИЕ ОБНАРУЖЕННЫХ СЕРВЕРОВ УСТРОЙСТВ К ATOMMIND SERVER'У

Если опция **Пытаться автоматически подключить сервер устройств к AtomMind Server'y** планируемой задачи Обнаружение серверов внешних устройств 2097 включена, AtomMind Server автоматически запускает процедуру Подключение обнаруженных серверов устройств 2097 к AtomMind Server для каждого найденного в процессе обнаружения 2096 Сервера Устройств. Эта опция отключена по умолчанию из соображений безопасности. Удобно на скорую руку подключать несколько Серверов Устройств, подключенных к сегменту локальной сети.

Автоподключение не получится, если:

- Сервер Устройств требует пароль для доступа к его настройкам, или
- Сервер Устройств уже настроен для подключения к AtomMind Server'у при загрузке, или
- Прошивка Сервера Устройств не поддерживает автоподключение.

ПОДКЛЮЧЕНИЕ СЕРВЕРА УСТРОЙСТВ, НЕ ОБНАРУЖЕННОГО, НО ДОСТУПНОГО ДЛЯ AtomMind Server'A

Эта процедура должна использоваться для подключения аппаратного Сервера Устройств, который не был обнаружен, но доступен AtomMind Server'у. Это возможно, например, если Сервера Устройств расположен в другом сегменте локальной сети, но его IP-адрес достижим с AtomMind Server'а.

Все требуемые параметры подключения должны быть указаны для этой процедуры вручную. Она инициируется действием Подключить Сервер Устройств к 2115 AtomMind Server'у 2115 контекста Сервера внешних устройств 2115.

Для подключения необнаруженного Сервера Устройств Вам необходимо указать следующие опции:

- **Тип подключения.** *Постоянный (Работать с* AtomMind Server*)* или *только Динамический DNS (немедленно отключиться после регистрации в DNS)*. Подробнее см. Режимы подключения 2098.
- **Метод доступа**. Широковещательный *(UDP)*, *Вне потока данных (UDP)*, *В потоке данных (TCP)* или *Telnet (TCP)*.
- **MAC-адрес**. Должен быть указан в случае широковещательного доступа.
- **IP-адрес**. Должен быть указан, если не используются широковещательные пакеты.
- **Номер порта**. Если указан, этот порт будет использован для связи с Сервером Устройств вместо порта по умолчанию для выбранного метода доступа.
- **Имя владельца**. См. описание выше.
- **Имя Сервера Устройств**. См. описание выше.
- **Принудительно подключиться, даже если Сервер Устройств уже настроен для работы с AtomMind Server или имеет несовместимую прошивку.** См. описание выше.

Последовательность подключения описана здесь 2098.

РЕЖИМЫ ПОДКЛЮЧЕНИЯ

Аппаратный Сервер Устройств может быть настроен для работы с AtomMind Server'ом в двух режимах:

- обычный
- только динамический DNS.

Обычный режим предполагает, что Сервер Устройств подключается к AtomMind Server'у во время загрузки и остается подключенным неограниченное количество времени. Все данные, посланные или отправленные с него обрабатываются драйвером устройств 518 , определённом в аккаунте Сервера Устройств 2087. В большинстве случаев Сервер Устройств должен подключаться к AtomMind Server'у в нормальном режиме.

Режим только динамического DNS заставляет Сервер Устройств отключаться от AtomMind Server'а сразу после соединения. Единственное, что выполняется сервером при подключении Сервера Устройств - регистрация в DNS. После отключения от AtomMind Server'а Сервер Устройств работает как обычный конвертер последовательного порта в ethernet. Он не взаимодействует с AtomMind до последующей перезагрузки. За более подробной информацией, как используется этот режим см. Сервис динамического DNS (dDNS) 2082.

ПОСЛЕДОВАТЕЛЬНОСТЬ ПОДКЛЮЧЕНИЯ

Процедура настройки аппаратного Сервера Устройств для подключения к AtomMind Server'у при запуске включает несколько шагов:

1. Если ни MAC-, ни IP-адрес не указаны в настройках соединения, процесс подключения завершится с ошибкой

2. Если метод доступа установлен в **В потоке данных (TCP)** или **Telnet**, AtomMind Server пытается установить исходящее соединение по TCP с Сервером Устройств

3. AtomMind Server пытается определить верное значение **IP-адреса назначения** аппаратного Сервера Устройств (т.е. куда подключаться Серверу Устройств). Если значение глобальной конфигурационной переменной AtomMind Server'a IP-адрес сервера 1791 определено, IP-адрес назначения устанавливается в это значение. Иначе AtomMind Serverпосылает специальную команду "скажи мне мой IP-адрес" к Серверу Устройств. Сервер Устройств анализирует пришедший пакет с командой и высылает в ответ IP-адрес, с которого команда была отправлена. Этим способом сервер "знает" IP-адрес, с которого он виден Серверу Устройств (т.е. адрес, по которому Сервер Устройств считает, что там находится AtomMind Server).

Если аппаратный Сервер Устройств переносится в другой сегмент сети после процедуры "подключение к AtomMind Server'y", может оказаться необходимым изменить его настройку IP-адреса назначения.

4. AtomMind Server пытается войти в Сервер Устройств и получить доступ к его конфигурации. При этом используется пароль, определённый в параметрах "Подключение к AtomMind Server'y".

5. Если опция **Принудительное подключение в случае, если Серер Устройств уже настроен или имеет несовместимую прошивку** отключена, проверяется версия прошивки Сервера Устройств - совместима ли она с AtomMind Server'ом. Если она слишком стара, процедура подключения прерывается.

6. Если **Принудительное подключение** отключено, AtomMind Server пытается определить, настроен ли уже Сервер Устройств для подключения к AtomMind Server'y при запуске и разрывает связь, если проверка успешна. Если эта опция включена, AtomMind Server перепишет настройки Сервера Устройств в любом случае, чтобы увеличить вероятность успеха подключения к AtomMind Server'y.

7. После этого AtomMind Server решает, какой пользователь 478 и аккаунт Сервера Устройств 2087 должен использоваться подключенным аппаратным Сервером Устройств. Если указаны параметры **Имя владельца** и **Имя устройства**, они используются как пользователь и имя аккаунта Сервера Устройств соответственно. Если нет, AtomMind Server читает настройки **Имя владельца** и **Имя устройства** с аппаратного Сервера Устройств и использует эти значения.

8. Если аккаунт пользователя, определённый на предыдущем шаге не существует или недоступен пользователю, инициирующему процедуру подключения, процедура завершится с ошибкой.

9. Теперь настройки Сервера Устройств изменены для подключения при загрузке. чтобы узнать, что за настройки изменились и каковы их новые значения, см. Ручная настройка 2099 AtomMind Server.

10. Если аккаунт Сервера Устройств 2087 с именем, определённом на шаге 7 существует, его настройки обновлены и позволяют подключиться к новому аппаратному Серверу Устройств Иначе автоматически создаётся новый аккаунт Сервера Устройств.

11. Аппаратный Сервер Устройств перезагружается.

После этих шагов Сервер Устройств должен подключиться и войти в AtomMind Server в течении нескольких секунд. Если этого не произошло, проверьте статусные LED'ы 2101 Сервера Устройств чтобы выяснить, что с ним происходит. Вы также можете использовать ручную настройку (см. ниже) для разрешения проблем с подключением.

РУЧНАЯ НАСТРОЙКА СЕРВЕРА УСТРОЙСТВ ДЛЯ ПОДКЛЮЧЕНИЯ К ATOMMIND SERVER'У

Эти настройки аппаратного Сервера Устройств должны быть правильно установлены для подключения и входа на AtomMind Server. Эти настройки доступны через действие Настроить Сервер Устройств 2112 любого контекста Внешний Сервер Устройств 2112.

Настройка	**Значение**
Имя владельца	Имя пользователя 478, владеющего аккаунтом Сервера Устройств 2087
Имя устройства	Имя аккаунта Сервера Устройств
MAC-адрес	Дожен быть установлен в любое допустимое значение MAC-адреса, уникального в данном сегменте локальной сети. Значение по умолчанию подходит в большинстве случаев.
IP-адрес	Должен быть верно настроен для работы в данном сегменте сети.
Регистрация на сервере dDNS	Отключена
Авторегистрация на AtomMind Server'e	Включена
Режим PPPoE	Отключен
IP-адрес маршрутизатора	Должен быть верно настроен для данного сегмента сети.

Маска подсети	Дожна быть верной для данного сегмента сети.
Время ожидания соединения	Отключено (0 минут)
Транспортный протокол	TCP
Вход в Link Service	Включено
Режим маршрутизации	Только клиент
Режим подключения	Немедленно (при включении)
IP-адрес назначения	IP-адрес AtomMind Server'a
Порт назначения	Должен совпадать со значением глобальной конфигурации AtomMind Server'a Номер порта для ожидания входящих соединений от серверов устройств.

Когда аппаратный Сервер Устройств настроен на работу в режиме 2098 "только динамический DNS", некоторые из этих настроек должны быть установлены по-другому:

Вход в Link Service: отключен

Регистрация в dDNS: включена

IP-адрес сервера dDNS: IP-адрес AtomMind Server'a

порт dDNS-сервера: Должен совпадать со значением настройки глобальной конфигурации AtomMind Server'a Номер порта для ожидания подключений Серверов Устройств

Если аппаратный Сервер Устройств не подключается к AtomMind Server'у в течении нескольких секунд после перезагрузки, см. информацию по разрешению проблем здесь 2101.

Другие операции

ИДЕНТИФИКАЦИЯ

Идентификация позволяет визуально идентифицировать Сервер внешнего устройства по его аккаунту. Это действие заставляет Сервер Устройств проиграть последовательность скоростного моргания красным и зелёным статусными светодиодами. Этим способом можно быстро соотнести аккаунт и связанное с ним устройство.

ПЕРЕЗАГРУЗКА

Эта операция перезагружает аппаратный Сервер Устройств. Перезагрузка может требоваться после изменения настроек Сервера Устройств.

ИНИЦИАЛИЗАЦИЯ

Эта операция сбрасывает настройки аппаратного Сервера Устройств в настройки по умолчанию. Она требует подтверждения и не может быть отменена.

Значения по умолчанию могут препятствовать обнаружению Сервера Устройств AtomMind Server'ом. Используете эту возможность осторожно.

Иконки Серверов Внешних Устройств

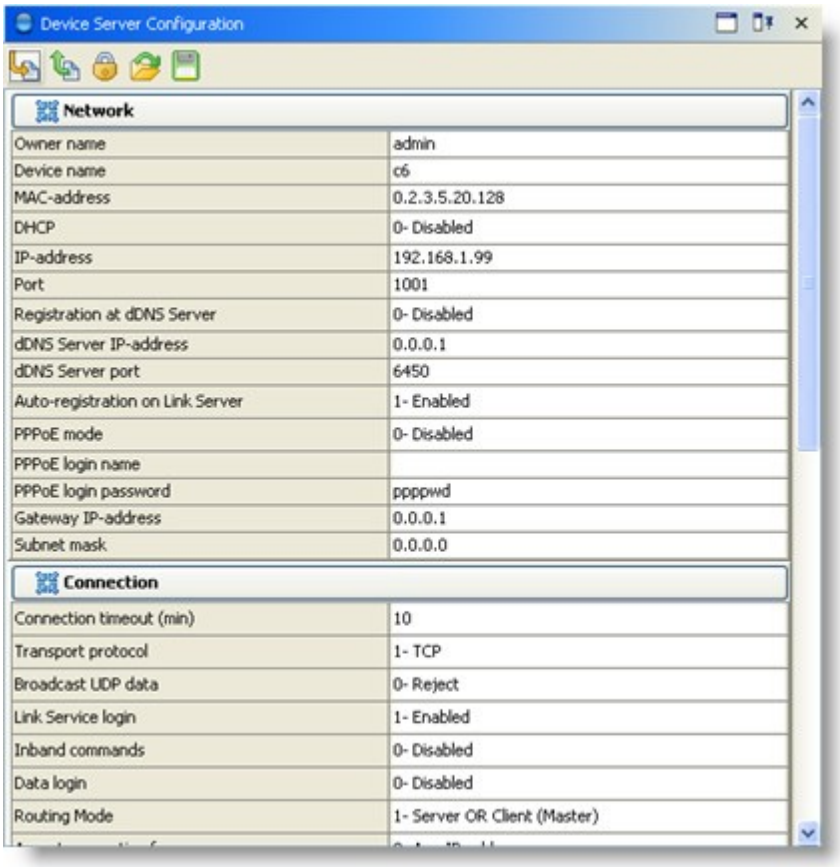
Сервер Внешних Устройств представляется разными иконками в пользовательских интерфейсах AtomMind Server'a:

	Обычный аппаратный Сервер Устройств с универсальной прошивкой.
	Сервер Устройств, реализованный в виде логической части Агентского 684 приложения на ТВЭЛ-BASIC, запущенный в программируемом режиме.

Конфигурирование Серверов Внешних Устройств

AtomMind Server предоставляет способ изменения внутренних настроек аппаратных Серверов Устройств. Эта функция сначала была реализована в программе DS Manager компании Tibbo. Настройки Сервера Устройств доступны через действие Настроить Сервер Устройств 2112 любого контекста Сервер внешних устройств 2112.

Вот пример диалога настроек Сервера Устройств (в клиенте):



19.5.8 Инструменты

Есть несколько инструментов, помогающих в управлении Серверами Устройств.

Запросы 829 к серверам устройств:

- **Все Сервера Устройств**. Этот запрос позволяет увидеть все сервера устройств и отредактировать настройки их профилей 2087 в одной таблице.
- **Команда идентифицироваться всем внешним Серверам Устройств**. Позволяет определить, какие из серверов устройств в настоящий момент доступны AtomMind Server'у, отдав всем достижимым через AtomMind Server устройствам команду идентифицировать. Обратите внимание, что обнаружение Серверов Устройств должно быть выполнено перед запуском этого запроса чтобы позволить AtomMind Server'у обнаружить Сервера Устройств в сегменте локальной сети. Подробнее об этом читайте в главе Внешние Сервера Устройств 2096.
- **Статистика трафика Серверов Устройств**. Показывает, как много данных было ппередано в и из каждого Сервера Устройств.

Задачи по расписанию 823 для серверов устройств:

- **Обнаружение и Подключение Внешних серверов устройств**. Это задание запускает периодическое Обнаружение Внешних Устройств 2096 и опционально автоподключение 2098. Это может быть отключено если нет доступных серверов устройств 2087 в этой инсталляции AtomMind.

Фильтры Событий 762 для серверов устройств:

- **События серверов устройств**. Этот фильтр отображает события, относящиеся к Серверам Устройств такие как попытки входа не имеющих профилей на AtomMind Server'е серверов устройств на AtomMind Server, регистрация новых профилей серверов устройств, проблемы связи, Успешные входы и пр.

19.5.9 Устранение проблем с подключением Серверов Устрой

Ниже перечислены варианты LED-сигналов Серверов Устройств для диагностики неполадок при их подключении:

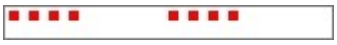
Не получен IP-адрес. Тут нет ничего общего с dDNS. Эта сигнализация означает, что Ваш Сервер Устройств настроен получать IP-адрес с DHCP-сервера и по какой-то причине этого не произошло. Сервер Устройств

попытается зарегистрироваться на AtomMind Server'е как только получит свой IP-адрес.

Отправка ARP-пакетов. Вы указали неверный IP-адрес шлюза или маску сети или Ваш маршрутизатор неверно настроен.

Подключение по TCP не было установлено. Вы указали неверный IP-адрес назначения или неверный IP-адрес сервера dDNS или Ваш роутер не настроен правильно..

TCP-соединение сброшено удаленным хостом. Вы указали неверный IP-адрес или порт назначения, неверный IP-адрес или порт сервера dDNS.



Вход не удался. AtomMind Server отказал в попытке входа с Вашего Сервера Устройств. Убедитесь, что Имя Владельца, Имя Сервера Устройств и Пароль Сервера Устройств соответствуют тем, что указаны в соответствующем профиле Сервера Устройств на AtomMind Server'e.

См. также: Эмулятор передачи сигнала LED

19.5.10 Файлы настроек

Некоторые Сервера Устройств не способны предоставить информацию AtomMind Server'у о своих внутренних настройках. Чтобы сделать возможной конфигурацию таких Серверов Устройств , AtomMind Server использует *Файлы Настроек* (SDF, *Setting Definition Files)*. Каждый SDF-Файл содержит список настроек, поддерживаемый Сервером Устройств с определённой версией прошивки. Файлы SDF входят в состав дистрибутива AtomMind Server'a и устанавливаются в поддиректорию `/sdf` установочного каталога AtomMind Server'a. Все файлы SDF загружаются в оперативную память во время запуска. Если в некоторых из файлов сделаны изменения или добавлен новый файл, необходимо перезапустить сервер для их активации.

Когда пользователь обновляет прошивку для старого Сервера Устройств, ему необходимо получить новый SDF-файл от ТВЭЛ и положить его в поддиректорию `/sdf`. Это позволит иметь доступ к настройкам для новой версии прошивки Сервера Устройств.

Обычный пользователь AtomMind Server'a (или даже администратор) никогда не будет иметь дела с содержимым SDF-файла. Считайте эту тему "углублённой", служащей расширению Вашего понимания внутренностей AtomMind Server'a.

19.5.11 Контексты

Этот раздел описывает контексты 41 управления серверами устройств.

19.5.11.1 Сервер Устройств

Этот контекст 41 используется для доступа и управления одним Сервером Устройств.

Уникальные действия [? 1450]

РЕДАКТИРОВАТЬ СВОЙСТВА СЕРВЕРА УСТРОЙСТВ (Действие по умолчанию 88)

Это действие используется для редактирования свойств 2089 профиля Сервера Устройств.

Тип действия: Настройка 105

Имя действия: editProperties

Иконка действия: 

НАСТРОИТЬ СЕРВЕР УСТРОЙСТВ

Это действие используется для редактирования настроек аппаратного Сервера Устройств. Оно отличается от стандартного действия Настроить 105 тем, что Сервер Устройств будет перезагружен после сохранения новых настроек.

Имя действия: configure

Иконка действия: нет иконки

НАСТРОИТЬ ДРАЙВЕР УСТРОЙСТВА

Это действие используется для редактирования свойств уровня Сервера Устройств[521] для Драйвера устройств[518], связанного с профилем Сервера Устройств. Тип драйвера установлен в основных свойствах[2089] профиля Сервера Устройств. За информацией о доступных свойствах см. описание драйвера.

Тип действия: Настройка[105]

Имя действия: configureDevicePlugin

Иконка действия: 

ПРОСМОТРЕТЬ СПИСОК СЕРВЕРОВ УСТРОЙСТВ

Показывает список Серверов Устройств

Тип действия: Вызов функции[103]

Имя действия: list

ПРОСМОТРЕТЬ СТАТУС СЕРВЕРА УСТРОЙСТВ

Это действие показывает[91] состояние[2090] профиля Сервера Устройств.

Тип действия: Настройка[105] (режим только для чтения)

Имя действия: status

Иконка действия: 

ИДЕНТИФИЦИРОВАТЬ

Это действие заставляет идентифицировать[2100] аппаратный Сервер Устройств.

Тип действия: Вызов функции[103]

Имя действия: buzz

ПЕРЕЗАГРУЗИТЬ

Вызов этой функции[103] перезагружает[2100] аппаратный Сервер Устройств.

Тип действия: Вызов функции[103]

Имя действия: reboot

Не интерактивный режим[99]: Поддерживается

ПРОСМОТРЕТЬ ПОТОК ДАННЫХ

Это действие используется для отслеживания данных, передаваемых между аппаратным Сервером Устройств и AtomMind Server'ом. Подробности здесь[2091].

Имя действия: dataFlow

Не интерактивный режим[99]: Не поддерживается

Права: Доступно на уровне[486] доступа *Обозреватель*.

Общие действия [?[1450]]

Удалить[106], Копировать[110], Редактировать права доступа[106], Показать журнал событий[109]

Состояния контекста и иконки

 Сервер Устройств отключен

 Сервер Устройств включен (подключен к AtomMind Server'у)

 Профиль Сервера Устройств заблокирован, попытки входа будут пресекаться

Некоторые драйверы устройств 518 могут влиять на состояние и иконку контекста. Например, драйвер Прозрачной маршрутизации данных (Link Service) 2093 устанавливает иконку контекста в  и статус в "Клиент подключен", когда какое-нибудь стороннее ПО подключено к AtomMind Server'у по специально выделенному и указанному в настройках драйвера для профиль Сервера Устройств порту и обменивается данными с текущим Сервером Устройств.

Другими словами, Вы видите эту иконку когда AtomMind Server используется как прозрачная "труба" между Сервером Устройств и другим ПО.

Дополнительная информация

Информация о контексте

Тип контекста 43: deviceServer

Имя контекста 42: предоставляется пользователем

Описание контекста 43: предоставляется пользователем

Путь контекста 42: users.USER_NAME.deviceservers.DEVICE_SERVER_ACCOUNT_NAME

Маска контекста 44: users.*.deviceservers.*

Права доступа к контексту 44: Наблюдатель

Публичные переменные (Свойства) [? 61]

ИНФОРМАЦИЯ О СЕРВЕРЕ УСТРОЙСТВ

Возвращает свойства 2089 профиля Сервера Устройств.

Имя переменной: deviceServerInfo

Число записей: 1

Права: Чтение на уровне 486 *Наблюдатель*

Формат 49 записи:

Имя	Тип	Описание
owner	строка	
name	строка	
password	строка	
description	строка	
registerindns	логическое значение	
blocked	логическое значение	
inbandpassthrough	логическое значение	

timezone	строка	
deviceplugin	строка	

СТАТУС СЕРВЕРА УСТРОЙСТВ

Возвращает состояние 2090 профиля Сервера Устройств.

Имя переменной: status

Число записей: 1

Права: Чтение на уровне 486 *Наблюдатель*

Формат 49 записей:

Имя поля	Тип поля	Заметки
name	строка	
online	логическое значение	
pluginStatus	строка	
dsToServer	длинное	
serverToDs	длинное	
creationtime	дата	
updatetime	дата	
realip	строка	Нулевое
realport	целое	Нулевое
logintime	дата	Нулевое
internalip	строка	Нулевое

Публичные функции [? 70]

ОТПРАВКА КОМАНД

Эта функция отправляет несколько команд к аппаратному серверу устройств и возвращает его ответы.

Имя функции: commands

Права: Доступна на уровне 486 *Наблюдетель*

Число записей на входе: 1...бесконечно

Входной формат 49:

Имя	Тип	Описание

command	строка	Текст команды

Число записей на выходе: 1...бесконечно

Выходной формат 49:

Имя	Тип	Описание
successful	логическое значение	True, если команда успешно отправлена и на нее пришел ответ.
errortext	строка	Текст сообщения об ошибке если возникла какая-то ошибка ввода-вывода при отправлении команды.
reply	строка	Текст ответа, полученный от сервера устройств если на команду пришёл ответ.

ОТПРАВКА БЕЗОТВЕТНЫХ КОМАНД

Эта функция отправляет несколько команд к аппаратному серверу устройств, не требующих ответа.

Имя функции: nonRepliedCommands

Права: Доступна на уровне 486 *Наблюдатель*

Число входных записей: 1...бесконечно

Входной формат 49:

Имя	Тип	Описание
command	строка	Текст команды

Число записей на выходе: 1...бесконечно

Выходной формат 49:

Имя	Тип	Описание
successful	логическое значение	True, если команда была успешно отправлена. **Успешная отправка команды, не требующей ответа по UDP не означает, что она успешно получена и обработана сервером устройств.**
errortext	строка	Текст сообщения об ошибке, если какая-то ошибка ввода-вывода возникла при отправке команды.

ИДЕНТИФИЦИРОВАТЬ

Идентифицирует 209 сервер устройств.

Имя функции: buzz

Права: Доступна на уровне 486 *Наблюдатель*

Число записей на входе: 0

Входной формат 49: нет

Число записей на выходе:	0
Выходной формат:	нет

ПЕРЕЗАГРУЗИТЬ

Перезагружает сервер устройств.

Имя функции:	reboot
Права:	Доступна на уровне *Наблюдатель*
Число записей на входе:	0
Входной формат:	нет
Число записей на выходе:	0
Выходной формат:	нет

ОТКЛЮЧИТЬ

Заставляет сервер закрыть TCP-соединение с аппаратным сервером устройств. Он попытается подключиться к серверу устройств через несколько секунд вновь.

Имя функции:	disconnect
Права:	Доступна на уровне *Наблюдатель*
Число записей на входе:	0
Входной формат:	нет
Число записей на выходе:	0
Выходной формат:	нет

УДАЛИТЬ

Удаляет профиля сервера устройств.

Имя функции:	remove
Права:	Доступна на уровне *Наблюдатель*
Число записей на входе:	0
Входной формат:	нет
Число записей на выходе:	0
Выходной формат:	нет

Общие события [?]

Общие события: info (Информация), contextStatusChanged (Изменение статуса)

ПОДКЛЮЧЕНИЕ

Это событие возникает при успешном входе устройства на сервер.

Имя события:	connection

Права: Доступно на уровне 486 *Наблюдатель*

Число записей: 0

ОТКЛЮЧЕНИЕ

Это событие возникает при отключении устройства от сервера.

Имя события: disconnection

Права: Доступно на уровне 486 *Наблюдатель*

Число записей: 0

ИЗВЕЩЕНИЕ

Это событие генерируется при получении извещения (т.е. команды, которая не является ответом на какую-либо другую команду) с сервера устройств. Такие извещения обрабатываются AtomMind Server'ом только если отключена опция команды-нотификации разрешены клиенту 2089 в профиле Сервера Устройств.

Имя события inband

Права: Доступно на уровне 486 *Наблюдатель*

Число записей: 1

Формат 49 записи:

Имя поля	Тип поля	Заметки
text	строка	Текст полученного извещения.

19.5.11.2 Серверы Устройств

Этот контекст 41 является контейнером для всех профилей Серверов Устройств 2087 для конкретного пользователя.

Уникальные действия [? 1450]

НОВЫЙ СЕРВЕР УСТРОЙСТВ (действие по умолчанию 88)

Это действие используется для создания нового профиля Сервера Устройств. Оно спрашивает пользователя о некоторых основных параметрах создаваемого профиля:

- Имя
- Пароль
- Описание
- Регистрация Сервера Устройств в DNS
- Блокирован ли

См. описание этих параметров здесь 2089.

Тип действия: Вызов функции 103

Имя действия: add

Иконка действия: 

ПРОСМОТРЕТЬ СПИСОК СЕРВЕРОВ УСТРОЙСТВ

Это действие показывает 90 основные свойства всех профилей Серверов Устройств этого контейнера в одной таблице в режиме только для чтения.

Тип действия: Настройка 105 (режим только для чтения)

Имя действия: list

ПРОСМОТРЕТЬ СТАТУС СЕРВЕРА УСТРОЙСТВ

Это действие показывает 91 состояние 2090 всех профилей Серверов Устройств, зарегистрированных в в этом контейнере в единой таблице в режиме только для чтения.

Тип действия: Настройка 105 (режим только для чтения)

Имя действия: status

Иконка действия:

Общие действия [? 1450]

Репликация в дочерний 111, Редактировать права контекста 106, Отслеживать связанные события 109, различные Групповые действия 101 дочерних контекстов.

Состояния и иконки контекста

Этот контекст не имеет состояний 44. Он всегда представляется иконкой .

Дополнительная информация

Информация о контексте

Тип контекста 43: deviceServers

Имя контекста 42: deviceservers

Описание контекста 43: Серверы Устройств

Путь контекста 42: users.USER_NAME.deviceservers

Маска контекста 44: users.*.deviceservers

Права доступа к контексту 44: Наблюдатель

Общие переменные (Свойства) [? 61]

СОСТОНИЕ СЕРВЕРОВ УСТРОЙСТВ

Возвращает состояния всех серверов устройств текущего пользователя. Подробнее о состояниях серверов устройств смотрите здесь 2090.

Имя переменной: status

Число записей: 0...бесконечность

Права: Чтения для уровня 486 доступа *Наблюдатель*.

Формат 49 записей:

Имя поля	Тип поля	Заметки
name	Строка	
online	Логическое значение	

pluginStatus	Строка	
dsToServer	Длинное	
serverToDs	Длинное	
creationtime	Дата	
updatetime	Дата	
realip	Строка	Может быть null
realport	Целое	Может быть null
logintime	Дата	Может быть null
internalip	Строка	Может быть null

Общие функции [? 70]

ПОКАЗАТЬ СПИСОК СЕРВЕРОВ УСТРОЙСТВ

Эта функция возвращает информацию о всех профилях серверов устройств, доступных под данным пользователем. Свойства профилей описываются здесь 2089.

Имя функции: list

Права: Доступно с уровнем 486 прав Наблюдатель

Записей на входе: 0

Формат 49 на входе: Нет

Записей на выходе: 0...бесконечность

Формат 49 на выходе:

Имя	Тип	Описание
owner	строка	
name	строка	
password	строка	
description	строка	
registerindns	логическое значение	
blocked	логическое значение	
inbandpassthrough	логическое значение	

timezone	строрка	
deviceplugin	строка	

ПОКАЗАТЬ СПИСОК ВСЕХ СЕРВЕРОВ УСТРОЙСТВ

Эта функция возвращает информацию о всех профилях серверов устройств в системе, доступных на данном уровне пользовательских привилегий. Свойства профилей описаны здесь 2089.

Имя функции: listAll

Права: Доступно с уровнем 486 прав *Наблюдатель*

Записей на входе: 0

Формат 49 на входе: Нет

Записей на выходе: 0...бесконечность

Формат 49 на выходе:

Имя	Тип	Описание
owner	строка	
name	строка	
password	строка	
description	строка	
registerindns	логическое значение	
blocked	логическое значение	
inbandpassthrough	логическое значение	
timezone	строка	
deviceplugin	строка	

ЗАРЕГИСТРИРОВАТЬ НОВЫЙ СЕРВЕР УСТРОЙСТВ

Создает новый профиль сервера устройств. Свойства профиля описаны здесь 2089.

Имя функции: add

Права: Доступно с уровнем 486 прав *Наблюдатель*

Записей на входе: 1

Формат 49 на входе:

Имя	Тип	Описание
name	строка	

password	строка	
description	строка	
registerindns	логическое значение	
blocked	логическое значение	

Записей на выходе: 0

Формат 49 на выходе: Нет

Общие события [? 73]

Общие события: info (Информация) 77

19.5.11.3 Сервер внешних устройств

Этот контекст 41 позволяет получить доступ к одному серверу устройств 2096.

Уникальные действия [? 1450]

ПОДКЛЮЧИТЬ СЕРВЕР УСТРОЙСТВ К ATOMMIND SERVER'У (действие по умолчанию 88)

Это действие используется для подключения сервера устройств к AtomMind Server. Подробности здесь 2097.

Тип действия: Вызов функции 103

Имя действия: connect

НАСТРОИТЬ DEVICE SERVER

Это действие используется для редактирования настроек аппаратного Сервера Устройств. Оно имеет два отличия от обычного действия Настройка 105: оно спрашивает 90 пароль к Серверу Устройств в начале (если таковой имеется у Сервера Устройств) и перезагружает Сервер Устройств как только новые настройки сохранены.

Имя действия: configure

ИДЕНТИФИЦИРОВАТЬ

Это действие идентифицирует 2100 Сервер Устройств.

Тип действия: Вызов функции 103

Имя действия: buzz

ПЕРЕЗАГРУЗИТЬ

Это действие перезагружает 2100 Сервер Устройств.

Тип действия: Вызов функции 103

Имя действия: reboot

ИНИЦИАЛИЗАЦИЯ

Это действие инициализирует 2100 Сервер Устройств.

Тип действия: Вызов функции 103

Имя действия: initialize

Общие действия [? 1450]

Реплицировать 110, Редактировать права контекста 106, Отслеживать связанные события 109

Состояния контекста и иконки

 Сервер Устройств с ограниченными функциями

 Agent

За подробностями обратитесь в этот 2100 раздел.

Дополнительная информация

Информация о контексте

Тип контекста 43: externalDeviceServer

Имя контекста 42: генерируется автоматически

Описание контекста 43: генерируется автоматически

Путь контекста 42: external_device_servers.NAME_OF_THIS_CONTEXT

Маска контекста 44: external_device_servers.*

Права контекста 44: Наблюдатель

Общие переменные (Свойства) [? 61]

Этот контекст предоставляет одну пользовательскую переменную на каждую настройку аппаратного сервера устройств когда он подключен к серверу. Когда одна из этих переменных читается или записывается, AtomMind Server шлёт команды **Get Setting** или **Set Setting** к аппаратному серверу устройств.

Общие функции [? 70]

ПОДКЛЮЧИТЬ СЕРВЕР УСТРОЙСТВ К ATOMMIND SERVER'У

Заставляет AtomMind Server подключиться 2097 к Внешнему серверу устройств.

Имя функции: connect

Права: Доступен для уровня 486 доступа *Наблюдатель*

Записей на входе: 1

Формат 49 на входе:

Имя	Тип	Описание
dnsOnly	строка	
password	строка	
owner	строка	Может быть null
name	строка	Может быть null
force	логическое значение	

Записей на выходе: 0

Формат 49 **на выходе:** Нет

ПОПЫТАТЬСЯ ПОДКЛЮЧИТЬ СЕРВЕР УСТРОЙСТВ АВТОМАТИЧЕСКИ К ATOMMIND SERVER

Заставляет AtomMind Server попытаться автоматически подключить 2098 Внешний AtomMind Device Server.

Имя функции: autoConnect

Права: Доступен для уровня 486 доступа *Наблюдатель*

Записей на входе: 0

Формат 49 **на входе:** Нет

Записей на выходе: 0

Формат 49 **на выходе:** Нет

ИДЕНТИФИЦИРОВАТЬ

Идентифицирует 2100 Внешний сервер устройств.

Имя функции: buzz

Права: Доступен для уровня 486 доступа *Наблюдатель*

Записей на входе: 0

Формат 49 **на входе:** Нет

Записей на выходе: 0

Формат 49 **на выходе:** Нет

ИНИЦИАЛИЗАЦИЯ

Инициализирует 2100 Внешний сервер устройств.

Имя функции: initialize

Права: Доступен для уровня 486 доступа *Наблюдатель*

Записей на входе: 0

Формат 49 **на входе:** Нет

Записей на выходе: 0

Формат 49 **на выходе:** Нет

ПЕРЕЗАГРУЗКА

Перезагружает 2100 внешний сервер устройств.

Имя функции: reboot

Права: Доступен для уровня 486 доступа *Наблюдатель*

Записей на входе: 0

Формат 49 **на входе:** Нет

Записей на выходе: 0

Формат 49 **на выходе:** Нет

Общие события [?73]

Общие события: info (информация)77

19.5.11.4 Серверы внешних устройств

Этот контекст41 является контейнером для всех профилей Внешних Серверов Устройств2096, предоставляющим действия для управления Внешними Серверами Устройств.

Уникальные действия [?1450]

НАЙТИ СЕРВЕРЫ УСТРОЙСТВ (действие по умолчанию88)

Это действие запрашивает AtomMind Server инициировать процесс обнаружения Серверов Устройств2096.

Тип действия:	Вызов функции103
Имя действия:	discover

ПОДКЛЮЧИТЬ DEVICE SERVER К ATOMMIND SERVER

Это действие позволяет подключить Device Server, ненайденный в процессе обнаружения2096 к AtomMind Server. Подробности здесь2098.

Тип действия:	Вызов функции103
Имя действия:	connect

Общие действия [?1450]

Реплицировать клиенту111, Редактировать права контекста106, Отслеживать связанные события109, различные Групповые действия101 с дочерними контекстами.

Статус контекста и иконки

Этот контекст не имеет состояний44. Он всегда представлен иконкой  .

Дополнительная информация

Информация о контексте

Тип контекста43: externalDeviceServers

Имя контекста42: external_device_servers

Описание контекста43: Внешние серверы устройств

Путь контекста42: external_device_servers

Маска контекста44: external_device_servers

Права контекста44: Наблюдатель

Общие переменные (Свойства) [?61]

Этот контекст не имеет общих переменных (свойств).

Общие функции [?70]

ОБНАРУЖИТЬ ЛОКАЛЬНЫЕ СЕРВЕРЫ УСТРОЙСТВ С ПОМОШЬЮ ШИРОКОВЕЩАТЕЛЬНОГО ОПРОСА

Заставляет сервер обнаруживать2096 серверы устройств, доступные в локальной сети и возвращает список найденных серверов устройств.

Имя функции: discover

Права: Доступен для уровня 486 доступа *Наблюдатель*

Записей на входе: 0

Формат 49 на входе: Нет

Записей на выходе: 0...бесконечность

Формат 49 на выходе:

Имя	Тип	Описание
mac	строка	Может быть null
ip	строка	
port	целое	Может быть null
owner	строка	Может быть null
name	строка	Может быть null
accessMethod	целое	

ПОДКЛЮЧИТЬ DEVICE SERVER К ATOMMIND SERVER

Заставляет AtomMind Server подключить 2098 Внешний Device Server.

Имя функции: connect

Права: Доступен для уровня 486 доступа *Наблюдатель*

Записей на входе: 1

Формат 49 на входе:

Имя	Тип	Описание
dnsOnly	строка	
accessMethod	целое	
mac	строка	
ip	строка	
port	целое	
password	строка	
owner	строка	Может быть null
name	строка	Может быть null
force	логическое значение	

Записей на выходе: 1

Формат 49 **на выходе:**

Имя	Тип	Описание
owner	строка	
name	строка	

Общие события [? 73]

Общие события: info (информация) 77

20 Приложения

В приложении содержится дополнительная информация, которая может и не встречаться в самом руководстве, однако она расширяет понимание некоторых элементов, используемых в системе.

20.1 Спецификация AtomMind

Общая системная архитектура	Распределённая мульти-серверная, мульти-клиентская система.
Распределённая архитектура	• Множество серверов на одном и более логических уровнях в пределах одной установки • Независимые равноправные отношения между серверами • Множество отдельных клиентов в пределах одной установки • Поддержка множественных серверных соединений в отдельных клиентах • Множество веб-клиентов в пределах одной установки
Архитектура высокой доступности	• N-узловая отказоустойчивая кластеризация • Ведущие, первичные ведомые и множественные вторичные ведомые узлы отказоустойчивого кластера • Независимая кластеризация базы данных • Собственная технология кластеризации БД с копированием записи и чтением баланса загрузки • Поддержка кластеризации СУБД средствами кластера СУБД
Платформа выполнения	Java SE (версия 1.8.0 и выше)
Операционная система	Любая операционная система, поддерживающая Java SE. Доступны дистрибутивы для: • Windows (x32 и x64) • Linux/Unix (x32 и x64) • Mac OS (x64)
Хранилище данных	• База данных "Ключ-значение" • NoSQL база данных • Реляционная база данных • Кольцевая база данных • Графовая база данных • Файловое хранилище
Поддерживаемые движки базы данных	Любая база данных с поддержкой JDBC. Протестировано с: • Apache Derby (интегрирован в дистрибутив) • MySQL 5.0 и выше (интегрирован в избранные дистрибутивы) • Oracle 10 и выше • PostgreSQL 9 и выше • Microsoft SQL Server 2008 и выше
Основные технологии разработки	Hibernate/JDBC, JSP/JSF, Swing, Tomcat Application Server
Системные требования	См. в отдельных разделах: • Системные требования сервера 146 • Системные требования клиента 359

Схема базы данных	Динамическая, описана в отдельном разделе 706
Безопасность	• Интегрированная модель контроля доступа пользователей на основе ACL • Безопасная SSL связь сервер-сервер • Безопасная SSL связь клиент-сервер • Безопасность связи сервер-устройство зависит от протокола передачи данных устройства

20.2 Коммуникационный протокол AtomMind

Протокол AtomMind используется для взаимодействия между AtomMind Server 144, AtomMind агентом 684 и AtomMind Clientом 359. Связь осуществляется по одиночному TCP-соединению, которое опционально защищается SSL/TLS шифрованием.

На протяжении коммуникационной сессии обе стороны обмениваются *командами*. В этой главе инициатор команд называется *клиентом*, а другая сторона называется *сервером:*

- Во взаимодействии AtomMind Server и AtomMind Client, инициатором является AtomMind Client
- Во взаимодействии AtomMind Server и агента AtomMind, инициатором является <%LS%>

На данный момент существуют 2 версии протокола: 2 и 3. Все различия будут описаны далее.

Инкапсуляция команды

Команды инкапсулируются при помощи символов *<STX> (0x02)* and *<CR> (0x0D).* Команды извлекаются из потока данных, а все остальные данные отбрасываются. Другие аспекты зависят от версии протокола.

ВЕРСИЯ 2

Структура инкапсуляции комманды следующая:

```
<STX>command<CR>
```

Если есть два и более символов <STX> без <CR>, все данные, принадлежащие текущему потоку, отбрасываются на каждый полученный <STX>:

`<STX>aaa<STX>bbb<CR>` понимается как команда `bbb`.

ВЕРСИЯ 3

Структура инкапсуляции комманды следующая:

```
<STX><L1><L2><L3><L4><T>command<CR>
```

После символа <STX> следует 4-байтовая символическая структура <L1><L2><L3><L4>. Это командная *длина* в байтах.

Далее <T> - это командный *тип* байта. В этой версии протокола команда может быть двух видов: *Raw* (0) и *Compressed* (1). Raw обознает, что команда отправлена в качестве первичного char массива. Compressed обозначает команду, которая отправляется сжатой в качестве ZLIB байтного массива.

Затем команда проходит в первичном или сжатом состоянии, инкапсуляция закрывается, используя символ <CR>.

Пример инкапсуляции команды для версии протокола №3: <STX><0><0><0><7><0>R/123/A<CR>

Структура команды

Каждая команда состоит из нескольких частей, разделенных символами 0x17 (“Разделитель команд”). Символ разделителя команд в данном руководстве представлен символом / . Например:

<STX>aaa/bbb/ccc<CR>

содержит три части: aaa, bbb и ccc.

Зарезервировано три символа, которые не могут появляться внутри команды:

Код символа	Имя символа
0x02	STX
0x0D	CR
0x17	ETB

Для каждой команды характерна следующая структура:

<STX>command_code/command_parameters…<CR>

command_code - это отдельный символ, один из:

M	Команда “Message”
R	Команда “Reply”

command_parameters - это один или несколько параметров, относящихся к коду команды, разделённые символом /.

Команды

КОМАНДА "MESSAGE”

Структура этой команды следующая (с этого момента, первый символ <STX> и оконечный <CR> не указываются):

M/message_ID/message_code/message_parameters…

message_ID уникальный для текущей сессии.

message_code - это символ, указывающий на тип сообщения. Доступные коды сообщения:

S	сообщение “Start”
O	сообщение “Operation”
E	сообщение “Event”

message_parameters - это один или более параметров, характерных для кода сообщения, разделяемого символом /.

КОМАНДА "REPLY”

R/reply_ID/reply_code/reply_parameters…

reply_ID должен соответствовать ID сообщения2120, на которое отвечает.

Параметр reply_code - это признак, который указывает на тип ответа. Доступные коды ответа:

A	Успешно (далее может следовать необязательный параметр data_table, см кодирование таблиц данных2123)
D	Отклонено
E	Ошибка (далее может следовать один или два параметра: сообщение об ошибке и/ или кодированная строка подробного описания ошибки)
L	Учетная запись пользователя в настоящее время заблокирована из-за нарушения ограничений системы безопасности
M	Операция запрещена, так как сервер нахоится в режиме обслуживания172

Сообщения

СООБЩЕНИЕ "START"

M/message_ID/S/protocol_version

Формат данного сообщения в дальнейшем не изменяется. Его должен отправлять клиент, как самую первую команду во время обмена данными с сервером. Сервер возвращает ошибку, если какое-либо другое сообщение отправляется раньше, чем было отправлено сообщение **Start**.

protocol_version - это целое число, указывающее на версию протокола взаимодействия, поддерживаемого клиентом. Текущая версия **3**.

Сервер отвечает на команду **Start** командой **Reply**. Если версия протокола, заданная клиентом, поддерживается сервером, код ответа - **Success**, в ином случае он отвегается. См. описание команды Ответ2120.

СООБЩЕНИЕ "OPERATION"

M/message_ID/O/operation/context_name/entity[/data_table]

Команда Operation приказывает серверу выполнить операцию на заданном контексте.

Operation - это свойство, определяющее операцию, которая должна быть выполнена. context_name - это имя контекста, для которого выполняется запрашиваемая операция. entity - это параметр операции. Это может быть имя переменной, если операция - "Set Variable", или имя функции, если операция - "Call Function".

Список доступных кодов операций:

G	Получить переменную
S	Установить переменную
C	Вызвать функцию
L	Добавить слушателя события
R	Удалить слушателя события

data_table - это необязательный параметр, содержащий, характерные для операции данные. См. приложение кодирование таблиц данных2123 .

СООБЩЕНИЕ "EVENT"

M//E/context_name/event_name/event_level/event_ID/event_listener_ID/data_table

Это сообщение отправляется клиенту, когда событие , именуемое event_name, происходит в context_name и этот клиент был ранее добавлен как слушатель данного события. Оно не требует ответа от клиента. Если событие постоянно, поле event_ID содержит его уникальный ID. event_listener_ID - это целое число, которое указывает на объект слушателя на стороне клиента. data_table содержит данные, специфичные для события.

Примечание: Событие "event" имеет пустой параметр message_ID.

Операции

ОПЕРАЦИЯ “ПОЛУЧИТЬ ПЕРЕМЕННУЮ”

M/message_id/O/G/context _name/variable_name

Эта операция возвращает variable_name из context_name.

Если переменная обнаружена, и не возникает ошибка, ответ на эту команду будет содержать таблицу данных со значением запрашиваемой переменной:

R/reply_id/A/data_table

ОПЕРАЦИЯ “УСТАНОВИТЬ ПЕРЕМЕННУЮ”

M/message_id/O/S/context _name/variable_name/data_table

Эта операция устанавливает для variable_name , принадлежащему context_name, значение, содержащееся в data_table.

Если значение переменной было успешно изменено, сервер ответит:

R/reply_id/A

ОПЕРАЦИЯ “ВЫЗВАТЬ ФУНКЦИЮ”

M/message_id/O/C/context _name/function_name/data_table

Эта операция вызывает function_name из context_name с data_table в качестве входного параметра.

Если не возникает ошибок, ответ содержит таблицу данных, возвращаемую функцией:

R/reply_id/A/data_table

ОПЕРАЦИЯ “ДОБАВИТЬ СЛУШАТЕЛЯ СОБЫТИЯ”

M/message_id/O/L/context _name/event_name/event_listener_ID

Эта операция регистрирует слушателя с заданным целым числом - идентификатором event_listener_ID для event_name в context_name. Когда происходит событие, клиент получает сообщение “Event” (E) с кодом event_listener_ID.

Опциональный параметр filter_text позволяет заранее отфильтровать события, основанные не представленном выражении 112 фильтра. Это выражение будет расситываться для каждого подходящего события, и если результатом будет false, событие не отправится клиенту.

Среда вычисления 114 выражения фильтра:	
Контекст по умолчанию 119	Отсутствует.
Таблица данных по умолчанию 120	Таблица данных, содержащая специфичные для события данные 75.
Строка по умолчанию 119	0
Переменные среды 123	Только стандартные 123 переменные.

При успешном добавлении слушателя события, сервер отвечает:

```
R/reply_id/A
```

ОПЕРАЦИЯ "УДАЛИТЬ СЛУШАТЕЛЯ СОБЫТИЯ"

```
M/message_id/O/R/context _name/event_name/event_listener_ID
```

Эта операция удаляет слушателя с заданным `event_listener_ID` , принадлежащим `event_name,` в `context_name`.

При успешном удалении слушателя события, сервер отвечает:

```
R/reply_id/A
```

Кодирование таблиц данных

Все значения данных, полученных или отправленных в рамках данного протокола, представлены в таблицах данных 49. Каждая таблица данных зашифрована в строку, которую следует вставить в команду. См. кодирование таблиц данных 2123.

20.3 Кодирование таблиц данных

Любая таблица данных 49 может кодироваться в или декодироваться из:

- Строки Юникода
- Байтового массива

Кодирование и декодирование строк часто используется для программного манипулирования таблицами данных, например, через Java SDK 1339 или .NET API 1397. Строковое представление табличных форматов также часто используется для компактного и удобного для чтения отображения данных.

Кодирование и декодирование байтовых массивов используется для переноса таблиц данных по сети при помощи Коммуникационного протокола AtomMind 2119 или хранения их в базе данных 692 AtomMind Server.

Кодирование байтовых массивов

Перекодирование таблицы данных в байтовый массив проходит в два этапа:

- Во-первых, таблица данных кодируется в строку Юникода
- Во-вторых, строка результата кодируется в байтовый массив при помощи кодировки UTF-8. См. документацию по UTF-8 (например, Статья в Wikipedia по UTF-8) для более подробного ознакомления.

Концепция кодирования строк

В целом, таблица данных и её различные части закодированы в строку как ряд *элементов* следующего формата:

```
<[element_name=]element_value>
```

Имя и значение могут включать любой символ, за исключением тех, которые использует протокол взаимодействия AtomMind 2120 или тех, которые используются для кодирования самого элемента 2130. Значение элемента может быть закодированным списком вложенных элементов.

Кодирование таблицы

Здесь представлен формат для закодированной таблицы данных:

```
<F=record_format>[<I=>][<R=record>][<R=record>]…
```

Имя элемента	Значение элемента
F	Дескриптор формата 49 таблицы, определяющий имена и типы всех колонок в таблице данных или других свойств таблицы. О его кодировании говорится здесь 2124.
D	ID формата таблицы. Более поробно см. раздел кэширование формата 1397.
I	Элемент для проверки. Если данный элемент найден в таблице данных, вся таблица считается неверной. Когда AtomMind Server запрашивает значение переменной от агента, агент сначала подтверждает запрос, а затем начинает опрос записей с аппаратного устройства. В это время сервер "думает", что он получает полную и правильную таблицу.

	Если в этот момент возникает внезапный разрыв связи между агентом и устройством, и по какой-то причине агент не получает данные, которые он должен получать от аппаратного устройства, агент вставляет проверочный элемент в таблицу данных, отправляемую серверу. Этот элемент сообщает серверу, что он не может полностью получить все эти данные. После получения этого элемента сервер "знает", что операция не может быть полностью завершена.
R	Запись таблицы данных. Записи закодированы одна за другой. Формат закодированных записей описан ниже.
T	Временная метка таблицы данных. Обычно указывает на время создания/получения самой таблицы данных или выборки данных, которую она представляет. Временная метка кодируется в строку как количество миллисекунд с начала отсчета времени (1 января 1970 г.).
Q	Качество таблицы. Оно объясняет, насколько надежна выборка данных, представленная таблицей данных. Качество является 32-битным подписанным целым значением, закодированным в строку.

Пример: `<F=<<IP><S><F=C>><M=1><X=1>><R=<192.168.1.88>>`

Этот пример показывает, что формат закодированной таблицы данных определяет одно поле строки, именуемое "IP", и что он содержит одну запись со значением "192.168.1.88". См. дополнительную информацию о записи и кодировке формата ниже.

Кодировка формата таблицы

`record_format` включён в каждую таблицу данных, даже если она пустая. Кодировка выглядит следующим образом:

```
<field_format><field_format>…[<F=flags>][<V=table_validators>][<R=record_validators>]
[M=min_records][X=max_records][<B=bindings>][<N=naming_expression>]
```

Элементы без идентификаторов (как показано в примере ранее `<field_format>`) считаются дескрипторами закодированных форматов 2125 полей таблиц. Дескрипторы формата полей закодированы один за другим, начиная с самого первого поля.

Имя элемента	Значение элемента
F	Отсутствие или комбинация следующих флагов: "R" (Перемещаемый (Reorderable)) - указывает, что пользователи AtomMind могут перемещать во время редактирования ряды данной таблицы "U" (Неизменяемый (Unresizable)) - указывает, что пользователи не могут добавлять/удалять ряды во время редактирования таблицы.
V	Валидаторы таблиц 2128, которые позволяют выполнять комплексное подтверждение всей таблицы.
R	Валидаторы записей 2128, которые используются для подтверждения каждой записи.
B	Привязки 741 таблиц.
M	Минимальное разрешённое количество записей в таблице.
X	Максимальное разрешённое количество записей в таблице.
N	Выражение имени таблицы.

Пример: `<<date><D>><M=1><X=1>`

Этот формат, описывающий таблицу с одной ячейкой (одно поле, минимальное и максимальное количество записей тоже одно). Единственное имя таблицы - "date", и его тип - Дата. См дополнительную информацию о кодировании формата ниже.

Пример: `<<id><S><D=Card ID><V=<L=10 10>>><<name><S><D=Cardholder Name>><M=0><X=255>`

Этот формат описывает таблицу с двумя полями, которая может содержать от 0 до 255 записей. Первое поле строки называется "id", а его описание - "Card ID". У него есть валидатор, который ограничивает длину значения ровно до 10 знаков. Второе поле также представляет собой тип строки и называется "name".

Кодирование формата поля

field_format - это строка, описывающая одно поле в таблице данных. Оно форматируется следующим образом:

```
<name><type>[<F=flags>][<A=default>][<D=description>][<H=help>][<S=selection_values>]
[<V=validators>][<E=editor>]
```

У первых двух элементов нет имен. Первый элемент - имя поля, а второй содержит код типа поля (см таблицу ниже).

Имя элемента	Значение элемента
F	Отсутствие или комбинация следующих флагов: "N" (Допускающий пустое значение (Nullable)) - указывает, что колонка может не содержать значение "O" (Необязательный (Optional)) - указывает, что колонка является необязательной "E" (Расширяемые значения выбора (Extendable selection values))– указывает, что поле может содержать значения, не перечисленные в selection_values "R" (Только для чтения (Read only)) – указывает, что значение поля доступно только для чтения "C" (Не реплицируется (Not replicated)) - указывает, что значение поля не реплицируется во время операции Копия таблицы данных. "H" (Скрыт (Hidden)) – указывает, что колонка не должна быть видна во время операций с таблицей данных "K" (Ключевое поле (Key field)) - указывает, что колонка является -- *ключевым полем*. Ключевые поля используются во время операции умное копирование таблицы данных. Кроме того используется *валидатор ключевых полей*, цель которого удостовериться, что таблица не содержит записей с равными комбинациями всех ключевых полей.
A	Значение поля, по умолчанию закодированное в строку (см раздел кодирование значения 2125)
D	Описание поля
H	Подсказка поля (подробное описание).
S	Список значений выбора для поля. См. правила кодирования здесь 2128.
V	Список валидаторов полей. См. правила кодирования здесь 2128.
E	Код редактора/отрисовщика. Этот элемент активирует пользовательское визуальное представление значения поля. Поддерживаемые редакторы и отрисовщики представлены здесь 52 .
O	Специфичные опции редактора. Опции, поддерживаемые каждым типом редактора/отрисовщика представлены здесь 52. Если опции редактора не определены, этот элемент должен быть опущен в определении формата таблицы.
I	Строковое ID иконки поля.
G	Группа полей.

Пример: <value><I>

Это самый простой из возможных дескрипторов формата полей, который определяет поле целых чисел, называемых "значениями".

Пример: <period><L><A=30000><D=Check Period><V=<L=100 1000000>><E=period><O=0 4>

Этот дескриптор формата определяет длинное поле (64-битное целое число) со значением по умолчанию, равным 30000. Описание поля - "Период проверки". У него есть валидатор ограничений, который ограничивает значения, диапазон которых составляет от 100 до 1000000. Тип редактора/отрисовщика инструктирует систему использовать редактора Периода времени для редактирования значения поля.

ТИПЫ ПОЛЕЙ И КОДИРОВАНИЕ ЗНАЧЕНИЙ

Код типа	Тип	Передача 2129	Комментарии и правила кодирования значения
S	String	Да	Вставлено как есть
I	Integer		Преобразовано в строку, например `123` или `-123`.
L	Long		Преобразовано в строку, например `123` или `-123`.
B	Boolean		**TRUE** закодировано как строка "**1**", а **FALSE** как строка "**0**"
F	Float		Преобразовано в строку в соответствии с ниже приведёнными правилами. Все упомянутые символы являются символами ASCII. • Если аргумент - NaN, результатом будет строка `NaN`. • В ином случае результатом будет строка, представляющая знак и магнитуду (абсолютное значение) аргумента. Если знак отрицательный, первым символом результата является `-`; если знак положительный, в результате не появляется никакого знака. Что касается магнитуды **m**: • Если **m** - это бесконечность, она представлена символом `Infinity`; таким образом, положительная бесконечность имеет результат `Infinity`, а отрицательная бесконечность имеет результат `-Infinity`. • Если **m** нулевая, она представлена символами `0.0`; таким образом, отрицательный ноль имеет результатом `-0.0`, а положительный ноль имеет результатом `0.0`. • Если **m** больше или равно 10^{-3}, но меньше 10^{7}, она представлена как целая часть **m**, в десятичной форме без начальных нулей, после которых стоит `.`, затем одна или несколько цифр, представляющих дробную часть m. • Если **m** меньше 10^{-3} или больше или равна 10^{7}, тогда она представлена так называемым "компьютеризированным экспоненциальным представлением". Пусть **n** будет уникальным целым числом, таким как 10^{n} <= **m** < 10^{n+1}; тогда пусть **a** будет математическим точным частным **m** и 10^{n}, так чтобы 1 <= **a** < 10. Тогда магнитуда представлена как целая часть **a**, как одно десятичное число, после которого стоит `.` и десятичные числа, представляющие дробную часть **a**, после которой стоит буква `E` и представление **n** в виде десятичного целого. Сколько цифр нужно ввести для дробной части **m** или **a**? Должна быть хотя бы одна цифра, чтобы представить дробную часть, плюс ровно столько цифр, сколько нужно, чтобы различить значение аргумента среди соседних значений типа Float (или Double, если обрабатывается число типа Double). То есть, предположим, что **x** - это точное математическое значение, представленное десятичным представлением,

			произведенным этим способом для конечного ненулевого аргумента **d**. Затем **d** должен стать значением типа Double, наиболее близким к **x**; или если два значения Double одинаково близки **x**, тогда **d** должен быть одним из них, а наименьшей мантиссой из **d** должен быть 0.
E	Double		Правила, как для типа Float
D	Date		Преобразовано в строку в форме "**yyyy-MM-dd HH:mm:ss.SSS**", где **yyyy** - год **MM** - месяц **dd** - день месяца **HH** - час (0-23) **mm** -минуты **ss** - секунды **SSS** -миллисекунды Конвертация должна использовать временную зону UTC.
T	Data Table	Да	Таблица с включенными данными закодирована в строку согласно правилам Кодирования таблиц данных 2123
C	Color		Преобразовано в строку в форме "***#RRGGBB***", где **RR** - значение красного цвета (0-255) - шестнадцатеричная форма **GG** - значение зеленого цвета (0-255) - шестнадцатеричная форма **BB** - значение синего цвета (0-255) - шестнадцатеричная форма
A	Data Block	Да	Конвертируется в строку в следующем виде: ***Version / ID / Name / Preview_length / Data_length / Preview Data*** Строка содержит несколько частей, разделенных символом /. Это следующие части: • **Версия**. Версия блока данных, кодирующего алгоритм, в данный момент 0. • **Идентификатор**. Уникальный идентификатор этого блока данных в инсталляторе AtomMind Server. Идентификатор NULL (неопределенный) представлен одним символом 0x1A (см. Кодирование значения NULL 2129). • **Имя**. Название блока данных, обычно это имя файла, загруженного в блок данных. Имя NULL (неопределенное) представлено одним символом 0x1A (SUB) (см. Кодирование значения NULL 2129). • **Preview_length**. Количество байтов в закодированном блоке данных *предварительного просмотра*. Предварительный просмотр - это краткое представление блока данных, например,

			черновой вариант изображения. Длина -1 означает, что предпросмотр недоступен. • **Data_length**. Количество байтов в закодированных данных. Длина -1 означает, что данные недоступны. • **Просмотр**. Закодированные байты данных просмотра блока. • **Данные**. Закодированные байты данных блока.  Между полями **Предпросмотр** и **Данные** нет разделителя. Данные предпросмотра должны отделяться от основных данных в соответствии с их длиной, определённой **Preview_length** и **Data_length**. При перекодировании блока данных в строку байты **Предпросмотра** и **Данных** конвертируются в символы Юникода с кодами 0...255, т.е. символами ASCII.

КОДИРОВАНИЕ ЗНАЧЕНИЙ ВЫБОРА

Значения для поля выбора кодируются как список элементов. Каждое имя элемента - это видимое **описание** значения выбора для пользователя (то, что пользователь увидит в окне списка). Значения элемента - это значение выбора, закодированное в строку, как описано в разделе кодирование значения 2125.

 **Пример:** список трёх значений выбора для поля с целым числом можно закодировать следующим образом: <Zero=0><One=1><Two=2>

Кодирование валидаторов

Валидаторы полей закодированы как список элементов, по одному для каждого валидатора. Имя валидатора - **Код типа** валидатора, в то время как его значение содержит **опции, специфичные для валидатора**.

ВАЛИДАТОРЫ ПОЛЕЙ

Список поддерживаемых валидаторов поля:

Код типа	Описание	Подходящие типы полей	Опции, характерные для валидатора
L	**Валидатор ограничений.** Проверяет, соответствует ли значение диапазону, заданному параметрами валидатора.	Строка, Целое, Длинное, Плавающее, Данные	Опции валидатора закодированы в строку как два целых числа, разделённые пробелом. Первое число обозначает минимальное значение диапазона, а второе число определяет максимальное. Эти числа имеют разное значение для разных типов полей: Для строковых полей эти параметры ограничивают минимальную и максимальную длину строки. Для целочисленных, длинных и плавающих полей они указывают минимальное и максимальное значение. Для полей с данными они ограничивают число байтов, которые могут содержаться в блоке данных. Границы включены для всех типов полей (т.е., ограничение "3" разрешило бы строке abc - включать 3 символа).
R	**Валидатор регулярных выражений.** Проверяет, соответствует ли значение строки регулярному	Строка	Строка опций валидатора содержит регулярное выражение 2142, которому значение поля должно соответствовать. За ним может следовать (не обязательно) сообщение об ошибке, отделённой от регулярного выражения строкой ^^ . Если валидация неуспешна (т.е. значение строки не соответствует

	выражению, заданному параметром валидатора.		регулярному выражению), это сообщение об ошибке будет показано пользователю.

Пример 1: `<L=0 255>`

Если данный валидатор ограничений добавлен в формат поля строки, он разрешит лишь строки, чья длина составляет от 0 до 255 символов. Если он определён для целочисленных полей, он будет ограничивать значения полей до чисел, которые больше или равны 0 и меньше или равны 225.

Пример 2: `<R=^[_A-Za-z0-9-]+(\\.[_A-Za-z0-9-]+)*@[A-Za-z0-9-]+(\\.[A-Za-z0-9-]+)*(\\.[_A-Za-z0-9-]+)^^Invalid E-Mail>`

`^[_A-Za-z0-9-]+(\\.[_A-Za-z0-9-]+)*@[A-Za-z0-9-]+(\\.[A-Za-z0-9-]+)*(\\.[_A-Za-z0-9-]+)` -- это регулярное выражение (Первый символ ^ входит в него). После этого идёт разделитель ^^ , затем следует текст для сообщения об ошибке "Invalid E-Mail".

Этот валидатор регулярных выражений проверяет, содержит ли поле правильный e-mail и отправляет сообщение об ошибке **Invalid E-Mail,** если адрес неправильный**.** За дополнительной информацией о регулярных выражениях обратитесь в раздел приложения Синтаксис регулярных выражений 2142.

ВАЛИДАТОРЫ ЗАПИСЕЙ

Список поддерживаемых валидаторов записей:

Код типа	Описание	Опции, характерные для валидатора
K	**Ключевые поля**. Проверяет, существует ли комбинация значений ключевых полей в таблице. Применяется во время добавления записи.	Отсутствуют - ключевые поля помечаются флажком Ключевое поле формата поля.

ВАЛИДАТОРЫ ТАБЛИЦ

Список поддерживаемых валидаторов таблиц:

Код типа	Описание	Опции, характерные для валидатора
K	**Ключевые поля**. Проверяет, является ли комбинация значений ключевых полей уникальной для каждой записи.	Отсутствуют - ключевые поля отмечены флажком Ключевое поле формата поля.
E	**Валидатор выражений.** Оценивает выражение 112, имеющее данную таблицу как таблицу по умолчанию 120. Если выражение возвращает NULL, таблица считается верной, в ином случае вывод выражения конвертируется в строку и используется как текст сообщения об ошибке.	Текст выражения.

Пример: `<E={activationThreshold} > {deactivationThreshold} ? null : 'Activation threshold must be greater than deactivation threshold'>`

Проверяет, больше ли одно поле другого.

КОДИРОВКА ПУСТЫХ ЗНАЧЕНИЙ

Пустое значение **NULL** ("<Not set>") кодируется одним символом 0x1A (SUB). Это правило применяется для кодирования пустых (NULL) значения ячеек таблицы, значений по умолчанию для полей таблицы, значений выбора и любого другого места, где могут появиться значения поля.

Если для кодирования Таблицы данных используются видимые разделители, пустые (**NULL)** значения кодируются как символы "**^**".

КОДИРОВКА ПЕРЕДАЧИ

Значения поля кодируются при передаче для того, чтоб убрать символы, используемые форматом кодирования Таблицы данных и Протоколом взаимодействия AtomMind 2119. Эти символы заменяются специальными шаблонами согласно следующей таблице:

Символ	Заменяется на
0x25 (%)	%%
0x02 (STX)	%^
0x0D (CR)	%$
0x17 (ETB)	%/
0x1C (FS)	%<
0x1D (GS)	%>
0x1E (RS)	%=

Обратите внимание, что шаблоны в колонке "Заменяется на" являются символьными строками - это то, что Вы видите в таблице.

Кодирование записей данных

Каждая запись данных кодируется в строку согласно следующему формату:

```
[<I=record_ID>]<field_value><field_value>…
```

Элементы без имен - это значения поля. Значения полей кодируются одно за другим, в том же порядке, как они появляются в дескрипторе формата таблиц 2124.

Имя элемента	Значение элемента
I	ID записи (Длинное число)

Режимы кодирования строки

Каждую таблицу данных 49 можно кодировать в двух режимах:

- используя видимые разделители
- используя невидимые разделители

Три специальных символа используются в качестве разделителей для кодирования:

Видимый разделитель	Невидимый разделитель (код)
<	0x1C
>	0x1D
=	0x1E

В этом разделе во всех примерах используются видимые разделители. Единственное ограничение при кодировании таблицы с использованием видимых разделителей - это его разные элементы, когда они закодированы в строку, то не должны содержать данные разделители.

20.4 Кодирование таблиц данных в формате XML

В этом приложении описывается, как Таблицы данных 49 кодируются в (и декодируются из) XML. Таблицы данных кодируются в XML только, когда их необходимо передать как строковые аргументы для функций Web-сервиса 1417, во всех других случаях AtomMind использует свой собственный синтаксис кодирования 2123.

В большинстве остальных компонентов AtomMind (базе данных 692, Протоколе взаимодействия AtomMind 2119 и пр.) таблицы данных хранятся и передаются при помощи так называемого **собственного** кодирования. Этот формат описан в отдельном приложении 2123.

XML - схема для таблиц данных

Следующая XML-схема используется для кодирования таблиц данных:

XML-схема - это описание типа XML-документа, обычно выраженного в виде ограничений на структуру и содержание документов такого типа. XML-схема предоставляет документ на высоком уровне абстракции.

Дополнительную информацию см. на сайте: http://www.w3.org/XML/Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
	<xs:element name="table">
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="format" minOccurs="0" maxOccurs="1"/>
				<xs:element ref="records" minOccurs="0" maxOccurs="1"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="format">
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="fields" minOccurs="0"/>
			</xs:sequence>
			<xs:attribute name="maxRecords" type="xs:integer"/>
			<xs:attribute name="minRecords" type="xs:integer"/>
		</xs:complexType>
	</xs:element>
	<xs:element name="fields">
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="field" maxOccurs="unbounded"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="field">
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="selectionValues" minOccurs="0"/>
				<xs:element ref="defaultValue" minOccurs="0"/>
			</xs:sequence>
			<xs:attribute name="description"/>
			<xs:attribute name="name" type="xs:NCName" use="required"/>
			<xs:attribute name="notReplicated" type="xs:boolean"/>
			<xs:attribute name="nullable" type="xs:boolean"/>
			<xs:attribute name="readonly" type="xs:boolean"/>
```

```
                <xs:attribute name="type" type="fieldType" use="required"/>
            </xs:complexType>
        </xs:element>
        <xs:element name="selectionValues">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="option" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="option">
            <xs:complexType mixed="true">
                <xs:complexContent>
                    <xs:extension base="fieldVal">
                        <xs:attribute name="description" use="required"/>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
        </xs:element>
        <xs:element name="defaultValue" type="fieldVal"/>
        <xs:element name="records">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="record" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="record">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="value" maxOccurs="unbounded" minOccurs="0">
                        <xs:complexType mixed="true">
                            <xs:complexContent>
                                <xs:extension base="fieldVal">
                                    <xs:attribute name="name"
use="required"/>
                                </xs:extension>
                            </xs:complexContent>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
```

```
		</xs:complexType>
	</xs:element>
	<xs:simpleType name="fieldType">
		<xs:restriction base="xs:string">
			<xs:enumeration value="A"/>
			<xs:enumeration value="I"/>
			<xs:enumeration value="D"/>
			<xs:enumeration value="T"/>
			<xs:enumeration value="S"/>
			<xs:enumeration value="B"/>
			<xs:enumeration value="L"/>
			<xs:enumeration value="F"/>
			<xs:enumeration value="O"/>
			<xs:enumeration value="C"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:complexType name="fieldVal" mixed="true">
		<xs:choice minOccurs="0">
			<xs:element ref="data"/>
			<xs:element ref="table"/>
		</xs:choice>
	</xs:complexType>
	<xs:element name="data">
		<xs:complexType>
			<xs:attribute name="name" type="xs:string"/>
			<xs:attribute name="contentType" type="xs:string"/>
		</xs:complexType>
	</xs:element>
</xs:schema>
```

Эта XML-схема определяет структуру XML-документа таблицы данных. Корневой элемент - `table` определяет Таблицу данных[49].

Элемент `table` может содержать один элемент `format` , который описывает формат[49] таблиц данных. Он может также содержать опциональный элемент `records` , который включает список записей таблиц с данными.

Элемент `format` должен включать подэлемент `fields` , содержащий список полей таблицы. Он может иметь атрибуты `minRecords` и `maxRecords` , которые определяют минимальное и максимальное количество записей в таблице. Если `minRecords` не определён, минимальное количество записей равно нулю. Если `maxRecords` не определено, максимальное количество записей не ограничено (на практике ограничено 2^64).

Элемент `fields` состоит из одного или более элементов `field`. Элемент `field` определяет формат одного поля[49]. Для него требуются атрибуты `name` и `type`. Элемент `fieldType` может быть представлен одним кодом типа, определённого здесь[2125]. Атрибуты `description`, `notReplicated`, `nullable` и `readonly` опциональны. Элемент `field` может быть также включать подэлементы `selectionValues` и `defaultValue`. Дополнительную информацию об этих атрибутах и элементах можно найти в разделе Таблицы данных[49].

Элемент `selectionValues` может содержать один или более элементов `option`. Элемент `value` относится к типу `fieldVal`. Элемент `value` требует атрибут `description` , который определяет описание определённого значения выбора.

defaultValue тоже относится к типу fieldVal.

Элементы типа fieldVal используются для хранения значений данных ячеек таблиц, значений выбора и значений по умолчанию. Элементы данного типа могут быть представлены:

- элементами table (включённые таблицы)
- элементами data (бинарные блоки данных)
- простым текстом (все типы значений, см правила кодирования здесь 2125)

Элементы records содержат последовательность элементов записи. Каждая запись представляет одну запись Таблицы данных. Она содержит количество элементов value. Их тип данных - fieldVal, описан ранее. У каждого элемента есть необходимый атрибут name, включая имя поля, чьё значение представляет value.

ПРИМЕРЫ ТАБЛИЦЫ ДАННЫХ, ЗАКОДИРОВАННЫХ В ФОРМАТЕ XML

```
<?xml version="1.0" encoding="UTF-8"?>

<table xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <!-- Data Table -->

     <format maxRecords="1" minRecords="1"> <!-- Table Format Descriptor -->

           <fields> <!-- Field Set -->

                 <field description="Username" name="username" notReplicated="true"
readonly="true" type="S"/>

                 <field description="First Name" name="firstname" notReplicated="true"
nullable="true" type="S"/>

                 <field description="Last Name" name="lastname" notReplicated="true"
nullable="true" type="S"/>

                 <field description="Password" name="password" notReplicated="true"
type="S"/>

                 <field description="Country" name="country" type="I">

                       <selectionValues>

                             <option description="Albania">1</option>

                             <option description="Algeria">2</option>

                             -- skipped --

                       </selectionValues>

                 </field>

                 <field description="Region/State/Province/Area" name="region"
nullable="true" type="S"/>

                 <field description="ZIP Code" name="zip" nullable="true" type="S"/>

                 <field description="City" name="city" nullable="true" type="S"/>

                 <field description="Address 1" name="address1" nullable="true"
type="S"/>

                 <field description="Address 2" name="address2" nullable="true"
type="S"/>

                 <field description="Comments" name="comments" nullable="true"
type="S"/>
```

```
                <field description="Company" name="company" nullable="true"
type="S"/>

                <field description="Department" name="department" nullable="true"
type="S"/>

                <field description="E-mail Address" name="email" notReplicated="true"
nullable="true" type="S"/>

                <field description="Phone No." name="phone" notReplicated="true"
nullable="true" type="S"/>

                <field description="Fax No." name="fax" notReplicated="true"
nullable="true" type="S"/>

                <field description="Time Zone" name="timezone" type="S">

                      <selectionValues>

                            <option description="GMT-12:00,
Etc/GMT+12">Etc/GMT+12</option>

                            <option description="GMT-11:00,
Etc/GMT+11">Etc/GMT+11</option>

                            -- skipped --

                      </selectionValues>

                      <defaultValue>Europe/Moscow</defaultValue>

                </field>

                <field description="Locale" name="locale" type="S">

                      <selectionValues>

                            <option description="Default"/>

                            <option description="aa">aa</option>

                            <option description="ab">ab</option>

                            -- skipped --

                      </selectionValues>

                      <defaultValue>en</defaultValue>

                </field>

                <field description="Date Pattern" name="datepattern" type="S"/>

                <field description="Time Pattern" name="timepattern" type="S"/>

                <field description="Enable Automatic Registration of Device Servers"
name="dsautoregistration" type="B"/>

           </fields>

     </format>

     <records> <!-- Data Table Records -->
```

```
        <record> <!-- First Record -->
              <value name="username">ronnie</value>
              <value name="firstname">Ronnie</value>
              <value name="lastname">O'Sullivan</value>
              <value name="password">11111</value>
              <value name="country">218</value>
              <value name="datepattern">dd.MM.yyyy</value>
              <value name="timepattern">HH:mm:ss</value>
              <value name="dsautoregistration">1</value>
        </record>
        <record> <!-- Second Record -->
              <value name="username">john</value>
              <value name="firstname">John</value>
              <value name="lastname">Doe</value>
              <value name="password">12345</value>
              <value name="country">83</value>
              <value name="datepattern">dd.MM.yyyy</value>
              <value name="timepattern">HH:mm:ss</value>
              <value name="dsautoregistration">1</value>
        </record>
    </records>
</table>
```

20.5 Кодирование свойств в формате XML

В этом приложении описывается, как переменные 61 (свойства) контекста кодируются (и декодируются) в формате XML.

Свойства кодируются и декодируются только если их экспортируют или импортируют, используя Редактор свойств 377. В противном случае, AtomMind хранит и обрабатывает свойства как Таблицы данных 49. Каждое свойство - это отдельная Таблица данных.

схема XML для свойств

Схема XML для кодирования свойств выглядит следующим образом:

Схема XML - это описание XML-документа, обычно с точки зрения правил структуры и содержания документов такого типа, а не только базовых правил синтаксиса, накладываемых самим форматом XML. Схема XML дает представление о типа документа на относительно высоком уровне абстракции.

Более подробно об этом см. http://www.w3.org/XML/Schema.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
attributeFormDefault="unqualified" elementFormDefault="qualified">
     <xs:element name="properties" type="propertiesType"/>
     <xs:complexType name="validatorsType">
          <xs:simpleContent>
               <xs:extension base="xs:string">
                    <xs:attribute type="xs:string" name="contentType"
use="optional"/>
               </xs:extension>
          </xs:simpleContent>
     </xs:complexType>
     <xs:complexType name="fieldType">
          <xs:sequence>
               <xs:element type="validatorsType" name="validators" minOccurs="0"/>
               <xs:element type="xs:string" name="help" minOccurs="0"/>
               <xs:element type="selectionValuesType" name="selectionValues"
minOccurs="0"/>
               <xs:element type="xs:string" name="group" minOccurs="0"/>
               <xs:element name="defaultValue">
                    <xs:complexType mixed="true">
                         <xs:sequence>
                              <xs:element type="xs:string" name="nullValue"
minOccurs="0"/>
                              <xs:element type="tableType" name="table"
minOccurs="0"/>
                              <xs:element type="dataType" name="data"
minOccurs="0"/>
                         </xs:sequence>
                    </xs:complexType>
               </xs:element>
          </xs:sequence>
          <xs:attribute type="xs:string" name="name" use="optional"/>
          <xs:attribute type="xs:string" name="type" use="optional"/>
          <xs:attribute type="xs:string" name="hidden" use="optional"/>
          <xs:attribute type="xs:string" name="keyfield" use="optional"/>
          <xs:attribute type="xs:string" name="notReplicated" use="optional"/>
          <xs:attribute type="xs:string" name="nullable" use="optional"/>
          <xs:attribute type="xs:string" name="description" use="optional"/>
          <xs:attribute type="xs:string" name="editor" use="optional"/>
          <xs:attribute type="xs:string" name="advanced" use="optional"/>
```

```
            <xs:attribute type="xs:string" name="readonly" use="optional"/>
            <xs:attribute type="xs:string" name="inline" use="optional"/>
            <xs:attribute type="xs:string" name="extendableSelectionValues"
use="optional"/>
      </xs:complexType>
      <xs:complexType name="fieldsType">
            <xs:sequence>
                  <xs:element type="fieldType" name="field" maxOccurs="unbounded"
minOccurs="0"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="formatType">
            <xs:sequence>
                  <xs:element type="fieldsType" name="fields"/>
                  <xs:element type="bindingsType" name="bindings" minOccurs="0"/>
                  <xs:element type="validatorsType" name="validators" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute type="xs:int" name="maxRecords" use="optional"/>
            <xs:attribute type="xs:byte" name="minRecords" use="optional"/>
            <xs:attribute type="xs:string" name="reorderable" use="optional"/>
      </xs:complexType>
      <xs:complexType name="valueType" mixed="true">
            <xs:sequence>
                  <xs:element type="xs:string" name="nullValue" minOccurs="0"/>
                  <xs:element type="tableType" name="table" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute type="xs:string" name="name" use="optional"/>
      </xs:complexType>
      <xs:complexType name="recordType" mixed="true">
            <xs:sequence>
                  <xs:element type="valueType" name="value" maxOccurs="unbounded"
minOccurs="0"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="recordsType">
            <xs:sequence>
                  <xs:element type="recordType" name="record" maxOccurs="unbounded"
minOccurs="0"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="tableType">
            <xs:sequence>
                  <xs:element type="formatType" name="format"/>
```

```
            <xs:element type="recordsType" name="records" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="entryType">
        <xs:sequence>
            <xs:element type="xs:string" name="string"/>
            <xs:element type="tableType" name="table"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="defaultValueType">
        <xs:sequence>
            <xs:element type="xs:string" name="nullValue" minOccurs="0"/>
            <xs:element type="tableType" name="table" minOccurs="0"/>
            <xs:element type="dataType" name="data" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="bindingType">
        <xs:sequence>
            <xs:element type="xs:string" name="reference"/>
            <xs:element type="xs:string" name="expression"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="bindingsType">
        <xs:sequence>
            <xs:element type="bindingType" name="binding" maxOccurs="unbounded"
minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="optionType" mixed="true">
        <xs:sequence>
            <xs:element type="xs:string" name="nullValue" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute type="xs:string" name="description" use="optional"/>
    </xs:complexType>
    <xs:complexType name="selectionValuesType">
        <xs:sequence>
            <xs:element type="optionType" name="option" maxOccurs="unbounded"
minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="dataType">
        <xs:sequence>
```

```
                <xs:element type="xs:string" name="nullValue"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="propertiesType">
            <xs:sequence>
                <xs:element type="entryType" name="entry" maxOccurs="unbounded"
minOccurs="0"/>
                <xs:element type="propertiesType" name="properties" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute type="xs:string" name="class" use="optional"/>
      </xs:complexType>
</xs:schema>
```

Данная схема XML определяет структуру XML-файла с экспортированными свойствами контекста. Корневой элемент - `properties`, содержит все экспортированные свойства.

Элемент `properties` содержит 0 или более элементов `entry`. Эти элементы содержат индивидуальные свойства.

Элемент `entry` включает два дочерних элемента: `string` и `table`. Элемент `string` содержит имя свойства. Элемент `table` содержит таблицу данных 49 для этого свойства.

Таблицы данных для свойств внутри элементов `table` хранятся в формате XML 2130.

ПРИМЕРЫ КОДИРОВАНИЯ СВОЙСТВ В ФОРМАТЕ XML

```
<?xml version="1.0" ?>
<properties>
      <properties class="linked-hash-map">
            <entry>
                <string>connectionProperties</string>
                <table>
                      -- skipped --
                </table>
            </entry>
            <entry>
                <string>performanceTesting</string>
                <table>
                      -- skipped --
                </table>
            </entry>
            <entry>
```

```
        <string>genericProperties</string>
        <table>
                -- skipped --
        </table>
    </entry>
    <entry>
        <string>managedVariables</string>
        <table>
                -- skipped --
        </table>
    </entry>
    <entry>
        <string>managedFunctions</string>
        <table>
                -- skipped --
        </table>
    </entry>
    <entry>
        <string>managedEvents</string>
        <table>
                -- skipped --
        </table>
    </entry>
    <entry>
        <string>statisticsProperties</string>
        <table>
                -- skipped --
        </table>
    </entry>
    <entry>
        <string>granulator</string>
```

```
                <table>
                        -- skipped --
                </table>
            </entry>
            <entry>
                <string>settingSyncOptions</string>
                <table>
                        -- skipped --
                </table>
            </entry>
            <entry>
                <string>ccmConfigurationManagement</string>
                <table>
                        -- skipped --
                </table>
            </entry>
            <entry>
                <string>ccmCompliancePolicies</string>
                <table>
                        -- skipped --
                </table>
            </entry>
        </properties>
</properties>
```

20.6 Синтаксис регулярных выражений

Регулярное выражение подобно сложному шаблону. Другими словами, это строка, которая описывается или соответствует другим строкам согласно определенным правилам синтаксиса.

AtomMind использует синтаксис Java для обработки регулярных выражений. См. ссылки ниже 2146.

Обзор компонентов регулярных выражений

Компонент	*Соответствие*
Символы	

x	Символ x
\\	Символ обратная косая черта
\0n	Символ с восьмеричным значением (octal value) 0n (0 <= n <= 7)
\0nn	Символ с восьмеричным значением (octal value) 0nn (0 <= n <= 7)
\0mnn	Символ с восьмеричным значением (octal value) 0mnn (0 <= m <= 3, 0 <= n <= 7)
\xhh	Символ с шестнадцатеричным значением (hexadecimal value) 0xhh
\uhhhh	Символ с шестнадцатеричным значением (hexadecimal value) 0xhhhh
\t	Символ табуляции ('\u0009')
\n	Символ новой строки ('\u000A')
\r	Символ возврата каретки ('\u000D')
\f	Символ перевода строки ('\u000C')
\a	Символ колокольчика ('\u0007')
\e	Символ escape ('\u001B')
\c x	Соответствует управляющему символу ASCII, который задан как X или x, где X или x является буквой управляющего
Классы символов	
[abc]	a, b, or c (простой класс)
[^abc]	Любой символ, кроме a, b, или c (отрицание)
[a-zA-Z]	c a по z и с A по Z включительно (диапазон)
[a-d[m-p]]	c a по (d или с m по p): [a-dm-p] (объединение)
[a-z&&[def]]	d, e или f (пересечение)
[a-z&&[^bc]]	c a по z, исключая b и c: [ad-z] (вычитание)
[a-z&&[^m-p]]	c a по z, и без от m до p: [a-lq-z](вычитание)
Предопределенный класс символов	
.	любой символ (может соответствовать или не соответствовать завершителю строки)
\d	числовой: [0-9]
\D	не числовой: [^0-9]
\s	пробельный символ: [\t\n\x0B\f\r]
\S	не пробельный символ: [^\s]
\w	словообразующий символ: [a-zA-Z_0-9]
\W	не словообразующий символ: [^\w]
Класс символов POSIX (только US-ASCII)	
\p{Lower}	алфавитный символ нижнего регистра: [a-z]
\p{Upper}	алфавитный символ верхнего регистра:[A-Z]
\p{ASCII}	все ASCII:[\x00-\x7F]
\p{Alpha}	алфавитный символ:[\p{Lower}\p{Upper}]
\p{Digit}	десятичный знак: [0-9]
\p{Alnum}	алфавитно-числовой символ:[\p{Alpha}\p{Digit}]
\p{Punct}	пунктуационные знаки: один из !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
\p{Graph}	видимый знак: [\p{Alnum}\p{Punct}]
\p{Print}	печатный знак: [\p{Graph}\x20]

\p{Blank}	пробел или табуляция: [\t]
\p{Cntrl}	управляющий знак: [\x00-\x1F\x7F]
\p{XDigit}	десятичный знак: [0-9a-fA-F]
\p{Space}	пробельный символ: [\t\n\x0B\f\r]
Обнаружители границ	
^	Начало строки
$	Конец строки
\b	граница слова
\B	граница не слова
\A	начало ввода
\G	окончание предыдущего совпадения (previous match)
\Z	конец ввода для указателя конца, если есть
\z	конец ввода
Жадные Квантаторы (Greedy quantifiers)	
X?	X, один раз или ни одного
X*	X, нуль или много раз
X+	X, один или более раз
X{n}	X, точное количество n-раз
X{n,}	X, по меньшей мере n-раз
X{n,m}	X, как минимум n-раз, но не более m-раз
Неохотные Квантаторы (Reluctant quantifiers)	
X??	X, один или ни одного
X*?	X, нуль или более раз
X+?	X, один или более раз
X{n}?	X, точное количество n-раз
X{n,}?	X, как минимум n-раз
X{n,m}?	X, как минимум n-раз, но не более m-раз
Ревнивые Квантаторы (Possessive quantifiers)	
X?+	X, один или ни одного
X*+	X, нуль или более раз
X++	X, один или более раз
X{n}+	X, точное количество n-раз
X{n,}+	X, как минимум n-раз
X{n,m}+	X, как минимум n-раз, но не более m-раз
Логические операторы	
XY	за X следует Y
X|Y	как X, так и Y
(X)	X, как группа захвата
Обратные ссылки	
\n	Все, что соответвует группе захвата n

Кавычки	
\	Не соответствует ни одному символу, только цитирует следующий знак
\Q	Не соответствует ни одному символу, только цитирует все знаки до \E
\E	Не соответствует ни одному символу, только заканчивает цитирование, которое началось с \Q

СИМВОЛЫ ОБРАТНАЯ КОСАЯ ЧЕРТА, СИМВОЛ ESC И СИМВОЛ ЭКРАНИРОВАНИЯ

Символ обратная косая черта ('\') служит для введения экранированных конструкций, как определено в таблице ниже, а также для экранирования символов, которые в ином случае были бы проинтерпретированы как не экранированные конструкции. Таким образом, выражение \\ соответствует одной обратной косой черте и левой фигурной скобке \{ .

Ошибочно использовать обратную косую черту раньше любого алфавитного символа, который не обозначает экранированную конструкцию; они сохранены для будущих расширений языка регулярных выражений. Обратную косую черту можно использовать до не алфавитного символа, не зависимо от того, что символ является частью не экранированной конструкции.

СИМВОЛЬНЫЕ КЛАССЫ

Символьные классы могут появлятся и в других классах символов, а также могут быть составлены оператором объединения (неявно) и оператором пересечения. (&&). Оператор объединения обозначает класс, которые содержит каждый символ, который является как минимум одним из своих классов операндов. Оператор пересечения обозначает класс, который содержит каждый символ, который относится к обоим своим классам операндов.

Приоритет операторов класса символов выглядит следующим образом (от высшего к низшему):

1	Экранирование символа	\x
2	Группировка	[...]
3	Диапазон	a-z
4	Объединение	[a-e][i-u]
5	Пересечение	[a-z&&[aeiou]]

Обратите внимание, что различный набор метасимволов в действительности находятся внутри символьного класса, а не снаружи символьного класса. Например, регулярное выражение . теряет свое особое значение внутри символьного класса, в то время как выражение - становится метасимволом, формирующим диапазон.

РАЗДЕЛИТЕЛИ СТРОК

Разделитель строк - это одно или двух символьная последовательность, которая отмечает конец строки последовательности символов ввода. Ниже представлены разделители строк:

- Символ новой строки ('\n'),
- Символ возврата каретки, за которым сразу же следует символ новой строки ("\r\n"),
- Изолированный символ возврата каретки ('\r'),
- Символ следующей строки ('\u0085'),
- Разделитель строк ('\u2028'), или
- Разделитель абзацев ('\u2029).

ГРУППЫ И ЗАХВАТ

Группы захвата нумеруются по открывающим их круглым скобкам слева направо. В выражении `((A)(B(C)))`, например, есть четыре таких группы:

1 ((A)(B(C)))

2 (A)

3 (B(C))

4 (C)

Группа ноль означает целое выражение.

Группы захвата называются так, потому что во время поиска соответствий каждая подпоследовательность введенной последовательности, которая соответствует такой группе, сохраняется. Захваченная подпоследовательность может использоваться в выражении позже, через обратную ссылку, а также может извлекаться из обнаружителя соответствий, как только завершается операция поиска соответствий.

Захваченный ввод, ассоциируемый с группой, всегда является подпоследовательностью, которой в самый последний раз соответствовала группа. Если группа рассчитывается второй раз из-за квантификации, тогда ее ранее захваченное значение, если такое было, будет сохранено при невыполнении второго расчета. Соответствие строки `aba` выражению `(a(b)?)+`, например, оставляет группе настройку на `b`. Весь захваченный ввод сбрасывается в начале каждого соответствия.

Группы, начинающиеся с `(?`, чистые, незахваченные группы, которые не захватывают текст и не входят в общее число групп.

Дополнительные источники информации о регулярных выражениях

- http://en.wikipedia.org/wiki/Regular_expression. Статья о регулярных выражениях в Wikipedia.
- http://www.regular-expressions.info/. Очень информативный сайт о регулярных выражениях.
- http://www.regular-expressions.info/reference.html. Основной синтаксис для регулярных выражений.
- http://www.regular-expressions.info/examples.html. Примеры регулярных выражений.
- http://java.sun.com/j2se/1.6.0/docs/api/java/util/regex/Pattern.html. Синтаксис регулярных выражений для Java.

20.7 Шаблоны форматирования времени и даты

Шаблоны форматирования времени и даты осуществляют контроль за отображением значений Времени и Даты (т.е. преобразование метки времени в строку). В строках шаблонов времени и даты не заключённые в кавычки буквы от 'A' до 'Z' и от 'a' до 'z' интерпретируются как буквы шаблона, представляющие компоненты строки даты или времени. Текст можно цитировать, используя кавычки ('), чтобы избежать интерпретации. "''" представляет одну цитату. Все остальные символы не интерпретируются, они просто копируются в строку вывода во время форматирования или соотносятся со строкой ввода во время синтаксического анализа.

Определены следующие буквы шаблона (все остальные символы от 'A' до 'Z' и от 'a' до 'z' зарезервированы):

Буква	Компонент даты или времени	Представление	Примеры
G	Указатель летоисчисления	Текст	AD
y	Год	Год	1996; 96
Y	Недельный год	Год	2009; 09
M	Месяц в году	Месяц	July; Jul; 07
w	Неделя в году	Число	27
W	Неделя в месяце	Число	2
D	День в году	Число	189
d	День в месяце	Число	10
F	День недели в месяце	Число	2
E	День в неделе	Текст	Tuesday; Tue
u	Номер для недели (1 = понедельник, ..., 7 = воскресенье)	Number	1
a	Метка AM/PM	Текст	PM
H	Час дня (0-23)	Число	0
k	Час дня	Число	24
K	Часы в AM/PM (0-11)	Число	0
h	Часы в AM/PM (1-12)	Число	12
m	Минута часа	Число	30

s	Секунда минуты	Число	55
S	Миллисекунды	Число	978
z	Часовой пояс	Общий часовой пояс	Pacific Standard Time; PST; GMT-08:00
Z	Часовой пояс	RFC 822 часовой пояс	-0800
X	Часовой пояс	ISO 8601 часовой пояс	-08; -0800; -08:00

Буквы шаблона обычно повторяются, поскольку их количество определяет точность представления:

- **Текст:** для форматирования, если число букв шаблона 4 или более, используется полная форма; в ином случае используется краткая форма или аббревиатура, если она есть. Для синтаксического анализа приемлемы обе формы, независимые от количества букв шаблона.
- **Число**: для форматирования число букв шаблона - минимальное количество цифр, а более короткие числа добавляются нулями к этому количеству. При синтаксическом анализе количество букв шаблона игнорируется, если только не требуется разделить два соседних поля.
- **Год:** для форматирования, если число букв шаблона равно 2, год усекается до 2 цифр; в ином случае год интерпретируется как число.
- **Месяц:** если число букв шаблона 3 и более, месяц интерпретируется как текст; в ином случае месяц интерпретируется как число.

Часы должны соответствовать диапазону от 0 до 23, а минуты - от 00 до 59. Формат - независим от языка, и цифры следует брать из Основного Латинского Блока юникода.

Примеры

Следующие примеры показывают, как шаблоны даты и времени интерпретируются согласно американской языковой настройке. Данная дата и время - 2001-07-04 12:08:56 местное время часового пояса тихоокеанской части США.

Шаблоны даты и времени	**Результат**
"yyyy.MM.dd G 'at' HH:mm:ss"	2001.07.04 AD at 12:08:56
"EEE, MMM d, ''yy"	Wed, Jul 4, '01
"h:mm a"	12:08 PM
"hh 'o''clock' a"	12 o'clock PM
"K:mm a"	0:08 PM
"yyyyy.MMMMM.dd GGG hh:mm aaa"	02001.July.04 AD 12:08 PM
"EEE, d MMM yyyy HH:mm:ss"	Wed, 4 Jul 2001 12:08:56
"yyMMddHHmmss"	010704120856

20.8 Шаблоны форматирования чисел

Шаблоны форматирования чисел осуществляют контроль за отображения числовых значений (т.е. за преобразованием чисел в строку). Шаблон форматирования чисел определяет правила для форматирования десятичных чисел. Он поддерживает различные виды чисел, включая целые числа (123), числа с фиксированной запятой (123.4), научную форму (1.23E4), проценты (12%) и валюту ($123).

Шаблоны

Шаблон содержит положительные и отрицательные подшаблоны, например, "#,##0.00;(#,##0.00)". У каждого подшаблона есть префикс, числовая часть и суффикс. Отрицательный подшаблон необязательный; если он отсутствует, тогда положительный подшаблон с префиксом в виде знака минус ('-' в большинстве языковых стандартов) используется как отрицательный подшаблон. Т.е., "0.00" эквивалентно "0.00;-0.00". Если есть явно определенный отрицательный подшаблон, он служит только для определения отрицательного префикса и суффикса; количество цифр, минимальное количество цифр и другие характеристики - все аналогичны положительному шаблону. Это означает, что представление "#,##0.0#;(#)" абсолютно идентично "#,##0.0#;(#,##0.0#)".

Разделитель групп обычно используется для тысяч, но в некоторых странах он разделяет десятитысячные числа. Размер групп - это количество цифр между символами группировки, например, 3 для 100,000,000 или 4 для 1,0000,0000. Если Вы используете шаблон с множеством символов группировок, интервал между последним и

концом целого числа - это, то что используется. Таким образом: "#,##,###,####" == "######,####" == "##,####,####".

Специальные символы шаблонов

Многие символы в шаблоне берутся точно как есть; они соотносятся во время синтаксического анализа и выводятся неизмененными во время форматирования. Специальные символы, с другой стороны, означают другие символы, строки или классы символов. Их следует брать в кавычки, если не указано иное, если они должны появиться в префиксе или суффиксе как литералы.

Символ	Расположение /Location	Значение
0	Число	Цифра
#	Число	Цифра, ноль отображает отсутствие.
.	Число	Десятичный разделитель или денежный десятичный разделитель
-	Число	Знак минус
,	Число	Разделитель групп
E	Число	Разделяет мантиссу и экспоненту в научной нотации. Должно быть заключено в кавычки в префиксе или суффиксе.
;	Граница подшаблона	Разделяет положительный и отрицательный подшаблоны.
%	Префикс или суффикс	Умножить на 100 и показать процент.
\u2030	Префикс или суффикс	Умножить на 1000 и показать как значение промилле.
¤ (\u00A4)	Префикс или суффикс	Знак валюты, замененный символом валюты. Если двойной, заменяет международный символ валюты. Если представлен в шаблоне, денежный десятичный разделитель используется вместо десятичного разделителя.
'	Префикс или суффикс	Используется для заключения в кавычки специальных символов в префиксе и суффиксе, например, "'#'#" форматирует 123 в "#123". Чтобы создать одну кавычку, используйте две в ряду: "# o''clock".

Научная форма

Числа в научной форме выражены как мантисса и степень, например, 1234 можно выразить как 1.234 x 10^3. Мантисса часто входит в диапазон 1.0 <= x < 10.0, хотя это и не требуется. Научная форма может быть передена в другой формат и подвергнута анализу научной формы только по шаблону; в настоящий момент нет генерирующего метода, который способен создать формат научной нотации. В шаблоне за символом экспоненты непосредственно следует один или более цифровых символов, и это является признаком научной формы. Пример: "0.###E0" форматирует число 1234 как "1.234E3".

- Количество цифровых символов после символа экпонента дает минимальное количество разрядов экспоненты. Максимума нет. Отрицательные экспоненты форматируются при помощи знака минус, не префикса и не суффикса из шаблона. Это разрешает такой шаблон, как например, "0.###E0 m/s".
- Минимальное и максимальное число целых чисел интерпретируются вместе:
- Если максимальное число целых чисел больше их минимального числа и больше, чем 1, это заставляет экспоненту быть множеством максимального числа целых чисел, пр этом минимальное количество целых чисел интерпретируется как 1. Наиболее частое использование этого - генерирование инженерной формы, в которой экспоненты - это мультипликатор трех, например, "##0.#####E0". Используя этот шаблон, число 12345 форматируется в "12.345E3", а 123456 форматируется в "123.456E3".
- В ином случае минимальное количество целых чисел получается путем корректировки экспоненты. Пример: 0.00123 форматируемое с "00.###E0," выдает "12.3E-4".
- Количество значащих разрядов в мантиссе - это сумма минимального целого числа и максимальных цифр после запятой, на нее не влияют максимальные целые числа. Например, 12345, форматируемое с "##0.##E0", - это "12.3E3". Чтобы показать все цифры, установите значимые цифры в нуль. Количество значимых цифр не влияет на синтаксический анализ.
- Экспоненциальные шаблоны могут не содержать разделители группировок.

Грамматика синтаксиса шаблона

Шаблоны формата числа имеют следующий синтаксис:

```
Pattern:
        PositivePattern
        PositivePattern ; NegativePattern
PositivePattern:
        Prefixopt Number Suffixopt
NegativePattern:
        Prefixopt Number Suffixopt
Prefix:
        any Unicode characters except \uFFFE, \uFFFF, and special characters
Suffix:
        any Unicode characters except \uFFFE, \uFFFF, and special characters
Number:
        Integer Exponentopt
        Integer . Fraction Exponentopt
Integer:
        MinimumInteger
        #
        # Integer
        # , Integer
MinimumInteger:
        0
        0 MinimumInteger
        0 , MinimumInteger
Fraction:
        MinimumFractionopt OptionalFractionopt
MinimumFraction:
        0 MinimumFractionopt
OptionalFraction:
        # OptionalFractionopt
Exponent:
        E MinimumExponent
MinimumExponent:
        0 MinimumExponentopt
```

20.9 Форматирование произвольных объектов

Это приложение объясняет, как можно форматировать произвольные объекты в строки. Описание разделено на две части. В первой части, Сводке, описываются основные концепции форматирования. Этот раздел предназначен для пользователей, которые хотят начать быстро и уже знакомы с языками форматирования печати и программирования (`printf()` на C или класса Java `Formatter`). Во второй части, Детали, представлены специфичные подробности реализации. Она предназначена для тех, кто хочет ознакомиться с более подробной спецификацией по форматированию поведения.

Сводка

Этот раздел предназначается, чтобы обеспечить краткий обзор форматирования понятий. Для точных поведенческих деталей обратитесь к разделу Деталей.

СТРОКОВЫЙ СИНТАКСИС ФОРМАТА

Каждый метод, который производит отформатированный вывод, требует строки формата и списка параметров. Строка формата является String который может содержать фиксированный текст и один или более встроенные спецификаторы формата. Рассмотрите следующий пример:

```
format("Duke's Birthday: %1$tm %1$te,%1$tY", d);
```

Эта строка формата является первым параметром format метод. Это содержит три спецификатора формата `%1$tm`, `%1$te` и `%1$tY`, которые указывают, как параметры должны быть обработаны и где они должны быть вставлены в текст. Остающиеся части строки формата являются фиксированным текстом включая "Dukes Birthday: " и любые другие пробелы или пунктуация. Список параметров состоит из всех параметров, которые передают к методу после строки формата. В вышеупомянутом примере список параметров имеет размер один и состоит из Дата `d`.

- У спецификаторов формата для общего, символа, и числовых типов есть следующий синтаксис:

  ```
  %[argument_index$][flags][width][.precision]conversion
  ```

  Дополнительный `argument_index` является десятичным целым числом, указывающим на позицию параметра в списке параметров. На первый параметр ссылаются `1$`, второе `2$` и т.д.

  Дополнительные `флаги` являются рядом символов, которые изменяют выходной формат. Набор допустимых флагов зависит от преобразования.

  Дополнительная `ширина` является неотрицательным десятичным целым числом, указывающим на минимальное число символов, которые будут записаны выводу.

  Дополнительная `точность` является неотрицательным десятичным целым числом, обычно используемым, чтобы ограничить число символов. Определённое поведение зависит от преобразования.

  Необходимое `преобразование` является символом, указывающим, как параметр должен быть отформатирован. Набор допустимых преобразований для данного параметра зависит от типа данных параметра.

- Спецификаторы формата для типов, которые представляют даты и времена, имеют следующий синтаксис:

  ```
  %[argument_index$][flags][width]conversion
  ```

  Дополнительные `argument_index`, `флаги` и `ширина` определяются, как выше.

  Необходимое `преобразование` является двумя символьными последовательностями. Первый символ `t` или `T`. Второй символ указывает на формат, который будет использоваться.

- У спецификаторов формата, которые не соответствуют параметрам, есть следующий синтаксис:

  ```
  %[flags][width]conversion
  ```

  Дополнительные `флаги` и `ширина` определяются, как выше.

  Необходимое `преобразование` является контентом указания символа, который будет вставлен в вывод.

ПРЕОБРАЗОВАНИЯ

Преобразования делятся на следующие категории:

1. **Общий** - может быть применён к любому типу параметра
2. **Символ** - может быть применён к основным типам, которые представляют символы Unicode
3. **Числовой**
   1. **Интеграл** - может быть применён к целочисленным типам Java: Целое или Длинное
   2. **Плавающая точка** - может быть применена к Java типы с плавающей точкой: Плавающая точка или Двойное
4. **Дата/Время** - может быть применена к типу Дата
5. **Процент** - производит литерал '%' ('\u0025')
6. **Разделитель строки** - производит специфичный для платформы разделитель строки

Следующая таблица суммирует поддерживаемые преобразования. Преобразования, обозначенные символом верхнего регистра (то есть. 'B', 'H', 'S', 'C', 'X', 'E', 'G', 'A', и 'T') то же самое как те для соответствующих строчных символов преобразования за исключением того, что результат преобразовывается в верхний регистр согласно правилам преобладания Locale.

Преобразование	Категория параметра	Описание
'b', 'B'	общий	Если аргумент параметра null, тогда результат false. Если аргумент является a boolean или Boolean, тогда результатом является строка. Иначе результатом является true.
'h', 'H'	общий	Если аргумент параметра null, тогда результат "null". Иначе результат получается путём конвертирования числа в шестнадцатеричную строку.
's', 'S'	общий	Если аргумент параметра null, тогда результат "null". Иначе аргумент конвертируется в строку.
'c', 'C'	символ	Результатом является символ Unicode
'd'	интеграл	Результат форматируется как десятичное целое число
'o'	интеграл	Результат форматируется как восьмеричное целое число
'x', 'X'	интеграл	Результат форматируется как шестнадцатеричное целое число
'e', 'E'	плавающая точка	Результат форматируется как десятичное число в компьютеризированном экспоненциальном представлении
'f'	плавающая точка	Результат форматируется как десятичное число
'g', 'G'	плавающая точка	Результат форматируется, используя компьютеризированное экспоненциальное представление или десятичный формат, в зависимости от точности и значения после округления.
'a', 'A'	плавающая точка	Результат форматируется как шестнадцатеричное число с плавающей точкой с мантиссой и экспонентой
't', 'T'	дата/время	Префикс для символов преобразования даты и времени.
'%'	процент	Результатом является литерал '%' ('\u0025') 25')
'n'	строковый разделитель	Результатом является специфичный для платформы разделитель строки

Любые символы, не явно определённые как преобразования, недопустимы и резервируются для будущих расширений.

ПРЕОБРАЗОВАНИЯ ДАТЫ/ВРЕМЕНИ

Следующая дата и символы суффикса преобразования времени определяются для 't' и 'T' преобразования.

Следующие символы преобразования используются для того, чтобы отформатировать время:

'H'	Час дня для 24-часовых часов, отформатированных как две цифры с начальным нулём по мере необходимости, то есть 00 - 23.
'I'	Час для 12-часовых часов, отформатированных как две цифры с начальным нулём по мере необходимости, то есть 01 - 12.
'k'	Час дня для 24-часовых часов, то есть 0 - 23.
'l'	Час для 12-часовых часов, то есть 1 - 12.
'M'	Минута в течение часа, отформатированного как две цифры с начальным нулём по мере необходимости, то есть 00 - 59.
'S'	Секунды в течение минуты, отформатированные как две цифры с начальным нулём по мере необходимости, то есть 00 - 60 ("60" - специальное значение, требуемое поддерживать секунды прыжка).
'L'	Миллисекунда в пределах второго, отформатированного как три цифры с начальными нулями по мере необходимости, то есть 000 - 999.

'N'	Наносекунда в пределах второго, отформатированного как девять цифр с начальными нулями по мере необходимости, то есть 000000000 - 999999999.
'p'	Специфичный для региональных настроек маркер утра или дня в нижнем регистре, например,"am" или "pm". Использование префикса преобразования 'T' силы этот вывод к верхнему регистру.
'z'	Стиль RFC 822 числовое смещение часового пояса от GMT, например. -0800.
'Z'	Строка, представляющая сокращение для часового пояса. Региональные настройки средства форматирования заменят региональные настройки параметра (если есть).
's'	Секунды с начала эпохи, начавшейся 1 января 1970 00:00:00 UTC, то есть минимальное подписанное 64-битовое целое значение, разделённое на 1000 до максимального подписанного 64-битового целого, разделённого на 1000.
'Q'	Миллисекунды с начала эпохи, запускающейся 1 января 1970 00:00:00 UTC, то есть минимальное подписанное 64-битовое целое значение, разделённое на 1000 до максимального подписанного 64-битового целого, разделённого на 1000.

Следующие символы преобразования используются для того, чтобы отформатировать даты:

'B'	Специфичное для региональных настроек полное имя месяца, например, "январь", "февраль".
'b'	Специфичное для региональных настроек сокращённое имя месяца, например, "янв", "фев".
'h'	То же самое, как 'b'.
'A'	Специфичное для региональных настроек полное название дня недели, например, "воскресенье", "понедельник".
'a'	Специфичное для региональных настроек краткое название дня недели, например, "вск", "пон"
'C'	Год из четырёх знаков, разделённый на 100, отформатированный как две цифры с начальным нулём по мере необходимости, то есть 00 - 99
'Y'	Год, отформатированный как по крайней мере четыре цифры с начальными нулями по мере необходимости, например. 0092 равняется 92 CE для Григорианского календаря.
'y'	Последние две цифры года, отформатированного с начальными нулями по мере необходимости, то есть 00 - 99.
'j'	День года, отформатированного как три цифры с начальными нулями по мере необходимости, например, 001 - 366 для Григорианского календаря.
'm'	Месяц, отформатированный как две цифры с начальными нулями по мере необходимости, то есть 01 - 13.
'd'	День месяца, отформатированного как две цифры с начальными нулями по мере необходимости, то есть 01 - 31
'e'	День месяца, отформатированного как две цифры, то есть 1 - 31.

Следующие символы преобразования используются для того, чтобы отформатировать общие составы даты/времени.

'R'	Время, отформатированное для 24-часовых часов как "%tH:%tM"
'T'	Время, отформатированное для 24-часовых часов как "%tH:%tM:%tS".
'r'	Время, отформатированное для 12-часовых часов как "%tI:%tM:%tS %Tp". Расположение маркера утра или дня ('%Tp') может быть зависимым от региональных настроек.
'D'	Дата, отформатированная как "%tm/%td/%ty".
'F'	ISO 8601 полная дата, отформатированная как "%tY-%tm-%td".
'c'	Дата и время, отформатированное как "%ta %tb %td %tT %tZ %tY", например. "Sun Jul 20 16:17:00 EDT 1969".

Любые символы, не явно определённые как суффиксы преобразования даты/времени, недопустимы и резервируются для будущих расширений.

ФЛАГИ

Следующая таблица суммирует поддерживаемые флаги. y означает, что флаг поддерживается для обозначенных типов параметра.

Флаг	Общий	Символ	Интеграл	Плавающая точка	Дата/Время	Описание
'-'	y	y	y	y	y	Результат будет выровнен слева по ширине.
'#'	y[1]	-	y[3]	y	-	Результат должен использовать зависимую от преобразования альтернативную форму
'+'	-	-	y[4]	y	-	Результат будет всегда включать знак
' '	-	-	y[4]	y	-	Результат будет включать ведущее пространство для положительных значений
'0'	-	-	y	y	-	Результат будет дополнен нулем
','	-	-	y[2]	y[5]	-	Результат будет включать специфичные для региональных настроек разделители группировки
'('	-	-	y[4]	y[5]	-	Результат заключит отрицательные числа в круглые скобки

[1] Зависит от определения.

[2] Только для 'd' преобразования.

[3] Только для преобразований 'o', 'x' и 'X'.

[4] Для 'd', применимого к Целому и Длинному.

[5] Только для преобразований 'e', 'E', 'f', 'g', and 'G'.

Любые символы, не явно определённые как флаги, недопустимы и резервируются для будущих расширений.

ШИРИНА

Ширина является минимальным числом символов, которые будут записаны в вывод. Для преобразования разделителя строки ширина не применима; если это обеспечено, то будет выдано исключение.

ТОЧНОСТЬ

Для общих типов параметра точность является максимальным количеством символов, которые будут записаны в вывод.

Для преобразований с плавающей точкой 'e', 'E', и 'f' точность является числом цифр после десятичного разделителя. Если преобразование 'g' или 'G', тогда точность является общим количеством цифр в получающейся величине после округления. Если преобразование 'a' или 'A', тогда точность не должна быть определена.

Для символа, интеграла и типов параметра даты/времени, процента и преобразований разделителя строки, точность не применима; если точность будет обеспечена, то будет выдано исключение.

ИНДЕКС ПАРАМЕТРА

Индекс параметра является десятичным целым числом, указывающим на позицию параметра в списке параметров. На первый параметр ссылаются "1$", на второй - "2$" и т.д.

Другой способ сослаться на параметры позицией состоит в том, чтобы использовать флаг '<' ('\u003c'), который заставляет снова использовать параметр предыдущий спецификатора формата. Например, следующие два оператора произвели бы идентичные строки:

- `format("Duke's Birthday: %1$tm %1$te,%1$tY", d)`
- `format("Duke's Birthday: %1$tm %<te,%<tY", d)`

Детали

Этот раздел предназначается, чтобы обеспечить поведенческие детали для того, чтобы они отформатировали, включая условия и исключения, поддерживаемые типы данных, локализацию, и взаимодействия между флагами, преобразованиями, и типами данных. Для краткого обзора форматирования понятий сошлитесь на Сводку.

Любые символы, не явно определенные как преобразования, суффиксы преобразования даты/времени, или флаги, недопустимы и резервируются для будущих расширений.

Если спецификатор формата содержит ширину или точность с недопустимым значением или который иначе неподдерживается, то форматирование не будет произведено.

Если спецификатор формата содержит символ преобразования, который не применим к соответствующему параметру, то форматирование не будет произведено.

ОБЩЕЕ

Следующие общие преобразования могут быть применены к любому типу параметра:

'b'	'\u0062'	Производит либо "true", либо "false". Если параметр null, тогда результат "false". Иначе результат "true".
'B'	'\u0042'	Прописная разновидность 'b'.
'h'	'\u0068'	Производит строку, представляющую значение хэш-кода объекта.
'H'	'\u0048'	Прописная разновидность 'h'.
's'	'\u0073'	Производит строку. Если параметр null, тогда результат "null". Иначе параметр конвертируется в Строку.
'S'	'\u0053'	Прописная разновидность 's'.

Следующие флаги применяются к общим преобразованиям:

'-'	'\u002d'	Левый знак выравнивает по ширине вывод. Пробелы ('\u0020') будут добавлены в конце преобразованного значения, как требуется, чтобы заполнить минимальную ширину поля. Если ширина не предоставляется, то форматрирование не будет выполнено. Если этот флаг не дан, то вывод будет выровнен по правому знаку.
'#'	'\u0023'	Требует, чтобы вывод использовал альтернативную форму. Определение формы задается преобразованием.

Ширина является минимальным числом символов, которые будут записаны в вывод. Если длина преобразованного значения будет меньше, чем ширина тогда, то вывод будет дополнен ' ' ('\u0020'), пока общее количество символов не будет равняться ширине. Дополнение слева по умолчанию. Если дается флаг '-', тогда дополнение будет справа. Если ширина не определяется, тогда нет никакого минимума.

Точность является максимальным количеством символов, которые будут записаны в вывод. Точность применяется перед шириной, таким образом вывод будет усеченным до символов точности, даже если ширина больше, чем точность. Если точность не определяется, тогда нет никакого явного лимита по числу символов.

ЧИСЛОВОЙ

Числовые преобразования делятся на следующие категории:

1. Целочисленный и Длинный

2. Плавающая точка и Двойной

Числовые типы будут отформатированы согласно следующему алгоритму:

Алгоритм локализации чисел

После того, как цифры получаются для целой части, дробной части, и экспоненты (соответствующие для типа данных), применяется следующее преобразование:

1. Каждый символ цифры d в строке заменяется специфичной для локали цифрой, вычисленной относительно нулевой цифры z текущей локали; это - d - '0' + z.

2. Если десятичный разделитель присутствует, специфичным для локали десятичный разделитель заменяется.

3. Если дается флаг ',' ('\u002c'), тогда вставляется специфичный для локали разделитель группировки, сканируя целую часть строки от менее значимых до более значимых цифр и вставляя разделитель с промежутками, определенными размером группировки локали.

4. Если дается флаг '0', тогда специфичные для локали нулевые цифры вставляются после символа знака, если таковые вообще имеются, и перед первой ненулевой цифрой, пока длина строки не равна требуемой ширине поля.

5. Если значение отрицательно и дается флаг '(', тогда а '(' ('\u0028') добавляется к началу и а ')' ('\u0029') добавляется к концу.

6. Если значение отрицательно (или отрицательный нуль с плавающей точкой) и не дается флаг '(', тогда а '-' ('\u002d') добавляется к началу.

7. Если дается флаг '+', и значение положительное или нуль (или положительный нуль с плавающей точкой), тогда а '+' ('\u002b') будет добавляться к концу.

Если значением будет NaN или положительная бесконечность, литеральные строки "NaN" или "Бесконечность" соответственно, то будет вывод. Если значение будет отрицательной бесконечностью, то вывод будет "(Бесконечность)", если дается флаг '(', иначе вывод будет "-Бесконечность". Эти значения не локализуются.

ЦЕЛОЧИСЛЕННОЕ И ДЛИННОЕ

Следующие преобразования можно применить к Целочисленным и Длинным значениям.

'd'	'\u0054'	Форматирует параметр как десятичное целое число. Применяется алгоритм локализации. Если '0' флаг дается, и значение отрицательно, тогда нулевое дополнение произойдет после знака.
'o'	'\u006f'	Форматирует параметр как целое число в основе восемь. Никакая локализация не применяется. Если x будет отрицателен, тогда результатом будет значение без знака, сгенерированное путем добавления 2n к значению, где n - число битов в типе, как возвращено статическим полем SIZE в классы Целочисленное или Длинное соответственно. Если дается флаг '#', тогда вывод будет всегда начинаться с индикатора основания '0'. Если дается флаг '0', вывод будет дополнен начальными нулями к ширине поля после любой индикации относительно знака.
'x'	'\u0078'	Форматирует параметр как целое число в основе шестнадцать. Никакая локализация не применяется. Если x будет отрицателен, тогда результатом будет значение без знака, сгенерированное путем добавления 2n к значению, где n - число битов в типе, как возвращено статическим полем SIZE в классы Целочисленное или Длинное соответственно. Если дается флаг '#', тогда вывод будет всегда начинаться с индикатора основания "0x". Если дается флаг '0', вывод будет дополнен начальными нулями к ширине поля после индикатора основания или знака (если есть).
'X'	'\u0058'	Прописная разновидность 'x'. Вся строка, представляющая число, будет преобразована в верхний регистр, включая 'x' (если есть) и все шестнадцатеричные цифры 'a' - 'f' ('\u0061' - '\u0066').

Если преобразование 'o', 'x', или 'X' и даются оба флага '#' и '0', тогда результат будет содержать индикатор основания ('0' для восьмеричного и "0x" или "0X" для шестнадцатеричного), некоторое число нулей (на базе ширины) и значение.

Если флаг '-' не дается, тогда дополнение пространства произойдет перед знаком.

Следующие флаги применяются к числовым интегральным преобразованиям:

'+'	'\u002b'	Требует, чтобы вывод включал положительный знак для всех положительных чисел. Если этот флаг не будет дан, тогда только отрицательные величины будут включать знак.
' '	'\u0020'	Требует, чтобы вывод включал единственное дополнительное пространство ('\u0020') для неотрицательных значений.
'0'	'\u0030'	Требует, чтобы вывод был дополнен начальными нулями к минимальной ширине поля после любого знака или индикатора основания, кроме случаев преобразовывая NaN или бесконечности.
','	'\u002c'	Требует, чтобы вывод включал специфичные для локали разделители группы, как описано в разделе "группа" алгоритма локализации.

'('	'\u0028'	Требует, чтобы вывод добавлял к к отрицательным значениям '(' ('\u0028') в начале и а ')' ('\u0029') в конце.

Если никакие флаги не даются, форматирование по умолчанию следующие:

- Вывод выровнен по правому знаку в пределах ширины
- Отрицательные числа начинаются с а '-' ('\u002d')
- Положительные числа и нуль не включают знак или дополнительное ведущее пространство
- Разделители группировки не включаются

Ширина является минимальным числом символов, которые будут записаны в вывод. Это включает любые знаки, цифры, разделители группировки, индикатор основания и круглые скобки. Если длина преобразованного значения меньше, чем ширина, тогда вывод будет дополнен пробелами ('\u0020'), чтобы общее количество символов равнялось ширине. Дополнение находится слева по умолчанию. Если дается флаг '-', тогда дополнение будет справа. Если ширина не определена, тогда нет никакого минимума.

Точность не применима.

ПЛАВАЮЩАЯ ТОЧКА И ДВОЙНОЕ

Следующие преобразованиям можно применить к значениям Плавающая точка и Двойное.

'e'	'\u0065'	Требует, чтобы вывод был отформатирован, используя компьютеризированное экспоненциальное представление. Применяется алгоритм локализации. Форматирование величины m зависит от ее значения. Если m - это NaN или бесконечное, то будут выведены литеральные строки "NaN" или "Бесконечность" соответственно. Эти значения не локализуются. Если m - положительный нуль или отрицательный нуль, то экспонент будет "+00". Иначе результатом является строка, которая представляет знак и величину (абсолютное значение) параметра. Форматирование знака описывается в алгоритме локализации. Форматирование величины m зависит от ее значения. Пусть n быть уникальным целым числом так, что 10n <= м. <10n+1; тогда пусть оно будет математически точным частным m и 10n так, чтобы 1 <= <10. Величина тогда представляется как целая часть а, как единственная десятичная цифра со стоящим после нее десятичным разделителем и десятичными цифрами, представляющими дробную часть а, после которых стоит символ экспонента 'e' ('\u0065'), знак экспонента и представлением n как десятичного целого числа, дополненного нулем, чтобы включать по крайней мере две цифры. Число цифр в результате дробной части m или а равно точности. Если точность не определяется? тогда значение по умолчанию - 6. Если точность меньше количества цифр, которые появились бы после десятичной точки в строке, тогда значение округляется, используя круглую половину алгоритма. Иначе могут быть добавлены нули, чтобы достигнуть точности.
'E'	'\u0045'	Прописная разновидность 'e'. Символ экспоненты будет 'E' ('\u0045').
'g'	'\u0067'	Требует, чтобы вывод был отформатирован в общем экспоненциальном представлении, как описано ниже. Применяется алгоритм локализации. После округления для точности, форматирование получающейся величины m зависит от его значения. Если m больше или равно 10-4, но меньше 10precision, тогда оно представляется в десятичном формате. Если m меньше 10-4 или больше или равно 10precision, тогда оно представляется в компьютеризированном экспоненциальном представлении. Общее количество существенных цифр в m равно точности. Если точность не определена, то значение по умолчанию - 6. Если точность 0, тогда берется 1.
'G'	'\u0047'	Прописная разновидность 'g'.
'f'	'\u0066'	Требует, чтобы вывод был отформатирован, используя десятичный формат. Применяется алгоритм локализации. Результатом является строка, которая представляет знак и величину (абсолютное значение) параметра. Форматирование знака описывается в алгоритме локализации. Форматирование величины m зависит от его значения.

		Если m - NaN или бесконечное, будут выведены литеральные строки "NaN" или "Бесконечность" соответственно. Эти значения не локализуются. Величина форматируется, поскольку после целой части m, без ведущих нулей, стоят десятичный разделитель, одна или более десятичных цифр, представляющими дробную часть m. Число цифр в результате дробной части m или a равно точности. Если точность не определяется, тогда значение по умолчанию - 6. Если точность меньше числа цифр, которые появились бы после десятичной точки в строке, тогда значение округляется, используя круглую половину алгоритма. Иначе могут быть добавлены нули, чтобы достигнуть точности.
'a'	'\u0061'	Требует, чтобы вывод был отформатирован в шестнадцатеричной экспоненциальной форме. Никакая локализация не применяется. Результатом является строка, которая представляет знак и величину (абсолютное значение) параметра x. Если x будет отрицателен или иметь отрицательное нулевое значение, тогда результат начнется с '-' ('\u002d'). Если x будет положителен или иметь положительное нулевое значение и дается флаг '+', тогда результат начнется с '+' ('\u002b'). Форматирование величины m зависит от ее значения. • Если значением будет NaN или бесконечность, то будут выведены литеральные строки "NaN" или "Бесконечность", соответственно. • Если m является нулем, тогда оно представляется строкой "0x0.0p0". • Если m является двойным значение с нормализованным представлением, тогда используются подстроки, чтобы представить поля экспоненты и мантисса. Мантисса представляется символами "0x1." с шестнадцатеричным представлением остальной части мантиссы как дроби. Экспонента представляется 'p' ('\u0070') со стоящей после нее десятичной строкой несмещенной экспоненты, как будто произведенной вызывающей Integer.toString на значении экспоненты. • Если m является двойным значение с субнормальным представлением, тогда мантисса представляется символами '0x0.', после которых стоит шестнадцатеричное представление остальной части мантиссы как дроби. Экспонента представляется 'p-1022'. Отметьте, что должна быть по крайней мере одна ненулевая цифра в субнормальной мантиссе.
'A'	'\u0041'	Прописная разновидность 'a'. Вся строка, представляющая число, будет преобразована в верхний регистр, включая 'x' ('\u0078') и 'p' ('\u0070' и все шестнадцатеричные цифры 'a' - 'f' ('\u0061' - '\u0066').

Применяются все флаги, определенные для Целого и Длинного числа.

Если дается флаг '#', тогда будет всегда присутствовать десятичный разделитель.

Если не дается никаких флагов, форматирование по умолчанию следующие:

- Вывод выровнен по правому знаку в пределах ширины
- Отрицательные числа начинаются с a '-'
- Положительные числа и положительный нуль не включают знак или дополнительное ведущее пространство
- Разделители группировки не включаются
- Десятичный разделитель появится, только если цифра будет следовать за ним

Ширина является минимальным числом символов, которые будут записаны в вывод. Она включает любые знаки, цифры, групповые разделители, десятичные разделители, экспоненциальный символ, индикатор основания, круглые скобки и строки, представляющие бесконечность и NaN как применимые. Если длина преобразованного значения будет меньше, чем ширина, тогда вывод будет дополнен пробелами ('\u0020'), пока общее количество символов не будет равняться ширине. Дополнение находится слева по умолчанию. Если дается флаг '-', тогда дополнение будет справа. Если ширина не определяется, тогда нет никакого минимума.

Если преобразования - 'e', 'E' или 'f', тогда точность является числом цифр после десятичного разделителя. Если точность не определена, то она условно равна 6.

Если преобразования - 'g' или 'G', тогда точность является общим количеством существенных цифр, присутствующих в величине после округления. Если точность не определена, то значение по умолчанию - 6. Если точность 0, тогда оно равно 1.

Если преобразования - 'a' или 'A', тогда точность является количеством шестнадцатеричных цифр после десятичного разделителя. Если точность не обеспечивается, то будут выведены все цифры.

ДАТА/ВРЕМЯ

Это преобразование можно применить к Длинному и Дате.

't'	'\u0074'	Префикс для даты и символов преобразования времени.
'T'	'\u0054'	Прописная разновидность 't'.

Следующие суффиксы символов преобразования времени определяются для преобразования 't' и 'T'.

Следующие символы преобразования используются для того, чтобы отформатировать время:

'H'	'\u0048'	Час дня для 24-часовых часов, отформатированный как две цифры с начальным нулем при необходимости, то есть 00 - 23. 00 соответствует полуночи.
'I'	'\u0049'	Час для 12-часовых часов, отформатированных как две цифры с начальным нулем при необходимости, то есть 01 - 12. 01 соответствует одному часу (утро или день).
'k'	'\u006b'	Час дня для 24-часовых часов, то есть 0 - 23. 0 соответствует полуночи.
'l'	'\u006c'	Час для 12-часовых часов, то есть 1 - 12. 1 соответствует одному часу (или утро или день).
'M'	'\u004d'	Минута в течение часа, отформатированного как две цифры с начальным нулем при необходимости, то есть 00 - 59.
'S'	'\u0053'	Секунды в течение минуты, отформатированные как две цифры с начальным нулем при необходимости, то есть 00 - 60 ("60" -специальное значение, требуемое для поддержки потерянной секунды).
'L'	'\u004c'	Миллисекунда в пределах секунды, отформатированная как три цифры с начальными нулями при необходимости, то есть 000 - 999.
'N'	'\u004e'	Наносекунда в пределах секунды, отформатированная как девять цифр с начальными нулями при необходимости, то есть 000000000 - 999999999. Точность этого значения ограничивается разрешением базовой операционной системы или аппаратных средств.
'p'	'\u0070'	Специфичный для локали маркер утра или дня в нижнем регистре, например, "am" или "pm". Использование префикса преобразования 'T' переводит этот вывод в верхний регистр.
'z'	'\u007a'	Числовое смещение часового пояса стиля RFC 822 от GMT, например 0800.
'Z'	'\u005a'	Строка, представляющая аббревиатуру часового пояса.
's'	'\u0073'	Секунды с начала эпохи, начинающейся 1 января 1970 00:00:00 UTC, то есть минимальное подписанное 64-битовое целое число, разделенное на 1000, до максимального подписанного 64-битового целого числа, разделенного на 1000.
'Q'	'\u004f'	Миллисекунды с начала эпохи, начинающейся 1 января 1970 00:00:00 UTC, то есть минимальное подписанное 64-битовое целое число, разделенное на 1000, до максимального подписанного 64-битового целого числа, разделенного на 1000. Точность этого значения ограничивается разрешением базовой операционной системы или аппаратных средств.

Следующие символы преобразования используются для того, чтобы отформатировать даты:

'B'	'\u0042'	Специфичное для локали полное имя месяца, например, "январь", "февраль".
'b'	'\u0062'	Специфичное для локали сокращенное имя месяца, например, "янв", "фев".
'h'	'\u0068'	То же самое, как 'b'.
'A'	'\u0041'	Специфичное для локали полное имя дня недели, например, "воскресенье", "понедельник"
'a'	'\u0061'	Специфичное для локали краткое название дня недели, например, "вск", "пон"
'C'	'\u0043'	Год из четырех знаков, разделенный на 100, отформатированный как две цифры с начальным нулем при необходимости, то есть 00 - 99
'Y'	'\u0059'	Год, отформатированный по крайней мере к четырем цифрам с начальными нулями при необходимости, например, 0092 равняется 92 CE по Григорианскому календарю.

'y'	'\u0079'	Последние две цифры года, отформатированного с начальными нулями при необходимости, то есть 00 - 99.
'j'	'\u006a'	День года, отформатированного как три цифры с начальными нулями при необходимости, например 001 - 366 по Григорианскому календарю. 001 соответствует первому дню года.
'm'	'\u006d'	Месяц, отформатированный как две цифры с начальными нулями при необходимости, то есть 01 - 13, где "01" - первый месяц года и "13" - специальное значение, требуемое для поддержки лунных календарей.
'd'	'\u0064'	День месяца, отформатированного как две цифры с начальными нулями при необходимости, то есть 01 - 31, где "01" - первый день месяца.
'e'	'\u0065'	День месяца, отформатированного как две цифры, то есть 1 - 31, где "1" - первый день месяца.

Следующие символы преобразования используются для того, чтобы отформатировать общие составы даты/времени.

'R'	'\u0052'	Время, отформатированное для 24-часовых часов как "%tH:%tM"
'T'	'\u0054'	Время, отформатированное для 24-часовых часов как "%tH:%tM:%tS".
'r'	'\u0072'	Время, отформатированное для 12-часовых часов как "%tI:%tM:%tS %Tp". Расположение маркера утра или дня ('%Tp') может быть зависимым от локали.
'D'	'\u0044'	Дата, отформатированная как "%tm/%td/%ty".
'F'	'\u0046'	Полная дата ISO 8601, отформатированная как "%tY-%tm-%td".
'c'	'\u0063'	Дата и время, отформатированное как "%ta %tb %td %tT %tZ %tY", например "Sun Jul 20 16:17:00 EDT 1969".

Применяется флаг '-', определенный для Общих преобразований. Если дается флаг '#', тогда форматирование не выполняется.

Ширина является минимальным количеством символов, которые будут записаны в вывод. Если длина преобразованного значения меньше, чем ширина, тогда вывод будет дополнен пробелами ('\u0020'), пока общее количество символов не будет равняться ширине. Дополнение слева по умолчанию. Если дается флаг '-', тогда дополнение будет справа. Если ширина не определяется, тогда нет никакого минимума.

Точность не применима. Если точность определяется, тогда форматирование не выполняется.

ПРОЦЕНТ

Преобразование не соответствует никакому параметру.

'%'	Результатом является литерал '%' ('\u0025') Ширина является минимальным числом символов, которые будут записаны в вывод, включая '%'. Если длина преобразованного значения меньше, чем ширина, тогда вывод будет дополнен пробелами ('\u0020'), пока общее количество символов не будет равняться ширине. Дополнение слева. Если ширина не определяется, тогда выводится только '%'. Применяется флаг '-', определенный для Общих преобразований, применяется. Точность не применима.

РАЗДЕЛИТЕЛЬ СТРОКИ

Преобразование не соответствует никакому параметру.

'n'	специфичный для платформы разделитель строки

Флаги, ширина и точность не применимы.

ИНДЕКС ПАРАМЕТРА

Спецификаторы формата могут сослаться на параметры тремя способами:

- Явная индексация используется, когда спецификатор формата содержит индекс параметра. Индекс параметра является десятичным целым числом, указывающим на позицию параметра в списке параметров. На первый параметр ссылаеются "1$", на второй - "2$" и т.д. На параметр можно сослаться не раз.

Например:

```
format("%4$s %3$s %2$s %1$s %4$s %3$s %2$s %1$s", "a", "b", "c", "d")
```

имеет результатом

```
d c b a d c b a
```

Relative indexing is used when the format specifier contains a '<' ('\u003c') flag which causes the argument for the previous format specifier to be re-used. If there is no previous argument, then formatting will fail. Относительная индексация используется, когда спецификатор формата содержит флвг а '<' ('\u003c'), который заставляет снова использовать параметр предыдущего спецификатора формата. Если нет никакого предыдущего параметра, то форматирование не выполняется.

```
format("%s %s %<s %<s", "a", "b", "c", "d")
```

имеет результатом

```
a b b b
```

Параметры "c" and "d" игнорируются, потому что на них не ссылаются.

- Обычная индексация используется, когда спецификатор формата не содержит ни индекса параметра, ни флага а '<'. Каждый спецификатор формата, который использует обычную индексацию, получает последовательный неявный индекс в списке параметров, который не зависит от индексов, используемых явной или относительной индексацией.

```
format("%s %s %s %s", "a", "b", "c", "d")
```

имеет результатом

```
a b c d
```

Возможно иметь строку формата, которая использует все формы индексации, например:

```
format("%2$s %s %<s %<s", "a", "b", "c", "d")
```

имеет результатом

```
b a a b
```

Параметры "c" и "d" игнорируются. потому что на них не ссылаются.

Если индекс параметра не соответсвует существующим параметрам, форматирование не выполняется.

Если параметров больше, чем спецификаторов формата, дополнительные параметры игнорируются.

20.10 Общие константы

Это приложение предлагает список и описывает общие числовые и строковые константы, используемые в различных модулях AtomMind.

Единицы времени

- 0 - Миллисекунда
- 1 - Секунда
- 2 - Минута
- 3 - Час
- 4 - День
- 5 - Неделя
- 6 - Месяц
- 7 - Квартал
- 8 - Год

Типы агрегации

- 0 - Среднее
- 1 - Минимальное
- 2 - Максимальное
- 3 - Суммирование
- 4 - Первое
- 5 - Последнее

20.11 Опции экспорта/импорта CSV

При экспорте/импорте из файла значений, разделенных символами (Character Separated Values) (CSV), Вы можете задать несколько настроек кодирования/декодирования:

Настройка	**Описание**
Разделитель полей	Символ, используемый в качестве разделителя полей. По умолчанию - точка с запятой (;).
Использовать текстовый квалификатор	Устанавливает, будут ли текстовые квалификаторы использоваться во время записи/синтаксического анализа или нет. При экспорте данных есть три возможности: Не использовать, Использовать, когда это необходимо (т.е. когда текст сам содержит ограничители полей или другие зарезервированные символы) и Использовать всегда.
Текстовый квалификатор	Символ должен использоваться как текстовый квалификатор в данных. По умолчанию двойные кавычки (").
Режим экранирования	Есть два режима, которые можно использовать, чтобы перейти от символов квалификатора к его отображению в виде данных: • Использовать символ обратная косая черта перед текстовым квалификатором для отображения текстового квалификтора; • Удвоить текстовый квалификатор для отображения текстового квалификатора.
Символ комментария	Символ используется как сигнал о комментарии.
Запись заголовка	Определяет содержание первой записи CSV-файла. Возможны следующие варианты: • **Отсутствует**. У CSV-файла нет записи заголовка, первый заголовок содержит сырые данные. В случае импортирования это означает, что данные в первой колонке CSV-файла будет импортирован в первую запись Таблицы данных и пр. • **Имена полей**. Первая запись CSV-файла содержит имена полей Таблицы данных. При импортировании помогает соотнести колонку CSV-файла и поля Таблицы данных. • **Описания полей**. Первая запись CSV-файла содержит описания полей Таблицы данных. При импортировании помогает соотнести колонку CSV-файла и поле Таблицы данных. • **Игнорировать как не представляющие ценность данные**. Эта опция доступна лишь во время операции импортирования. Заставляет операцию импортирования пропускать первую запись Таблицы данных и т.д.
Экспортировать значения значений вместо значений	Определяет, что записывается в файл CSV, если у поля есть Значения выбора 51: • **Актуальные значения**, если эта опция отключена • **Описание значений**, если эта опция включена

20.12 Снятие дампов памяти (heap) и потоков

В некоторых очень редких случаях AtomMind Server или AtomMind Client могут перестать отвечать из-за потери доступа к памяти, 100% нагрузки на процессор или блокировки потоков. Такие ситуации наиболее вероятно могут произойти в версиях **Предпросмотра** AtomMind.

В большинстве случаев инженеры ТВЭЛ решают такие вопросы через удалённое соединение с машиной, на которой запущен AtomMind. Однако если удаленный доступ ограничен, например, из-за соображений

безопасности, клиентам приходится осуществлять *снятие дампов потоков и памяти (heap)*, чтобы передать их ТВЭЛ для последующего анализа.

Создание отслеживания стека зависшего / неотвечающего AtomMind Server или AtomMind Client

Для создания дампа потоков, используемого для диагностирования неэффективной блокировки потоков и их зависания:

1. Проверьте версию Java Runtime Environment (JRE), используемую для работы AtomMind при помощи `%AtomMind Server Folder% /jre/bin/java.exe -version`
2. Установите комплект разработчика Java (JDK) той же версии, что JRE, используемая на AtomMind
3. Выясните идентификатор Процесса (PID) процесса AtomMind:
   3.1. Используя Task Manager в Windows
   3.2. Путём запуска `ps ax|grep java` на Linux
4. Запустите `%JDK Folder% /bin/jstack -l PID > stack.txt`
5. Отправьте конечный файл `stack.txt` сотрудникам AtomMind для анализа

Создание дампа памяти (heap) AtomMind Server или AtomMind Client с высокой загрузкой памяти Java

Примечание 1: Использование памяти AtomMind Server можно отследить при помощи Датчика использования памяти сервера 2185.

Примечание 2: Использование памяти AtomMind Client можно отследить при помощи утилиты JVisualVM, которая входит в комплект разработчика Java (JDK).

Примечание 3: Утечку памяти можно успешно определить, если использование памяти сервера постоянно росло в течение нескольких дней и выросло до 200-300% начального использования памяти, зарегистрированного после нескольких запусков сервера.

Примечание 4: Создание графика истории Датчик использования памяти сервера 2185 поможет ясно увидеть растущее использование памяти.

Для создания дампа памяти, который используется для диагностирования её утечек:

1. Установите комплект разработчика Java (JDK) той же версии, что JRE, используемая на AtomMind
2. Выясните идентификатор Процесса (PID) процесса AtomMind:
   2.1. Используя Task Manager в Windows
   2.2. Путём запуска `ps ax|grep java` на Linux
3. Запустите команду `jmap -dump:format=b,file=dump.bin PID`
4. Сожмите конечный файл `dump.bin` и перешлите сотрудникам AtomMind для анализа

20.13 Настройка DCOM для удаленного доступа

Прежде чем получить доступ к Вашему WMI-хосту или OPC-серверу, запущенному на ПК с Microsoft Windows, следует убедиться, что распределённая объектная модель компонентов (Distributed COM (DCOM)) настроена должным образом на данном ПК. Следуйте контрольному перечню этапов, чтобы правильно выполнить настройку.

1. Запуск необходимых сервисов

Пожалуйста, убедитесь, что сервисы **Server** и **RemoteRegistry** запущены на ПК с COM-сервером.

2. Настройка прав доступа

Чтобы избежать ошибочной ситуации, когда Доступ отклонён, лучше подключиться к COM-серверу под идентификатором зарегистрированного в настоящий момент пользователя. Если разрешение прав доступа уровня "администратора" получить сложно, можно создать локального пользователя в группе "Пользователи".

2A. НАСТРОЙКА ДОСТУПА К DCOM

Перейдите в Панель Управления (**Control Panel)>** Администрирование **(Administrative Tools) >** Локальная политика безопасности (**Local Security Policy) >** Настройки безопасности (Security Settings) > Локальная политика (Local Policies) > Опции безопасности **(Security Options)**:

- Дважды щёлкните мышью по **DCOM: Политика ограничения прав доступа** к ПК (Machine Access Restrictions policy), кликните **Редактировать безопасность** (click Edit Security), добавьте пользователя, созданного ранее, (или авторизовавшегося в настоящий момент пользователя), разрешите "Удаленный доступ".
- Дважды щёлкните мышью по **DCOM**: **Политика ограничений для запуска** ПК (Machine Launch Restrictions policy), щёлкните **Редактировать безопасность** (Edit Security), добавьте пользователя, созданного ранее, (или авторизовавшегося в настоящий момент пользователя), разрешите "Локальный запуск", "Удаленный запуск", "Локальная активация, "Удалённая активация".
- Дважды щёлкните по элементу **Доступ по сети** (Network access): **Модель защиты и совместного пользования для политики локальных учетных записей** (Sharing and security model for local accounts policy), выберите элемент **Классическая модель (Classic) - пользователь авторизуется под своим именем.**

2B. НАСТРОЙКА БЕЗОПАСНОСТИ COM

Перейдите в **панель управления** (Control Panel) > **Администрирование** (Administrative Tools) > **Служебные компоненты** (Component Services) > **Компьютеры** (Computers) > щёлкните правой кнопкой мыши по элементу **Мой компьютер** (My Computer) > щёлкните по элементе **Свойства** (Properties) > вкладка **Параметры по умолчанию** (Default Settings):

- проверьте "Активировать распределённую COM на этом компьютере"
- установите уровень авторизации для подключения по умолчанию
- установите Уровень имперсонализации для идентификации по умолчанию

Перейдите в **панель управления** (Control Panel) > **Администрирование** (Administrative Tools) > **Служебные компоненты** (Component Services) > **Компьютеры** (Computers) > щёлкните правой кнопкой мыши по элементу **Мой Компьютер** (My Computer) > щёлкните **Свойства** (Properties) > щёлкните по вкладке **Безопасность COM** (COM Security):

- А разделе Права доступа (**Access Permissions),** щёлкните Редактировать по умолчанию (**Edit Default)** > добавьте пользователя, созданного ранее, (или авторизовавшегося в данный момент пользователя), разрешите "Удаленный доступ (**"Remote Access**")
- В разделе Права доступа к запуску и активации (**Launch and Activation Permissions)** > щёлкните Редактировать по умолчанию (**Edit Default)** > добавьте пользователя, созданного ранее (или авторизовавшегося в данный момент пользователя), разрешите "Локальный запуск" (**Local Launch**), "Удаленный запуск" (**Remote Launch)**, "Локальная активация" (**Local Activation)**, "Удалённая активация" (**Remote Activation).**

Что касается раздела Служебные компоненты (Component Services), Вы можете перейти к определённому компоненту и дать разрешение оттуда, а не из элемента "Мой компьютер" (My Computer), что является общим разрешением (blanket grant)

3. Настройка Windows Firewall

Иногда Windows Firewall будет действовать некорректно, если его неверно настроить, поэтому, пожалуйста, убедитесь, что Вы настроили его для протокола DCOM или отключили его. Следует также убедиться, что должная настройка firewall позволит предотвратить неполадки приложений DCOM Windows.

4. Отключение UAC

Когда Управление доступом пользователя (User Access Control (UAC)) активно, у учетной записи администратора есть два токена безопасности, обычный токен пользователя и административный токен (который активируется лишь, когда Вы пройдёте процедуру UAC). К сожалению, удалённые запросы, которые поступают по сети, дают администратору обычный токен пользователя, и поскольку нет возможности обрабатывать процедуру UAC удалённо, токен нельзя повысить до токена уровня администратора.

Т.о., UAC следует отключить, чтобы разрешить удаленный доступ к DCOM.

5. Установка последних обновлений

Пожалуйста, убедитесь, что Ваш ПК с Windows (где развёрнут COM-сервер) имеет все пакеты обновления Microsoft. Многие проблемы возникают из-за неправильной конфигурации ПК.

Примечания для особых версий Windows

Если перечисленные инструкции не помогли, обратитесь к примечаниям для особых версий Windows, расположенных ниже.

НАСТРОЙКА DCOM НА WINDOWS 2000

1. Кликните Старт (Start), Запустить (Run), а затем наберите DCOMCNFG.

2. Кликните Свойства по умолчанию (Default Properties). Выберите Активировать Распределённую COM на этом компьютере. Установите Уровень авторизации для подключения по умолчанию (вариант None тоже подходит). Установите уровень Имперсонализации для идентификации по умолчанию (вариант Impersonate тоже подходит).

3. Кликните по элементу Безопасность по умолчанию.

4. Под элементом Права доступа по умолчанию (Default Access Permissions) кликните Редактировать (Edit Default). Добавьте SYSTEM и INTERACTIVE. Пользователь, чьи учётные данные будут использоваться для доступа к COM-приложению, должны быть тоже включены в этот список. Существует много способов, как это сделать. Вы можете добавить специального пользователя или просто добавить группу, к которой принадлежит пользователь. Возможные значения включают:

- Домен\Имя пользователя (специальный пользователь)

- Домен\Администраторы (Все администраторы на специальном домене)

- Все (Все пользователи)

5. Под элементом Права доступа к запуску по умолчанию (Default Launch Permissions) кликните Редактировать по умолчанию (Edit Default). Убедитесь, что Права доступа к запуску имеют те же значения, что и Права доступа по умолчанию (Default Access Permissions).

6. Кликните Протоколы по умолчанию (Default Protocols). Убедитесь, что ориентированный на подключение TCP/IP перечисляется в начале.

7. Теперь Вы должны настроить приложение COM, к которому необходимо получить доступ. Кликните на приложении, а затем кликните правой кнопкой мыши по приложению, которое необходимо настроить. Выберите Свойства. Если Ваше приложение является библиотекой, следует сначала создать для замены EXE, используя инструментарий SetDllHost. После создания EXE, его имя появится в списке приложений. Выберите Свойства для него и продолжите действия.

8. Кликните Общее (General). Установите Уровень авторизации по умолчанию.

9. Кликните Расположение (Location). Выберите Запустить приложение на этом компьютере (Run application on this computer).

10. Кликните по элементу Безопасность (Security). Выберите Использовать права доступа по умолчанию (Use default access permissions) и используйте Права доступа для запуска по умолчанию (default launch permissions).

11. Кликните по элементу Идентичность (Identity). Выберите Запускающего пользователя (launching user). Этот параметр задает учётную запись, которая будет использоваться для запуска COM после того, как она запущено программой клиента. Запускающий пользователь - это учетная пользовательская запись процесса клиента, который запустил сервер, и это рекомендуемая настройка. В зависимости от приложения COM, к которому необходимо подключиться, Вам, возможно, потребуется изменить его на:

- интерактивного пользователя (interactive user) - пользователь, который в настоящий момент авторизован на ПК, размещающем приложение COM.

- данного пользователя - определите пользовательскую учётную запись, которая всегда будет использоваться для запуска приложения COM, независимо от того, к какому из приложений подключается пользователь.

12. Щёлкните по элементу Endpoints. Выберите протоколы по умолчанию (Default System Protocols).

13. Если Вы всё ещё получаете сообщение об ошибке Доступ не разрешён (Access Denied) или Нет прав доступа (Permission Denied) после настройки параметров Вашей DCOM, попробуйте перезапустить ПК, чтобы позволить новым параметрам вступить в силу.

НАСТРОЙКА DCOM НА WINDOWS XP И WINDOWS SERVER 2003

1. Если компьютер принадлежит не домену, а рабочей группе, следует убедиться, что он не использует простое совместное использование файлов. Откройте Windows Explorer или дважды кликните по элементу Мой компьютер (My Computer), кликните по элементу Инструментарий, а затем перейдите к элементу Опции папки (Folder Options), кликните Просмотр (click) и отмените выбор Использовать простое совместное использование файлов (Use simple file sharing) (рекомендовано) в Дополнительных параметрах (Advanced settings).

2. Кликните Старт (Start), кликните Программы (Programs), кликните Администрирование (Administrative Tools,) кликните Сервисы компонента (Component Services).

3. Разверните Сервисы компонента (Component Services), разверните компьютеры (Computers) и правой кнопкой мыши щёлкните по элементу Мой компьютер (My Computer). Выберите свойства.

4. Кликните Свойства по умолчанию (Default Properties). Выберите Активировать распределённую COM на этом компьютере. Установите Уровень авторизации для подключения по умолчанию (None тоже подходит). Установите Уровень имперсонализации для идентификации по умолчанию (Impersonate тоже подходит).

5. Кликните Безопасность COM по умолчанию (Default COM Security).

6. Под элементом Права доступа по умолчанию (Default Access Permissions) кликните по Редактировать по умолчанию (Edit Default). Добавьте SYSTEM, INTERACTIVE или NETWORK. Пользователь, чьи учётные данные будут использоваться для доступа к приложению COM, должны быть также включены в этот список. Существует много способов, как это сделать. Вы можете добавить специального пользователя или просто добавить группу, к которой принадлежит пользователь. Возможные значения включают:

- Домен\Имя пользователя (специальный пользователь)

- Домен\Администраторы (Все администраторы на специальном домене)

- Все (Все пользователи)

7. Под элементом Права доступа к запуску по умолчанию (Default Launch Permissions) кликните Редактировать по умолчанию (Edit Default). Убедитесь, что Права доступа к запуску имеют те же значению, что и Права доступа по умолчанию (Default Access Permissions).

8. Кликните Протоколы по умолчанию (Default Protocols). Убедитесь, что ориентированный на подключение TCP/IP перечисляется в начале.

9. Теперь Вы должны настроить приложение COM, к которому необходимо получить доступ. Кликните по приложению, а затем кликните правой кнопкой мыши по приложению, которое необходимо настроить. Выберите Свойства. Если Ваше приложение является библиотекой, следует сначала создать для замены EXE, используя инструментарий SetDllHost. После создания EXE, его имя появится в списке приложений. Выберите Свойства для него и продолжите действия.

10. Кликните Общее (General). Установите Уровень авторизации по умолчанию.

11. Кликните Расположение (Location). Выберите Запустить приложение на этом компьютере (Run application on this computer).

12. Кликните по элементу Безопасность (Security). Установите Использовать права доступа к запуску по умолчанию (Set Launch Permissions to Use Default). Установите права доступа по умолчанию (Access Permissions to Use Default). Установите права доступа к настройкам по умолчанию (Configuration Permissions to Use Default).

13. Кликните по элементу Идентичность (Identity). Выберите запускающего пользователя (launching user). Эта настройка задает учётную запись, которая будет использоваться для запуска приложения COM после того, как оно запущено программой клиента. Запускающий пользователь -- это пользовательская учетная запись, которая запускает сервер, это рекомендуемая настройка. В зависимости от приложения COM, к которому необходимо подключиться, может потребоваться изменить его на:

- интерактивного пользователя - пользователь, который в текущий момент авторизован на ПК, размещающем приложение COM.

- этот пользователь - задать пользовательскую учётную запись, которая будет всегда использоваться для запуска приложения COM, независимо от того, какой пользователь имеет к ней доступ.

14. Щёлкните по элементу Endpoints. Выберите протоколы по умолчанию (Default System Protocols).

15. Если Вы все ещё получаете сообщение об ошибке Доступ не разрешён (Access Denied) или Нет прав доступа (Permission Denied) после настройки параметров Вашей DCOM, попробуйте перезапустить ПК, чтобы позволить вступить в силу новым параметрам.

НАСТРОЙКА DCOM НА WINDOWS XP SP2

Microsoft добавил некоторые опции оптимизации безопасности DCOM в XP Service Pack 2. Помимо выше упомянутых параметров настройки для Windows XP DCOM, Вам потребуется выполнить следующие этапы:

1. Если компьютер принадлежит не к домену, а рабочей группе, убедитесь, что он не использует простое совместное использование файлов. Откройте Windows Explorer или дважды кликните по элементу Мой компьютер (My Computer), кликните по элементу Инструментарий, а затем перейдите к элементу Опции папки (Folder Options), кликните Просмотр и отмените выбор Использовать простое совместное использование файлов (Use simple file sharing) (рекомендовано) в Дополнительных параметрах (Advanced settings).

2.Кликните Старт (Start), кликните Программы (Programs), кликните Администрирование (Administrative Tools), кликните Сервисы компонента (Component Services).

3. Разверните Сервисы компонента (Component Services), разверните компьютеры (Computers) и правой кнопкой мыши щёлкните по элементу Мой компьютер (My Computer). Выберите свойства.

4. Кликните Безопасность COM по умолчанию (Default COM Security).

5. Под элементом Права доступа по умолчанию (Default Access Permissions) кликните Редактировать по умолчанию (Edit Default). Убедитесь, что SYSTEM, INTERACTIVE, NETWORK и пользователь, чьи учётные данные будут использоваться для доступа к приложению COM, все имеют права Локального и Удаленного доступа.

6. Под элементом Права доступа по умолчанию (Default Access Permissions) кликните Редактировать ограничения (Edit Limits). Service Pack 2 включает в себя следующие значения по умолчанию: ANONYMOUS LOGON (Локальный доступ) и Все (Локальные и удаленный доступ). Убедитесь, что эти значения перечислены, а затем добавьте пользователя, чьи учётные данные будут использоваться для доступа к приложению COM. Разрешите этому пользователю иметь локального и удаленного доступа (Local and Remote Access permissions).

7. Под элементом Права доступа к запуску по умолчанию (Default Launch Permissions) кликните Редактировать по умолчанию (Edit Default). Убедитесь, что SYSTEM, INTERACTIVE, NETWORK и пользователь, чьи учётные данные будут использоваться для доступа к приложению COM, имеют права локального и удаленного доступа, а также права локальной и удалённой активации.

8. Под элементом Права доступа к запуску по умолчанию (Default Launch Permissions) кликните Редактировать ограничения (Edit Limits). Service Pack 2 включает в себя следующие значения по умолчанию: MACHINE\Administrators (Локальный и удаленный запуск, локальная и удалённая активация) и Все (Локальный запуск и локальная активация). Убедитесь, что эти значения перечислены, а затем добавьте пользователя, чьи учётные данные будет использоваться для доступа к приложению COM. Разрешите этому пользователю иметь права доступа к локальному и удалённому запуску (Local and Remote Launch permissions), а также разрешения локальной и удалённой активации (Local and Remote Activation permissions).

9. Service Pack 2 включает в себя встроенный Windows Firewall. Если firewall включён, вам придётся позволить приложению COM подключиться по сети к Вашему ПК. Вы можете это сделать, открыв Windows Firewall и добавив Ваше приложение COM в список программ, расположенных во вкладке Исключения (Exceptions). Если выбрано Отображать уведомление, когда Windows Firewall блокирует программу, тогда система вам подскажет разблокировать приложение COM, когда AtomMind Server подключается к Вашему DCOM-серверу в первый раз. Выберите Разблокировать (Unblock), когда система предлагает так поступить.

10. Если все ещё всплывает сообщение об ошибке Доступ не разрешён (access denied) или Нет прав доступа (permission denied) после настройки параметров DCOM, попробуйте перезапустить Ваш ПК, чтобы изменения вступили в силу.

20.14 Подготовка сервера Linux без графического инте

Эта статья объясняет, как подготовить сервер Linux без графического интерфейса пользователя для запуска AtomMind Server, который активирует веб-виджеты, т.е. виджеты 943, которые работают в веб-браузере без поддержки Java.

Чтобы активиривать операцию веб-виджета, сначала установите **Xvfb**. Xvfb - это виртуальный кадровый буфер X, он действует как оперативная память сервера X (на экране нет никакого вывода).

Инсталляция Xvfb

Следуйте ниже приведенной инструкции для установки Xvfb как сервиса.

Сначала нужно установить Xvfb и необходимые дополнения:

Ubuntu/Debian:

```
sudo apt-get install xvfb
sudo apt-get install libxrender-dev
sudo apt-get install libxtst-dev
```

RedHat/Fedora:

```
yum install xorg-x11-server-Xvfb
yum install libXrender.x86_64
yum install libXtst.x86_64
```

Вы можете воспользоваться опцией поиска менеджера пакетов, чтобы задать правильное имя пакета, который вам необходим: `yum search xvfb`

Активация использования Xvfb

Найдите конфигурационный файл запуска или службу AtomMind Server (`am_server.conf` или `am_server.service` -- см. Режим сервиса 159) и раскомментируйте строку, отвечающую за запуск `am_server` через `xvfb-run`. Команда в смежной строке запускает `am_server` напрямую.

Перезагрузите 158 сервер после завершения конфигурирования.

20.15 Потоки и пулы потоков AtomMind Server

Здесь перечислены наиболее важные потоки и пулы потоков, используемые AtomMind Server. Это помогает опытным системным администраторам и разработчикам модулей/решений основательно понимать статус внутреннего сервера и устранять проблемы во время работы.

Получить доступ к текущему статусу и трассировкам стека всех потоков сервера (включая описанные здесь) можно через переменные действий и статуса корневого контекста 1559 (такие переменные, как **Все потоки**, **Статистика пулов потоков** и **Статистика потоков** доступны через действие **Просмотр статуса сервера**).

Префикс имени потока	Информация о потоке и пуле потоков
`Action/`	Потоки действий выполняют действия 87 сервера. Такой поток может либо выполнять код действия на стороне сервера, либо блокироваться в ожидании ответа пользователя на UI процедуру.
`ClientCommandProcessor/`	Пул потоков обработки команд клиентов обрабатывает команды десктопных и API-клиентов, а именно, **Get/Set Variable**, **Call Function** и **Subscribe/Unsubscribe Event**. После начала обработки команды имя потока обработки меняется так, чтобы отражать суть команды, по следующем шаблону: • *Context_name variable_name* - при обработке команды **Get Variable** • *Context_name variable_name* - при обработке команды **Set Variable** • *Context_name function_name* - при обработке команды **Call Function**
`ClientThread/`	Клиентские потоки обслуживают подключения десктопных и API-клиентов. Для каждого подключения создается отдельный поток.
`ContextOperationExecutor/`	Пул потоков операций контекста используется для ускорения параллельно выполняемых действий контекстов 41 сервера. Он используется, в большей степени, при запуске и остановке сервера. Например, каждый ресурс (такой, как тревога 779) в контексте контейнера инициализируется в отдельной задаче, которая предъявляется пулу исполнителей операций контекстов. Именно поэтому проверка статуса пулов потоков может очень пригодиться для диагностики производительности сервера при запуске и остановке.
`EventDispatcher/`	Потоки диспетчера событий получают события 73 из очередей событий и вызывает слушателей 75 событий, которые эти события обрабатывают. Таким образом, проверка статуса диспетчеров событий может быть хорошей идеей при увеличении очереди событий или при общем запаздывании обработки событий, с точки зрения пользователя. Основной поток диспетчера событий сервера называется `EventDispatcher/Master`.
`Master`	Имя основного потока AtomMind Server.
`Synchronizer/`	Потоки синхронизирующих устройств выполняют синхронизацию 514 устройств. Потоки вызывают драйвера устройств и увязывают данные между драйвером и контекстом устройства 1494. Проверка состояния потоков синхронизатора следует проводить для диагностики запаздывания синхронизации устройств и других связанных проблем. Это особенно касается пользовательских драйверов устройств, созданных на базе набора для разработки драйверов 1345.
`SyncTimer/`	Потоки данного пула - это потоки таймера, который отвечает за начало периодической синхронизации 514 устройств.

20.16 Разработка на платформе AtomMind

В этой главе описаны рекомендованные стандарты по разработке на платформе AtomMind. Следование данным рекомендациям позволит упростить как разработку приложений, так и последующее их сопровождение.

именование Объектов

CAMELCASE

При добавлении любого объекта (устройства, модели 810, переменной 1900 и т.д.) для него необходимо заполнить поле **Name**. При этом рекомендуется использовать стиль lowerCamelCase - все составные слова пишутся слитно, без пробелов, каждое слово в фразе начинается с заглавной буквы, кроме первого, как в приведенных ниже примерах:

- Имя контекста 41: monitoringMainPage, mqttDevice
- Имя функции 70: getUserInformation, xmlHttpRequest
- Имя переменной 1900: inputData, unitsList

ПОНЯТНЫЕ ИМЕНА

Имя должно всегда отражать основную суть объекта (устройства, модели 810, переменной 1900 и т.д.). При разработке приложения постоянно приходится обращаться к объектам, например получить данные из переменной, и в таком случае, правильное имя поможет быстрее найти ее в списке других переменных. В дальнейшем, если возникнет необходимость доработать приложение или исправить баги, понятные имена объектов в коде выражения 37 упростят его понимание.

Примеры именования объектов:

- Предположим, проект содержит несколько настроенных экранов, реализованных при помощи инструментальных панелей 912. Из названия инструментальной панели должно быть понятно, какую функциональную роль она имеет: startPage, monitoringSettings, alertsHistory.
- Предположим, модель содержит несколько функций 70 с похожим функционалом. Использование имен "findUser1" и "findUser2" может вызвать путаницу и непонимание. Правильнее будет указать в имени более полную информацию, показывающую назначение функции, например, "findUserByName" и "findUserById".

В имени контекста не надо указывать его тип. Тип контекста понятен по группе в которой он лежит, а также является частью его пути

Именование Функций и наборов ПРавил

Любая функция 70 или набор правил 743 в модели 810, не зависимо от сложности, всегда выполняют определенное действие. Именно поэтому, их имя должно отвечать на вопрос “Что делает эта функция/набор правил?”, чтобы упростить их использование из любого места приложения. Имена должны всегда начинаться с глагола. Кроме того, при сортировке списка функций/наборов правил в модели по имени, они будут отсортированы еще и по логическому смыслу (например, сначала будут идти все функции по чтению переменных, потом все функции по обновлению данных в переменных и т.д.).

Примеры именования фукнций и наборов правил:

- Предположим, что есть устройство, которое выходит на связь раз в сутки. В модели создана привязка 735, которая при выходе устройства на связь выполняет функцию getDeviceTelemetry. Благодаря имени функции сразу стало понятно назначении привязки - при выходе устройства на связь получить телеметрию с утройства.
- Другие примеры имен для функций: createSensor, getUserInformation, sendAlertsListByEmail

Так как наборы правил это те же функции, только находящиеся в другой таблице модели, возможна ситуация совпадения имен. Это приведет к конфликту при вызове функции. Чтобы гарантированно избежать этого, для каждого набора правил можно добавлять постфикс "RuleSet".

Заполнение полей с Описаниями и Комментариями

Всегда заполняйте поля с описанием и комментариями объектов (устройств, моделей 810, переменных 1900 и т.д.), если они предусмотрены. Краткие и емкие описания объектов позволяют легко определить назначение объектов, что сильно помогает при разработке и обслуживании приложения.

По комментариям в наборах правил 743 можно быстро понять как назначения отдельного шага, так и общую структуру всей выполняемой функции.

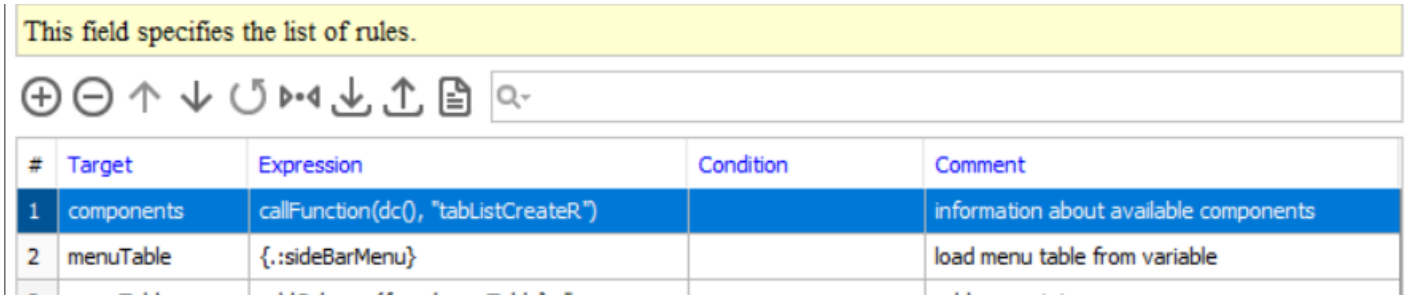
Комментарий должен отражать суть действия, а не описывать каждую используемую операцию (например, почему используется именно это операция, а не другая). Такой подход решает сразу ряд задач:

- Комментарии не будут перегружены лишней информацией
- По комментариям можно понять назначение различных шагов, не читая их код
- При необходимости исправить часть функционала набора правил, по комментарию можно будет быстро найти нужный шаг, не перебирая их один за другим

В коде выражений 37 комментарии надо сводить к минимуму. В грамотно написанном выражении комментарии обычно не нужны. Для этого необходимо правильно оформлять код, использовать понятные имена объектов и разбивать сложные выражения на отдельные шаги или функции 70.

Примеры заполнения полей описания и комментариев:

- Предположим, есть функция, которая показывает информацию об указанном во входящих параметрах пользователя. Ей можно дать имя getUserInformation, а в описании указать более подробную информацию “Получить информацию об указанном пользователе”
- По информации в комментарии набора правил мы понимаем назначение шага, даже не читая его код:



Удаление закомментированных фрагментов кода выражений

Иногда для отладки выражения 37 приходится заменять переменные окружения 123 прямыми ссылками. Так как процесс отладки может занимать несколько итераций, в коде выражения часто присутствуют сразу два варианта исполнения, один из которых заключен в блок комментария. Подобные куски кода надо обязательно убирать по окончании отладки, чтобы избежать потенциальной уязвимости приложения, особенное если в нем содержатся важные данные, например имена пользователей, пароли и т. д.:

```
callFunction(
      {users.admin.models.links:serviceModel$path}
      , "readDeviceSetings"
      , {env/deviceId}
      , {env/userName}
      , {env/password}
)

/*
callFunction(
      "distributedServer.users.admin.models.serviceModel"
      , "readDeviceSetings"
      , "027472251"
      , "admin"
      , "d46geWSff3w"
)
*/
```

Единая модель именования ОБъектов для Тестирования и Отладки Приложения

Для тестирования приложения лучше использовать отдельные, специально предназначенные для этого объекты (устройства, модели 810, переменные 1900 и т.д.). В описание такого объекта необходимо добавлять специальный

префикс или постфикс (например, "test"). Это очень пригодится на финальном этапе разработки, когда надо будет приводить приложение в порядок и убирать все мусорные ресурсы.

Примеры:

- Предположим, в наборе правил 743 есть шаг, который возвращает таблицу 49 с телеметрией указанного устройства. Полученные данные используются в дальнейших шагах набора правил, но доступа к устройству временно нет, и проверить работоспособность становится невозможным. В таком случае можно создать копию нужного шага, в комментарии указать "test" и вместо исходного выражения указать таблицу нужного формата с тестовыми данными. В оригинальной строке в графе с условием пропишите “false”, чтобы она не отработала. Таким образом можно проверить работоспособность набора правил, и все строки с тестовыми данными будут сразу видны, что позволит легко все вернуть на место.
- В примере выше можно поступить другим образом. Создайте копию самого набора правил. В оригинальном наборе правил добавьте префикс “на отладке”, а в описании копии допишите “для тестирования”. В новом наборе правил замените все ресурсы, к которым нет доступа, на константы или тестовые данные. По окончании работ, можно будет легко найти набор правил с тестовыми данными, удалить его, а оригинальный вернуть на место.

Не забудьте сразу удалить все ресурсы, которые использовались для тестирования. Если планируется использовать их в дальнейшем, лучше создать специальные группы 751 контекстов и перенести туда тестовые ресурсы.

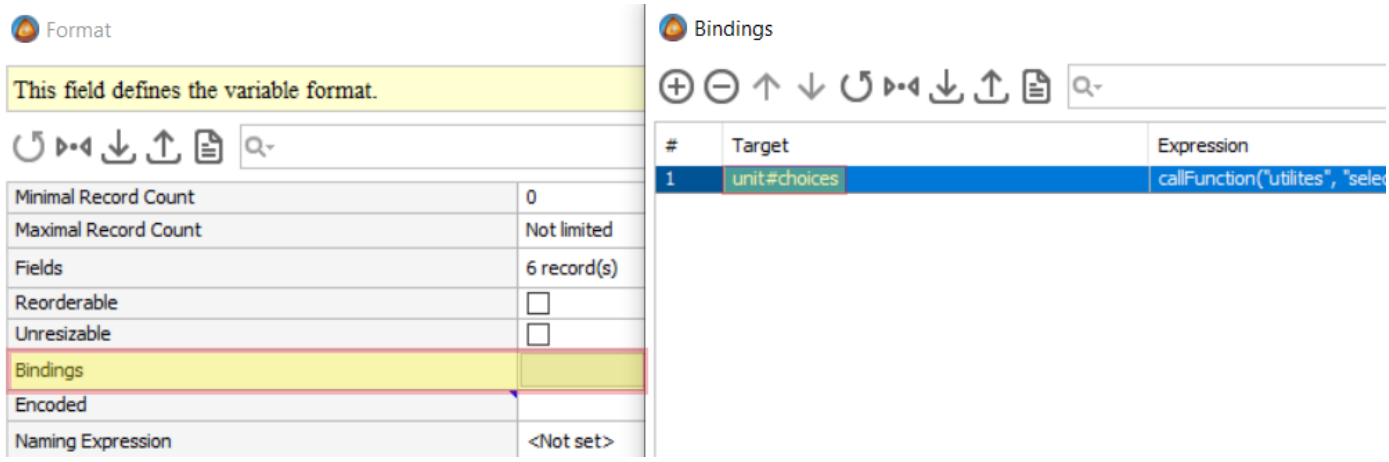
Использование Справочников и Развязочных Таблиц вместо Прямых Ссылок и Констант

Использовать прямые ссылки и константы может быть очень удобно при написании кода, однако в долгосрочной перспективе это может привести к нежелательным последствиям. Если один из таких параметров изменится, потребуются огромные трудозатраты для поиска всех мест, где он использовался, и изменения его вручную. Еще один важный момент - использование справочников закладывает фундамент для дальнейшей масштабируемости проекта. К тому же, использование констант может привести к ситуации, когда один и тот же параметр называется по-разному в системе. В результате возникает путаница и необходимость создавать сложные конструкции с условиями, чтобы связать части приложения в единое целое.

В большинстве случаев, в качестве имени параметра в справочнике лучше использовать цифровое значение, а в описании уже писать подробную информацию.

Примеры использования справочников:

- Хороший пример использования справочника - справочник единиц измерения, который можно разделить на три поля: код, краткое наименование, полное наименование. В дальнейшем значения этого справочника будут использованы во всем проекте. Для переменных 1900 есть возможность привязать значения другой переменной в качестве допустимых значений. В формате переменной необходимо добавить соответствующую запись в привязках. В качестве цели указывается поле, к которому надо привязать справочник с параметром #choices:



В выражении 37 пишется специальная функция с указанием пути к справочнику и именам полей, которые будут использоваться в качестве имени и описания выбранного поля переменной:

```
callFunction(
       "utilities"
     , "selectionValues"
     , "{users.admin.models.resources:unit}"      //path to a variable with dictionary
     , "{value}"                                  //field name used as value
     , "{description}"                            //field name used as description
)
```

- Другой полезный вариант использования справочников - развязочная таблица с кодами ошибок. В функции 70 может быть предусмотрен механизм останова сценария при определенных условиях, например, если пришла неверная контрольная сумма пакета. Если функция будет вместо текста возвращать код ошибки, а в системе будет специальная развязочная таблица таких сообщений, то будет гораздо легче и удобнее реагировать на ошибки и информировать пользователя. Давайте сравним.

На одном из шагов набора правил идет проверка контрольной суммы. Если контрольная сумма окажется неверной, происходит выход из набора правил. В результате возвращается текстовое сообщение, указанное в выражении. Обрабатывать такую строку не очень удобно:

The expression evaluation result is written into the Target.

```
callFunction(dc(), "checkCrc", {env/calue}) == {env/crc}
? "OK"
: "CRC is not valid"
```

5	crcCheck	callFunction(dc(), "checkCrc", {env...		check CRC for valid name
6	Final Rule Set Result	{env/crcCheck}	{env/crcCheck}!="OK"	exit if CRC is not valid

Для более удобной работы в подобных случаях лучше использовать справочник. В рассмотренном выше примере текст сообщения надо заменить на код ошибки:

The expression evaluation result is written into the Target.

```
callFunction(dc(), "checkCrc", {env/calue}) == {env/crc}
? 0
: "Err-00134"
```

В справочнике для данного кода ошибки заполнены поля с кратким и полным описанием ошибки на нескольких языках.

#	Error Number	Short Description En	Full Description En	Short Description Es	Full Description Es
1	Err-00134	Error in receiving data	CRC mismatch (device settings have been changed)	Error al recibir datos	Falta de coincidencia de CRC (la configuración del dispositivo ha cambiado)

В случае ошибки запустится специальное правило обработки. Оно будет универсальным для всего проекта и его лучше вынести в отдельную функцию.

5	crcCheck	callFunction(dc(), "checkCrc", {env/calue}...		check CRC for valid name
6	Final Rule Set Result	callFunction("users.admin.models.service"...	{env/crcCheck}!=0	exit if CRC is not valid

Правило обработки получает на вход номер ошибки. Затем в зависимости от роли текущего пользователя и используемого им языка из справочника будет подбираться нужное сообщение об ошибке.

The expression evaluation result is written into the Target.

```
select(
      getVariable("users.admin.models.service", "errorList")
      , {env/userRole} == "admin"
      ? "fullDescription" + {env/local}
      : "shortDescription" + {env/local}
      , "errorNumber"
      , {env/errNumber}
)
```

Пример показывает, как справочник помогает сделать удобный обработчик ошибок, учесть ролевую модель, а также имеет мультиязыковую совместимость.

Расположение параметров Функции в Столбик

В языке выражений для большинства функций 70 необходимо задавать различные параметры. В описании каждой функции указаны ее параметры, записанные в одну строку через запятую. Если параметров меньше трех и они достаточно короткие, то допустимо в коде выражения 37 писать их в одну строку. Однако, в большинстве случаев необходимо указывать параметры в столбик. Это значительно облегчает работу с функцией и делает ее более читаемой. Например, сразу видно, сколько у функции параметров, где они начинаются и заканчиваются, некоторые из них можно очень легко закомментировать для быстрой отладки. При таком подходе все разделители строки (знак "," или "+") необходимо ставить в начале строки. Это помогает понять, где начинается новый параметр, и облегчает работу с кодом.

Примеры:

- Допустим, в выражении собирается одна таблица на основе другой. В результате, параметров у функции становится довольно много и перечисление их одной строкой приведет к трудностям чтения такого кода:

The expression evaluation result is written into the Target.

```
table(
"<<room1><S><<temperature1><S>><<room2><S><<temperature2><S>><<room3><S><<temperature3><S>><<room4>
<S><<temperature4><S>>",cell({env/dataTable},"room",0),cell({env/dataTable},"temperature",0),cell({
env/dataTable},"room",1),cell({env/dataTable},"temperature",1),cell({env/dataTable},"room",2),cell(
{env/dataTable},"temperature",2),cell({env/dataTable},"room",3),cell({env/dataTable},"temperature",
3))
```

Если же параметры расположить в столбик, то код становится намного более читаемый и понятный:

The expression evaluation result is written into the Target.

```
table(
        "<<room1><S><<temperature1><S>><<room2><S><<temperature2><S>>"
      + "<<room3><S><<temperature3><S>><<room4><S><<temperature4><S>>"
      , cell({env/dataTable},"room",0)
      , cell({env/dataTable},"temperature",0)
      , cell({env/dataTable},"room",1)
      , cell({env/dataTable},"temperature",1)
      , cell({env/dataTable},"room",2)
      , cell({env/dataTable},"temperature",2)
      , cell({env/dataTable},"room",3)
      , cell({env/dataTable},"temperature",3)
)
```

- Благодаря тому, что символ ",” стоит в начале строки, можно не только легче определить, какие и сколько параметров используется в функции, но и упростить отладку - один или несколько параметров легко закомментировать, а при необходимости вместо них подставить нужное значение:

The expression evaluation result is written into the Target.

```
addColumns(
        {env/table}
      , "<temperature1><S>", {env/temperature1}
      , "<temperature2><S>", {env/temperature2}
      , "<temperature3><S>", {env/temperature3}
//    , "<temperature4><S>", {env/temperature4}
      , "<temperature5><S>", {env/temperature5}
//    , "<temperature6><S>", {env/temperature6}
)
```

Разбивка Длинных Строк

В языке выражений[37] параметры во многих функциях[70] задаются в формате строки (String). Их длина может быть довольно большой, что затрудняет восприятие выражения. В таких случаях рекомендуется разбивать одну строку на несколько. Чтобы разбить строковое выражение, необходимо в нужном месте поставить закрывающие кавычки, затем написать знак “+” и открыть кавычки перед оставшейся частью текста. Этим способом очень удобно разделять параметры вложенных функций.

Очень важно все разделители ставить в начале строки. К ним относятся:

- "+" (плюс) - отделяет части одного строкового выражения
- " " (пробел) - разделяет слова в строке
- "," (запятая) - разделяет параметры в функциях

В примере показано разделение строки при работе с SQL запросом. Запрос может обращаться ко множеству контекстов[41], таблиц данных[49], переменных[1900]. Если все они будут записаны одной строкой, работать с запросом может быть проблематично. Разделение длинной строки на несколько маленьких не только облегчит восприятие кода, но и поможет в дальнейшей работе и отладке. Указав в запросе каждую переменную с новой строки, а также условие для фильтрации, любую из них можно очень быстро закомментировать, добавить, удалить или заменить:

The expression evaluation result is written into the Target.

```
callFunction(
        ""
      , "executeQuery"
      , "select distinct data.deviceModels$model as value"
//    + ", data deviceModels$resourceType as resourceType"
      + ", data deviceModels$impulse as impulse"
      + " from user.admin.models.serviceModel:deviceModels as data"
      + " where data.deviceModels$model='"+{form/comboBoxModel:currentOptions$value}+"'"
//    + " and data.deviceModels$resourceType='"+{form/comboBoxResource:currentOptions$value}+"'"
)
```

Использование Табуляции для Вложенных Функций

Очень важно при написании кода выражения [37] использовать табуляцию для параметров, перечисленных в столбик, и вложенных функций [70]. Это сильно облегчает восприятие кода. Хорошо видны начало и конец каждого уровня вложенности, число параметров и функций и т. д. См. пример ниже.

Часто выражения содержат несколько функций со множеством параметров. Если все их писать в одну строку без четкого разделения, прочитать такое выражение становится очень трудно

The expression evaluation result is written into the Target.

```
addColumns(table("<<value><S>><<lable><S>>", cell(getVariable("users."+cell(callFunction({:},
"sessionGet","var")), "settings"),"enabled"),cell(getVariable("users."+cell(callFunction({:},
"sessionGet","var")), "settings"),"enabled")?"active":"blocked"), "<tooltip><S>" , "{label}")
```

С первого взгляда практически невозможно понять, сколько в примере выше функций и параметров. Если же разложить данное выражение на составляющие и каждую вложенную функцию писать, используя табуляцию, код станет намного понятнее:

The expression evaluation result is written into the Target.

```
addColumns(
      table(
            "<<value><S>><<lable><S>>"
            , cell(
                  getVariable(
                        "users."+cell(callFunction({:},"sessionGet","var"))
                        , "settings"
                  )
                  , "enabled"
            )
            , cell(
                  getVariable(
                        "users."+cell(callFunction({:},"sessionGet","var"))
                        , "settings"
                  )
                  , "enabled"
            ) ? "active" : "blocked"
      )
      , "<tooltip><S>"
      , "{label}"
)
```

Декомпозиция Длинных Выражений

Выражения 37 могут содержать множество функций 70, как вложенных, так и выполняемых последовательно с различными условиями. Понять такие выражения может быть тяжело, а работать с ними неудобно. В таких случаях лучше всего использовать наборы правил 743 в моделях 810. Благодаря этому каждый отдельный шаг выражения будет отчетливо виден. Это облегчит не только работу по созданию функционала проекта, но и его дальнейшую поддержку.

Иногда вместо использование языка выражений лучше написать скрипт на Java. Например, для организации цикла или обработки большого массива данных. Главное помнить, что не все разработчики на платформе знают язык Java.

Пример декомпозиции длинного выражения:

Допустим, для реализации функции необходимо выполнить нескольких последовательных действий. Если записать их одной строкой, то выражение становится трудно понять:

The expression evaluation result is written into the Target.

```
1  union(addColumns(sort(filter({env/device1},"{value}>=2"),"value",1),"<data><D>",now()), addColumns(
   sort(filter({env/device2},"{value}<5"),"value",1),"<data><D>",now()),sort({env/device3},"value",1))
```

Даже если использовать табуляцию для вложенных функций, выражение будет перегруженным. Именно поэтому в подобных случаях надо разбивать выражение на отдельные шаги. Благодаря комментариям и четко выделенным действиям, работать с функцией становится намного легче:

This field specifies the list of rules.

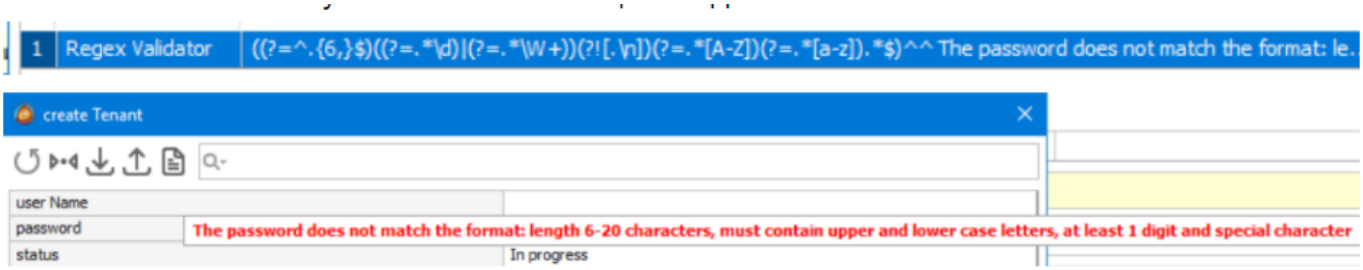
#	Target	Expression	Condition	Comment
1	device1	{0}		parameters of device 1
2	device2	{1}		parameters of device 2
3	device3	{2}		parameters of device 3
4	filterDevice1	filter({env/device1},"{value}>=2")		remove rows with bad parameters
5	sortDevice1	sort({env/filterDevice1},"value",1)		sorting table
6	newTableDevice1	addColumns({env/sortDevice1},"<data>...		final table for device 1
7	filterDevice2	filter({env/device2},"{value}<5"),"value",1)		remove rows with bad parameters
8	sortDevice2	sort({env/filterDevice2},"value",1)		sorting table
9	newTableDevice2	addColumns({env/sortDevice2},"<data>...		final table for device 2
10	newTableDevice3	sort({env/device3},"value",1)		final table for device 3
11	Final Rule Set Result	union({env/newTableDevice1},{env/new...		final result

использование Валидаторов

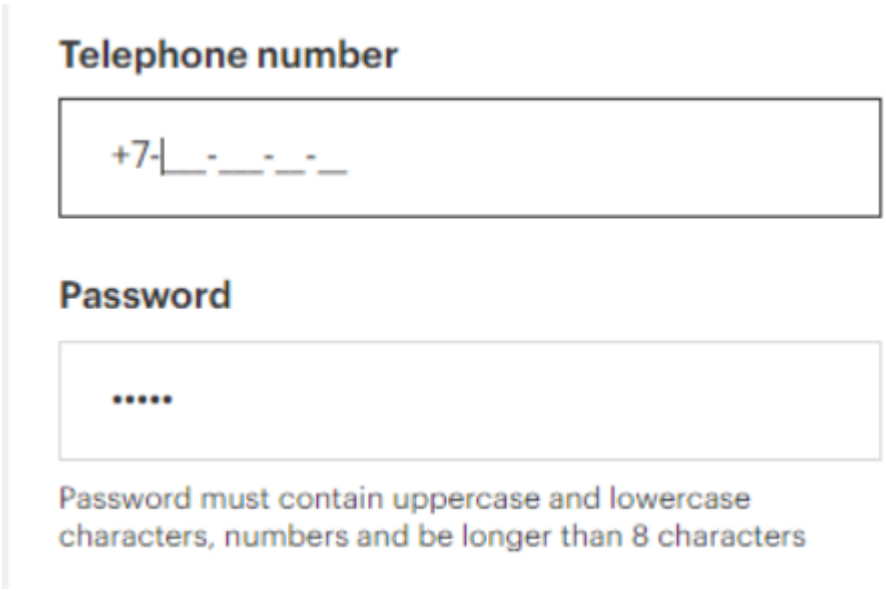
Валидаторы - очень важная часть любого проекта. Их грамотное использование позволит избежать непредсказуемого поведения различных элементов проекта. Особенно важно использовать валидаторы там, где пользователь должен вводить какие-то данные. При этом ограничения должны быть обоснованными, например для ввода IP адреса будет 4 блока с цифровыми значениями от 0 до 255. Также очень важно информировать пользователя об ограничениях, установленных в системе, чтобы ему не пришлось гадать, почему не работает функция или не принимается параметр.

Примеры использования валидаторов:

- Валидаторы можно установить на любую переменную 1900, что защитит от записи нежелательных данных в базу. В настройках переменной для нужного поля необходимо заполнить специальную графу с валидатором. Например, можно использовать регулярное выражение, чтобы задать правила записи поля с паролем. Там же можно указать текст сообщения для пользователя:



- Еще один вариант валидации - создание маски ввода. Задав нужные условия, можно не только ограничить пользователя в вводимых данных, но и более наглядно показать информацию. Данный метод очень удобно применять в виджетах[943] и инструментальных панелях[912]. Один из самых удачных примеров использование маски - поле с номером телефона:



20.17 Устаревшие разделы

Enter topic text here.

20.17.1 Общие данные

Общие данные представляют собой пользовательский модуль хранения данных, которые могут быть использованы различными компонентами AtomMind Server, и его плагинов[207] для выполнения специальных операций, таких как репликация данных, разделение данных и управление конфигурацией. Для более подробной информации см. Использование общих данных[217].

Рекомендуется вместо модуля Общие данные использовать Модели[810], которые полностью покрывают весь функционал модуля Общие данные. В будущих релизах AtomMind, модуль Общие данные может быть недоступен.

Контейнер[217] общих таблиц подобен базе данных, построенной из *таблиц*, поддерживающей следующие операции:

- **Создание таблиц**. Контейнер общих данных позволяет создавать таблицы. Единственным ограничением является то, что все таблицы внутри Контейнера должны иметь уникальные имена. Каждая таблица имеет имя и описание, но описание необязательно должно быть уникальным.

- **Удаление таблиц**. Таблица может быть удалена вместе со всем ее содержимым.
- **Описание и модификация структуры таблицы.** Пользователь (или определенный плагин) может динамично изменять структуру таблицы, чтобы успешно выполнялась задача.
- **Просмотр и изменение данных.** Пользователи могут просматривать и изменять содержание общих таблиц.

Общая таблица является таблицей данных 49 определяемого пользователем формата, администрирование которой осуществляется посредством контекста общих данных, а сама таблица является доступной для различных пользователей или устройств.

Контейнеры общих данных

Каждый контейнер общих данных представлен контекстом общих данных 1483. Пользовательские контейнеры принадлежат пользователям AtomMind Server. Каждый пользователь обладает своим собственным контекстом общих таблиц, путь которого равен `users.USER_NAME.common`. Пользователь может использовать свой контейнер общих таблиц для хранения данных, которые необходимо разделять между его устройствами или группами 751, такими как регистрационный файл, отображаемый у всех Device. Для более подробной информации см. Использование общих данных 2177.

Администрирование общих данных

Для администрирования общих данных (и таблиц их содержащих) используются два типа контекста: контекст Общие данные 1483, который служит контейнером для нескольких общих таблиц, и контекст Таблица общих данных 1485, который содержит информацию об одной общей таблице.

20.17.1.1 Использование общих таблиц

Таблицы общих данных используются в различных сценариях. Для более подробной информации см. следующие статьи:

- Пользовательское хранилище данных 2177
- Пользовательские свойства 2177
- Мастер-свойства членов группы 2178

20.17.1.1.1 Пользовательское хранилище данных

Администратор может вручную создать таблицу данных для хранения информации, которая автоматически не предоставляется системой. Это может быть, например, перечень принадлежащего фирме оборудования. Такая таблица может содержать описание и местоположение устройства, данные об ответственном за него лице и так далее. Администратор может назначить пользователей для просмотра и редактирования этого списка. На эту таблицу можно ссылаться из различных частей системы, например, можно создать запрос 829, показывающий, за какое оборудование ответственен тот или иной пользователь 478.

20.17.1.1.2 Пользовательские свойства

Возможно добавить общую таблицу к одному или более контекстам 41 в виде **пользовательского свойства**. Данный метод позволяет администраторам AtomMind добавлять новые свойства к различным системным объектам, чтобы в дальнейшем ссылаться на данные свойства из различных частей системы. Например, можно добавить свойство "**Местоположение**" для всех устройств, содержащее информацию о здании, этаже и номере комнаты, где данный элемент оборудования расположен физически. После этого не составит труда посмотреть местоположение всего оборудования, используя запрос 829, или создать отчет.

Для данного применения между различными контекстами распределяется только **формат** общей таблицы, в то время как каждый из них имеет собственные **данные** (значение пользовательского свойства).

Сопутствующая документация: Добавление пользовательских свойств 1686

Пользовательское свойство добавляется только к контекстам, которые доступны для владельца общей таблицы в соответствии с его правами доступа 477.

Чтобы прочитать это свойство, пользователь должен иметь действительный уровень прав доступа 489 **Наблюдатель** в двух контекстах:

- Контекст общей таблицы 1485 общей таблицы, которая служит базой для этого пользовательского свойства
- Контекст, к которому добавляется это свойство

Чтобы записать это свойство, у пользователя должен быть действительный уровень прав доступа 489 **Менеджер** в тех же контекстах.

Действие Редактировать пользовательские свойства

Действие Конфигурировать 105, называемое "**Редактировать пользовательские свойства**", используется для изменения значений пользовательских свойств 2177 контекста. Оно добавляется к контексту, когда в нем присутствует хотя бы одно пользовательское свойство.

Иконка действия: 

20.17.1.1.3 Мастер-свойства членов группы

Общая таблица может содержать **мастер-значение** для определенного свойства всех членов группы 751. Группа контролирует изменения общей таблицы, и при их обнаружении группа копирует определенное свойство 61 для всех ее членов согласно содержанию общей таблицы. Это легко, поскольку различные значения свойства контекста 41 представлены таблицей данных 49, так же как и общая таблица 2178.

В данном случае общая таблица не создается посредством добавления полей и записей. Наоборот, чтобы ее создать, необходимо использовать действие Создать новую таблицу из переменной 1484, которое берет значение определенной пользователем переменной контекста и создает общую таблицу с таким же форматом и содержанием.

Формат 2178 общей таблицы должен совпадать (или расширять) с форматом 49 свойства, мастер-значение которого она содержит. Таким образом, она должна содержать все поля, назначенные для данного свойства, но также может содержать и дополнительные поля. Имя общей таблицы должно совпадать с именем соответствующего свойства - это позволяет группе найти данную общую таблицу и использовать ее.

Более подробную информацию о данном свойстве см. здесь 759.

20.17.1.2 Содержание общей таблицы

Каждая общая таблица по сути является таблицей данных 49. Ее формат 50 определен свойствами полей 2179 и может быть изменен при помощи действия Редактировать данные 1485. Количество записей ограничено минимальными и максимальными значениями, определенными в свойствах таблицы 2179. Записи в таблицы могут быть заново упорядочены.

При изменении формата таблицы (точнее формата поля (полей) таблицы), когда таблица уже содержит некоторые данные, сервер пытается максимально сохранить данные и конвертировать их в новый формат. В большинстве случаев данные таблицы не изменяются. Например, если целочисленное поле было конвертировано в строковое, значения данных полей преобразуются в строки. Но если строковое поле было изменено в целочисленное, не каждое строковое значение поля может быть конвертировано в целое число и некоторые данные могут быть потеряны. Именно поэтому вам следует избегать изменений структуры таблицы, если она содержит важные данные.

Единственным исключением из вышеуказанного правила является переименование поля, которое возможно только если таблица не заполнена (не содержит никаких данных).

Изменения формата, которые могут повлиять на данные в таблице:

- Изменение типа поля (значения, которые не могут быть конвертированы в новый тип, будут установлены на значения по умолчанию для данного поля).
- Изменение типа поля с "**Пустого**" (Nullable) на "**Непустое**" (non-Nullable) (значения *NULL* будут изменены на заданные по умолчанию).
- Удаление **Допустимых значений**, если отключены **Расширяемые допустимые значения** для данного поля (ячейки, которые содержали удаленные значения, будут установлены по умолчанию).
- Отключение **Расширяемых допустимых значений** (ячейки, содержащие значения, которые не отображаются в значениях выборки, будут установлены по умолчанию).
- Когда **Минимальное количество записей** является величиной большей, чем текущее количество записей в таблице, для "дополнения" таблицы создаются пустые записи со значениями по умолчанию.

- Когда **Максимальное количество записей** является величиной меньшей, чем текущее количество записей в таблице, все "лишние" записи обрезаются с конца таблицы (удаляются).

Данные общей таблицы хранятся в базе данных[692] AtomMind Server совместно с другими свойствами контекста. Просмотр или изменение данных доступно через выполнение действия Редактировать данные[1485].

Изменения формата или содержания таблицы фиксируются другими системными компонентами. Для более подробной информации см. Использвание общих таблиц[217] (а именно: **Интеграция с группами** и **Интеграция с терминалами данных**).

Содержание общей таблицы (скриншот из AtomMind Client):

	ID-code	Name	Category	Exp. date
1	442	Victor	0	24.08.2007
2	123	Anton	0	24.08.2007
3	x1	Lisa	0	24.08.2007
4	zzz	John	2	24.08.2007
5	for2	Jennifer	1	17.10.2007

20.17.1.3 Настройка общей таблицы

Данный раздел посвящен свойствам конфигурации общей таблицы.

Все свойства, описанные далее, доступны через действие Настроить[105] контекста Общая таблица[1485].

20.17.1.3.1 Свойства общей таблицы

Данное свойство определяет основные опции общей таблицы:

Описание поля	Имя поля
Имя таблицы. Имя контекста общей таблицы[1485]. Должно соответствовать соглашениям о наименовании[42]. Данное имя используется для ссылки на данную таблицу из других частей системы.	name
Описание таблицы. Текстовое описание таблицы. Является также описанием[43] контекста общей таблицы.	description
Минимальное количество записей.	minRecords
Максимальное количество записей.	maxRecords
Включить права доступа для индивидуальных записей. Права доступа к каждой записи позволяют вам контролировать, какие контексты[41] имеют доступ ко всем записям в таблице (в отличие от контроля доступа ко всей таблице).	recordPermissions

Данной свойство доступно через переменную childInfo[1486].

20.17.1.3.2 Поля общей таблицы

Данное свойство определяет поля общей таблицы.

Описание поля	Имя поля
Имя. Уникальное имя поля.	name
Тип. Один из типов, поддерживаемых форматом таблицы данных[49].	type
Описание. Текстовое описание поля.	description
По умолчанию. Значение поля по умолчанию.	default

Только для чтения. Указывает, что поле доступно только для чтения.	readonly
Может быть NULL. Указывает, что поле может содержать значения *NULL* ("не задано").	nullable
Ключевое. Указывает, что данное поле является ключевым 49.	key
Допустимые значения. Список возможных значений для данного поля.	selvals
Расширяемые допустимые значения. Указывает, что поле может содержать значения, не входящие в список допустимых значений.	extselvals
Справка. Дополнительная информация о поле.	help
Редактор/Средство просмотра. Определяет, как изменяется значение поля. Например, поле блока данных 49 может быть изменено при помощи *редактора изображения, редактора звука* или типового *редактора файлов*.	editor

Более подробную информацию о свойствах поля см. здесь 49. Данное свойство доступно через переменную fields 1486.

20.17.1.3.3 Пользовательские свойства

Данное свойство определяет, как общая таблица используется в качестве пользовательского свойства 2177.

Описание поля	Имя поля
Использовать общую таблицу в качестве пользовательского свойства. Если включено, данная общая таблица добавляется в качестве пользовательского свойства в контексты согласно выражению пригодности.	enabled
Выражение пригодности. Определяет, к какому контексту (контекстам) добавляется данное пользовательское свойство. Для более подробной информации см. Пригодность ресурса 2177.	validityExpression
Правила обновления пригодности. Список контекстных масок и имен событий. Если возникает событие, обусловленное полем **Событие**, в любом контексте, совпадающем с маской, указанной в поле **Маска** той же записи, **выражение пригодности** будет заново рассчитано для данного контекста. Это позволяет добавлять/удалять пользовательские свойства из некоторых контекстов при возникновении некоторых изменений.	validityListeners

Данное свойство доступно через переменную customProperties 1487.

20.17.1.4 Права доступа к записям общей таблицы

Существует метод назначения прав доступа к каждой записи в общей таблице. Данная опция называется "Редактировать права доступа" и может быть включена или отключена для определенной таблицы посредством изменения основных свойств 2179 таблицы.

Права доступа к записи определяют, какие контексты 41 AtomMind Server имеют доступ к данной записи. Данная опция используется в мастер-свойствах членов группы 2178. Когда контекст группы 751 или терминала данных 497 извлекает содержание общей таблицы, он получает только те записи, к которым у него есть доступ.

Пример:

Предположим, что общая таблица содержит контрольный список действий, которые должны быть выполнены на определенном аппаратном устройстве при его запуске. Данный список записан на каждое устройство, используя интеграцию с устройствами, что приводит к проверке оператором всех элементов во время запуска устройства. Разумеется, некоторые из них могут потребовать такое действие, как "Авторизуйтесь при помощи вашей смарт карты" (для системы контроля транспорта), но только некоторые из них могут потребовать действие, такое как "Проверьте отсутствие утечки в гидравлической системе". В таких случаях права доступа к каждой записи являются очень полезными.

Для каждой записи в этой гипотетической контрольной таблице мы можем назначить список Device (точнее, *контекстов* Device), к которым она применяется. Мы также можем определить их в группы 751 (см. ниже). Определенный элемент проверки (запись) будет записан только на те Device, у которых есть права доступа к данной таблице.

Права доступа к каждой записи общей таблицы отличаются от таблиц прав доступа 487 пользователя 478. Права доступа пользователя определяют доступные для него контексты и какие операции он может выполнять в каждом контексте. Напротив, права доступа к записям общей таблицы определяют, какие контексты имеют доступ к данной записи. На данный момент это применимо только в следующих сценариях:

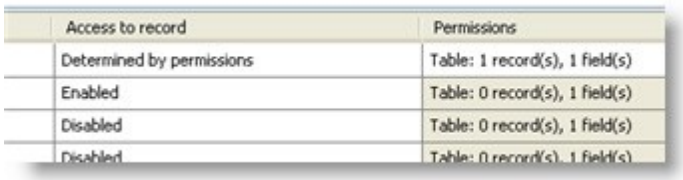
Для **интеграции с группами** список прав доступа может содержать только групповые контексты 1529. Только группы, обозначенные в списке, могут копировать данную запись для своих членов.

Для **интеграции с терминалами данных** список прав доступа может содержать контексты терминалов данных 1494 и контексты группы 1529 групп терминалов данных. В первом случае запись может быть доступна для определенного терминала данных. Во втором случае для всех терминалов данных в группе.

Если вы хотите обезопасить таблицу, вам следует проконтролировать, кто из пользователей может вносить изменения в саму таблицу - устанавливайте права не только на доступ к записям, но и на саму таблицу.

Когда включены права доступа к записи, каждая запись в таблице содержит два дополнительных поля:

- Доступ к записи
- Права доступа



Поле **Доступ к записи** имеет три возможных значения: **включен**, **выключен** и **определяется правами доступа**. Когда доступ **включен**, права доступа к данной записи выключаются, и каждая группа или терминал данных имеют к ней доступ. Когда доступ **отключен**, группы и Device не получат данную запись при просмотре содержания таблицы. Когда значение поля установлено на о**пределяется правами доступа**, вы можете задать список групп или Device, для которых данная запись будет доступна. Если пользователю разрешено изменять саму таблицу, он также может модифицировать права доступа к каждой записи.

Когда свойство интеграции с Device используют для работы с общей таблицей, список прав доступа может содержать не только обычные контексты терминала данных, но и группы терминалов данных. При добавлении группы терминала данных в список прав доступа определенной записи общей таблицы, все его члены будут иметь доступ к данной записи.

20.17.1.5 Производительность общих таблиц

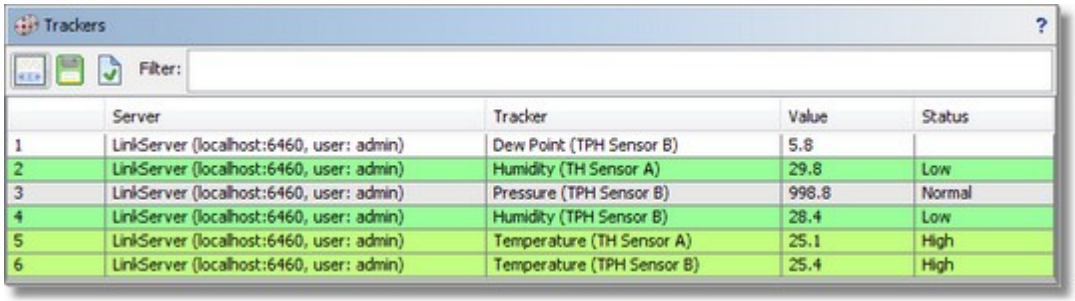
Общие таблицы требуют столько памяти, сколько необходимо для хранения всех данных таблицы. Заметьте, что если таблица используется как пользовательское свойство 2177, создается множество экземпляров таблицы, требующих большей памяти.

Таблицы, используемые как пользовательские свойства, могут в очень редких случаях вызывать значительную нагрузку процессора, если переоценка Выражения Пригодности таблицы часто активируется Правилами Обновления Пригодности таблицы. См. производительность выражения 141 для оценки воздействия на производительность.

20.17.2 Датчики

Датчики помогают отслеживать данные сервера для решения критически важных задач в реальном времени. Все датчики сведены в единую таблицу (например, окно Датчики 414 в AtomMind Client)с целью обеспечения эффективного мониторинга. Для обозначения *состояния* датчика используется цветовое выделение.

Ниже приведена таблица датчиков в AtomMind Client:



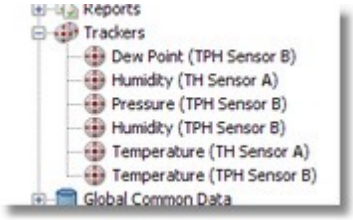
Основу датчика составляет *отслеживаемое значение*, написанное на языке выражений 112 AtomMind. Данное выражение может ссылаться на данные сервера или данные, полученные из терминалов данных 497 аппаратных устройств. Простое отслеживаемое выражение может быть использовано для мониторинга некоторого значения напрямую, тогда как более сложное выражение позволяет отследить комбинации нескольких значений. Например, датчик может быть легко установлен для наблюдения за температурой конденсации, рассчитываемой исходя из величин температуры и абсолютной влажности от разных сенсоров.

Каждый пользователь 478 имеет свой набор датчиков.

Значение выражения каждого активного датчика периодически пересчитывается в соответствии со свойством датчика Период проверки 2183.

Администрирование датчиков

Для администрирования датчиков используются два контекста: общий контекст Датчики 1593, который выступает в роли контейнера, и контекст Датчик 1596, который содержит информацию об одном датчике.



20.17.2.1 Статус датчика

Статус датчика позволяет системным операторам быстро проверить состояние датчика, выделив его в списке датчиков. Статусы помогают определить датчики, обнаруживающие предупреждения и критические условия контролируемых значений. Для каждого датчика можно назначить неограниченное количество определяемых пользователем состояний.

Свойство "Таблица статусов 2183" используется для определения текущего статуса датчика. Статус определяется один раз за период проверки 2183, сразу же после расчета **отслеживаемого выражения**. Таблица статусов датчика просматривается построчно, начиная с самой верхней строки. Происходит оценка **выражения**, заданного в текущей строке таблицы. Выражение статуса должно содержать специальную ссылку 117 `{tracker/}`, указывающую на значение **отслеживаемого выражения**. Если результат является TRUE, статус датчика меняется на указанный в только что проверенной строке таблицы статусов. Если результат является FALSE, обрабатывается следующая строка таблицы. Если таблица статусов пуста или ни одно выражение не разрешилось в TRUE, статус считается *неопределенным*.

При изменении статуса датчика его **описание** появляется в списке датчиков и выделяется цветом, указанным в таблице статусов (при наличии). В это же время формируется событие Изменение состояния 1598. История 73 данного события может быть использована для просмотра прошлых изменений статусов при помощи действия Контролировать связанные события 109 контекста датчика 1596.

Далее приведен пример таблицы статусов для простого температурного датчика (скриншот сделан в AtomMind Client):

	Description	Expression (use {tracker/} to refer tracker value)	Color	Level of state change event
1	Extremely Low	{tracker/} < 15		Warning
2	Very Low	{tracker/} < 19		Warning
3	Rather Low	{tracker/} < 22		Info
4	Low	{tracker/} < 24		Info
5	Moderate	{tracker/} < 25		Info
6	High	{tracker/} < 26		Info
7	Rather High	{tracker/} < 27		Info
8	Very High	{tracker/} < 28		Warning
9	Extremely High	{tracker/} < 30		Warning
10	Critical	true		Error

Согласно данной таблице, статус датчика будет равен "Чрезвычайно низкая", если температура ниже 15 градусов, "Очень низкая", если температура держится между 15 и 19 градусами, и .т.д.

20.17.2.2 Конфигурация датчика

Данный раздел охватывает свойства конфигурации датчика.

Все свойства, описанные в разделе далее, доступны через действие конфигурировать 1051 в контексте датчика 1596.

20.17.2.2.1 Свойства датчика

Данное свойство определяет основные опции датчика.

Описание поля	Имя поля
Имя датчика. Имя контекста датчика. Должно соответствовать соглашениям о наименовании 42 контекстов. Имя необходимо для ссылки на датчик из других частей системы.	name
Описание датчика. Текстовое описание датчика. Также является описанием 43 контекста датчика.	description
Отслеживаемое выражение. Выражение, которое необходимо отследить.	expression
Активный. Если датчик выключен, периодические расчеты его значения и статуса не выполняются.	enabled
Период проверки. Определяет, как часто происходит перерасчет значения и статуса датчика.	period
Скрыть, если статус не определен. Если на данный момент датчик не имеет установленный пользователем статус 2183 (т.е. ни одно выражение статуса не разрешилось в TRUE), он будет скрыт в списке датчиков, если флажок установлен.	hideIfNoStatus
Время хранения истории. Устанавливает период окончания хранения необработанной истории датчика. Технически оно определяет, как долго изменения переменной датчика будут храниться в базе данных.  Другим способом хранения истории датчика является использование статистики датчика 2184.	historyStorageTime

Данные свойства доступны через переменную childInfo 1597.

20.17.2.2.2 Таблица статусов

Данное свойство определяет правила расчета статуса 2182 датчика и его характеристики.

Описание поля	Имя поля

Описание. Текстовое описание статуса датчика.	description
Выражение Выражение, которое должно быть true, чтобы активировать статус.	expression
Цвет. Цвет, используемый для выделения датчика, когда его статус активен.	color
Уровень события изменения статуса. Серьезность 73 события изменение статуса 1598, которое формируется при смене статуса.	level

Таблица статусов доступна через переменную statusTable 1597.

20.17.2.3 Статистика датчика

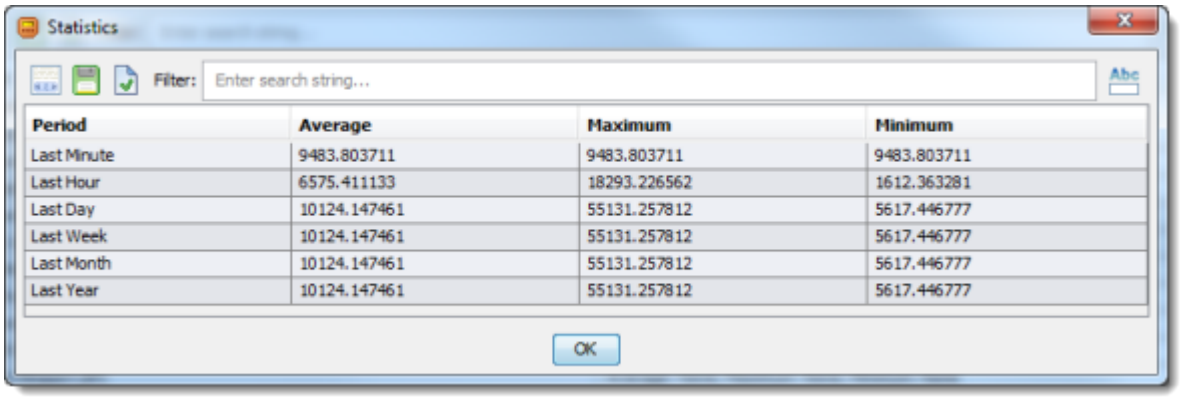
Каждый датчик имеет статистический канал 718, который способствует длительному хранению истории для датчиков, представляющих числовые значения.

Если новый канал добавляется в таблицу Статистических Каналов датчика, этот канал заранее конфигурируется для перевода значения датчика в число и собирает его статистику.

Просмотр статистики датчика

Для просмотра краткой статистики датчика запустите действие "Посмотреть статус 111" ( ) и перейдите во вкладку "Статистика".

Краткая статистика датчика в AtomMind Client выглядит следующим образом:



Просмотр подробной статистики

Для просмотра подробной статистики используйте глобальное действие Посмотреть статистику 1562.

20.17.2.4 Безопасность датчиков

Отслеживаемое выражение датчика оценивается при наличии прав доступа 477 владельца датчика. Таким образом, датчик может ссылаться только на данные, доступные владельцу.

Если вы создаете копию определенного датчика под другой учетной записью пользователя 478, копия может не функционировать нормально, если новый владелец датчика не имеет прав доступа к данным **Отслеживаемого выражения** клонированного датчика.

20.17.2.5 Производительность датчиков

Нагрузка процессора и использование памяти, вызванные датчиком, полностью определяются процессором и воздействием на память 141 Отслеживаемого Выражения и выражения Статуса.

Датчики должны использоваться только для расчета значений, которые постоянно контролируются системными операторами. Использование датчиков для предварительного расчета значений с целью их отображения на виджетах 943 или панелях инструментов 912 - это плохая идея, поскольку значение датчика периодически пересчитывается, даже когда его значение не отображается, вызывая потерю производительности. Чтобы избежать этого, избавьтесь от датчиков, используя один из этих способов:

- Для одноразовых датчиков просто скопируйте отслеживаемое выражение датчика в выражение привязки[1296] виджета, которое до этого ссылалось на значение датчика
- Для повторно используемых датчиков определите модель[810] с несколькими функциями[818], производящую необходимые расчеты и ссылающуюся на эти функции из ваших выражений привязки виджета

20.17.2.6 Примеры датчиков

Эта статья описывает простые выражения датчика.

Подсчет устройств

Отслеживаемое выражение: `aggregate("users.*.devices.*", "{env/previous} + ({.:status$connectionStatus} == 1 ? 1 : 0)", 0)`

Это выражение использует функцию `aggregate()` для отображения всех контекстов устройств (т.е., всех контекстов, соответствующих маске[44] `users.*.devices.*`) и подсчета тех, у которых есть поле `connectionStatus` переменной `status`, равной `1` (т.е., "онлайн").

Оно возвращает количество устройств в режиме онлайн, доступных пользователю, владеющему датчиком. Таким образом, если оно выполняется с правами доступа администратора по умолчанию[479], то возвращает общее количество устройств системы в режиме онлайн.

Подсчет использования памяти AtomMind Server

Отслеживаемое выражение: `round(({:status$totalMemory} - {:status$freeMemory}) * 100 / {:status$maxMemory})`

Во-первых, объем памяти, используемый Java VM в настоящее время, рассчитывается путем вычитания свободной части из текущего объема.

Во-вторых, полученное число делится на максимальный размер памяти, который повзоляет разместить JVM. Это дает соответствующее использование памяти, которое умножается на 100 для расчета процента.

20.17.2.7 Встроенные датчики

Некоторые датчики встроены в AtomMind Server и появляются под учетной записью администратора по умолчанию[479]:

- **Общее количество пользователей.** Показывает количество учетных записей пользователей[478] в системе.
- **Общее количество устройств.** Показывает количество Device в системе.
- **Подключенные Devices.** Показывает количество подключенных Device.
- **Отключенные Devices.** Показывает количество отключенных Device.
- **Неактивные Devices.** Показывает количество неактивных Device.
- **Использование памяти AtomMind Server**. Показывает память, используемую AtomMind Server в виде процентного отношения от всей доступной памяти согласно конфигурации запуска виртуальной Java-машины (JVM). Использование отображается в виде графика "пилообразной волны", т.к. она возрастает при нормальной работе сервера и мгновенно падает во время *сбора мусора* в JVM.

Сбор мусора представляет собой автоматическое управление памятью. Сборщик мусора периодически убирает мусор или очищает память, занятую объектами, которые больше не используются программой.

20.17.2.8 Датчики в распределенной архитектуре

Этот раздел описывает особенности поведения датчиков в распределенной инсталляции[1332] AtomMind.

Чтобы датчик, подключенный из сервера-поставщика, появился в таблице датчиков клиентов, подключенных к серверу-потребителю, убедитесь, что путь контекста подсоединенного на сервере-потребителе контекста датчика соответствует маске контекста `users.*.trackers.*`, т.е. что удаленный датчик появился среди локальных датчиков.

20.17.3 Избранное

Интерфейс AtomMind многофункционален, и его дерево контекстов 41 может быть комплексным и глубоким. Для упрощения навигации в него было добавлено средство "Избранное".

"Избранное" позволяет пользователям AtomMind выполнять распространенные действия 87 одним нажатием кнопки мыши. Вам не придется "копаться" в дереве контекстов - вы будете иметь быстрый доступ к тому, что вы используете чаще всего. Средство "Избранное" функционирует как "ярлыки" или "быстрые ссылки", указывающие на различные контексты. В "Избранное" можно легко поместить любой пользовательский интерфейс AtomMind. Например, "Избранное" в AtomMind Client выглядит следующим образом:

	Description	Server	Context Mask	Action
1	Edit Template in Query XXX	LinkServer (192.168.1.2:6460, user: admin)	Query XXX	Edit Template
2	My Alert Configuration	LinkServer (192.168.1.2:6460, user: admin)	AuthKey Problem	Configure Alert
3	Buzz in admin.c6 : Auto-registered Device Server 2	LinkServer (192.168.1.2:6460, user: admin)	admin.c6 : Auto-registered Device Server 2	Buzz
4	Create New Table in User Common Data	LinkServer (192.168.1.2:6460, user: admin)	User Common Data	Create New Table
5	View Device Servers Summary in Device Servers	LinkServer (192.168.1.2:6460, user: admin)	Device Servers	View Device Servers Summary
6	Show Report in Checklist by Vehicle	LinkServer (localhost:6460, user: test)	Checklist by Vehicle	Show Report
7	Perform self-test in admin.agent.0_0 : TR510 #0 on port 0	LinkServer (localhost:6460, user: test)	admin.agent.0_0 : TR510 #0 on port 0	Perform self-test

В таблице отображаются следующие столбцы:

Номер: Номер строки избранного действия в таблице.

Описание: Текстовое описание запускаемого действия (может быть изменено пользователем).

Сервер: Сервер, на котором сконфигурировано избранное действие.

Маска контекстов: Маска контекстов 44, к которой применяется избранное действие, или описание контекста 41, если маска разрешается в один контекст (т.е. не содержит специальные символы).

Действие: Описание действия, которое будет выполняться по контекстной маске при запуске избранного действия.

Каждая строка в таблице является ссылкой. При нажатии на любую строку запускается элемент "Избранного". Вы можете выбрать любой столбец.

В AtomMind Client, когда рабочее пространство 363 содержит несколько соединений с сервером, итоговая таблица "Избранное" включает все избранные действия, доступные для всех учетных записей. По существу избранные действия собираются из всех соединений AtomMind Server и отображаются в одной общей таблице в AtomMind Client. Это позволит вам пользоваться только одной таблицей "Избранное" для навигации по всему AtomMind.

В отличие от действий, загружаемых "напрямую" (например, из системного дерева 370 AtomMind Client), действия, запуск которых был осуществлен при помощи средства "Избранное", могут использовать пользовательские параметры выполнения 102.

Каждый пользователь 478 имеет свой набор действий в средстве "Избранное".

Сопутствующая документация: "Выполнение действия для множества устройств" 1636

Администрирование "Избранного"

Для администрирования "Избранного" используются два контекста 41: общий контекст "Избранное 1524", который выступает в роли контейнера, и контекст "Избранный 1526", который содержит информацию о конкретном избранном элементе.

Запуск виджетов, отчетов и инструментальных панелей через "Избранное"

Виджеты 943, отчеты 928, инструментальные панели 912 и некоторые другие системные объекты могут быть *абсолютными* и *относительными*. Абсолютный объект запускается сам по себе, относительные объекты запускаются в **определенном контексте** 41, который служит источником данных.

Таким образом, добавление действия запуска относительного объекта напрямую в "Избранное" не позволит системе распознать, для какого контекста оно должно быть запущено при нажатии ссылки. Вместо этого действие запуска объекта, **расположенное в целевом контексте**, должно быть добавлено в "Избранное".

Пример: Предположим, что мы имеем относительный виджет **"График трафика"**, который является пригодным для всех сетевых устройств[497]. Вы можете добавить график трафика **определенного устройства** в "Избранное". Для этого нужно сделать следующее:

- Перетащите выбранное устройство в узел "Избранное".
- Выберите "**График трафика**" в диалоговом окне предлагаемых действий.

20.17.3.1 Настройка избранного

Каждое избранное действие характеризуется несколькими свойствами:

Описание поля	Имя поля
Описание. Текстовое описание избранного действия. Также является описанием[43] контекста "Избранное".	description
Маска[44] **контекстов.** Определяет область применения действия избранного. При выполнении избранное действие запускается из всех контекстов, соответствующих маске.	mask
Действие. Имя запускаемого действия.	action
Активный. Избранное действие также может быть отключено (скрыто из списка избранных).	enabled
Параметры. Параметры выполнения[102] действия.	executionParameters

Данные свойства доступны через переменную childInfo[1471].

20.17.3.2 Избранное в распределенной архитектуре

Этот раздел описывает особенности поведения "Избранного" в распределенной инсталляции[1332] AtomMind.

Чтобы "Избранное", подключенное с сервера-поставщика, появилось в таблице "Избранное" клиентов, подключенных к серверу-потребителю, убедитесь, что путь контекста подключенного контекста "Избранное" на сервере-потребителе соответствует маске контекстов `users.*.favourites.*`, т.е. удаленное "Избранное" появляется наряду с локальным "Избранным".