ООО «ТриниДата»

РУКОВОДСТВО АДМИНИСТРАТОРА

платформы виртуализации данных АрхиГраф.MDM

Екатеринбург 2022

Содержание

Основные сведения

АрхиГраф.MDM – платформа виртуализации данных, которая может быть использована и в качестве классической MDM-системы. Общая архитектура платформы показана на рис. 1.

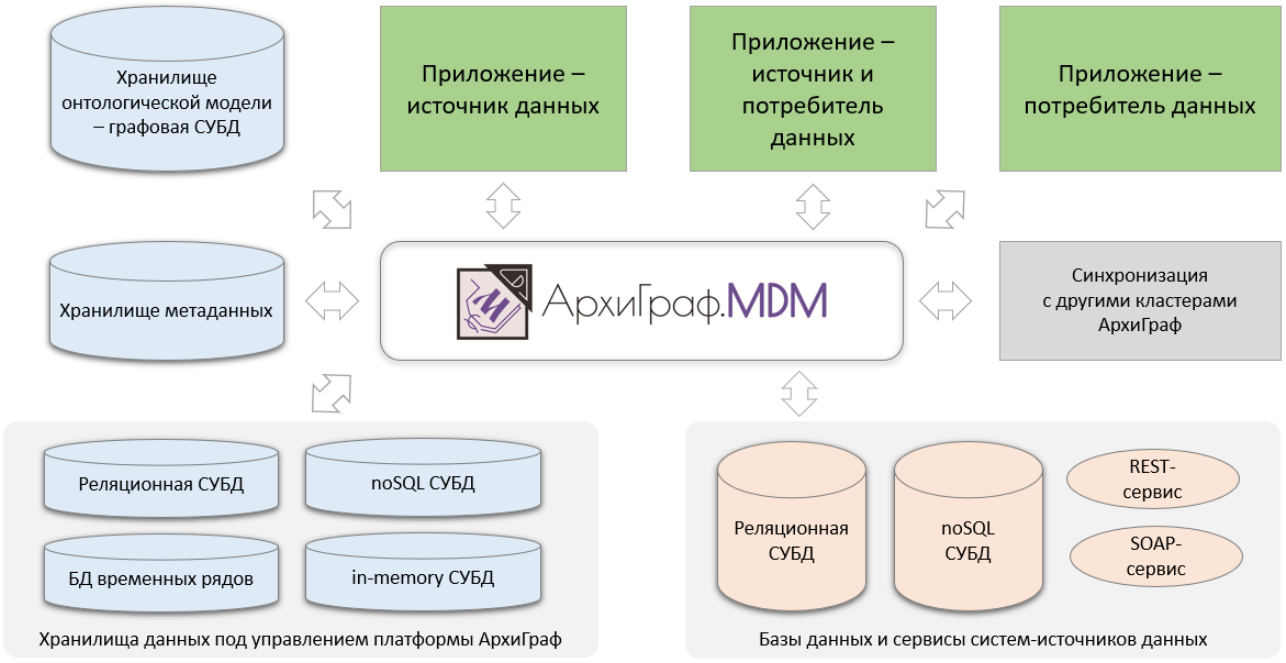

Рис. 1. Архитектура АрхиГраф.MDM

Платформа АрхиГраф.MDM представляет промежуточный слой (middleware), предоставляющий сервисы для работы с данными, расположенными во множестве различных источников как под управлением платформы, так и вне ее. Сервисы предоставляются как в синхронном варианте (HTTP-запросы, REST-сервисы), так и в асинхронном (обмен через очереди Kafka, RabbitMQ).

АрхиГраф.MDM использует онтологическую модель для описания структуры обрабатываемых данных. Онтологическая модель размещена в графовой СУБД (продукт класса RDF Triple Store), тогда как сами данные располагаются в различных базах данных, подключенных к платформе.

Создание и редактирование онтологической модели выполняется при помощи редактора АрхиГраф.Мир[1]. Наполнение данными может выполняться вручную при помощи АрхиГраф.Мир или с помощью специально разработанных

[1] Подробнее о редакторе онтологий АрхиГраф.Мир на сайте: https://trinidata.ru/archigraph_mir.htm

пользовательских интерфейсов, а также средствами автоматизации обмена информацией, использующими API АрхиГраф.MDM.

Задачи администрирования АрхиГраф.MDM состоят в:

- Настройке «точек доступа» – отдельных пространств данных, каждое из которых включает онтологическую модель и набор хранилищ информации.
- Конфигурировании хранилищ данных.
- Управлении распределением объектов данных, относящихся к различным классам, между хранилищами.
- Создании учетных записей для клиентов MDM и настройке прав доступа на уровне классов и атрибутов онтологической модели.

Для понимания принципов работы с содержимым баз данных под управлением АрхиГраф.MDM желательно познакомиться с концепциями семантического (онтологического) моделирования. В качестве введения в эту проблематику рекомендуем методическое руководство, разработанное компанией ТриниДата: http://trinidata.ru/files/SemanticIntro.pdf.

В АрхиГраф.MDM предусмотрены два инструмента конфигурирования: визуальный конфигуратор и консольная утилита mdmctl, предназначенная в первую очередь для создания сценариев автоматизированного развертывания.

Визуальный интерфейс конфигурирования АрхиГраф.MDM

Управление точками доступа

Страница управления точками доступа имеет такой вид:

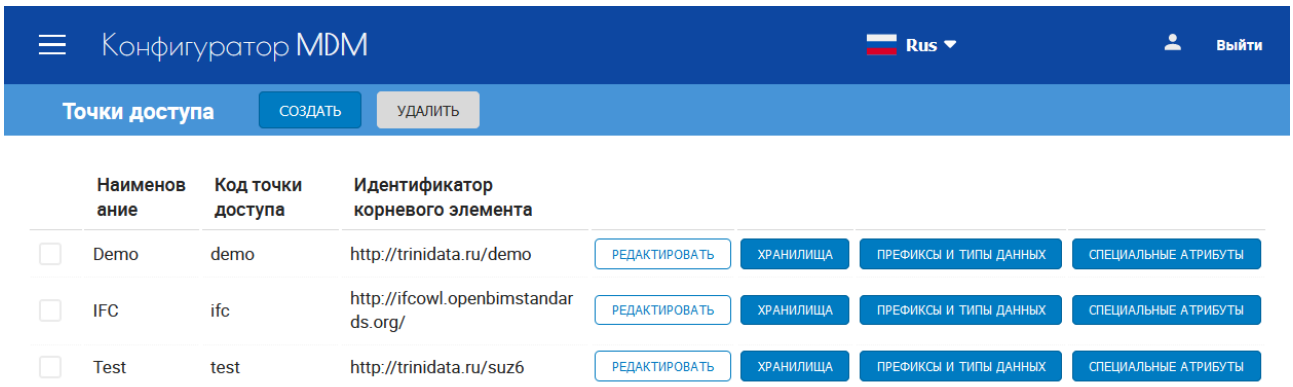

Рис. 2. Управление точками доступа

Отображается список точек доступа, основные свойства каждой из них:

- читаемое наименование,
- код, используемый при обращении через API,
- идентификатор корневого элемента онтологии (его подклассами должны быть классы верхнего уровня онтологии, рекомендуем использовать http://www.w3.org/2002/07/owl#Thing; указывать идентификатор нужно в полном виде, не сокращая префикс).

Отметив одну или несколько точек доступа переключателями и нажав кнопку «удалить», можно удалить точки доступа. Нажатие кнопки «Создать» открывает диалоговое окно создания новой точки доступа, в котором необходимо ввести значения трех основных свойств точки доступа. Значения тех же свойств можно изменить для любой существующей точки доступа, нажав кнопку «Отредактировать».

Нажатие на кнопку «**Префиксы и типы данных**» открывает страницу, на которой можно настроить набор префиксов, которые будет поддерживать для этой точки доступа редактор АрхиГраф.Мир (а также префикс по умолчанию, который поддерживает MDM), и набор типов данных, специфических для этой точки доступа.

Область работы с префиксами выглядит так:

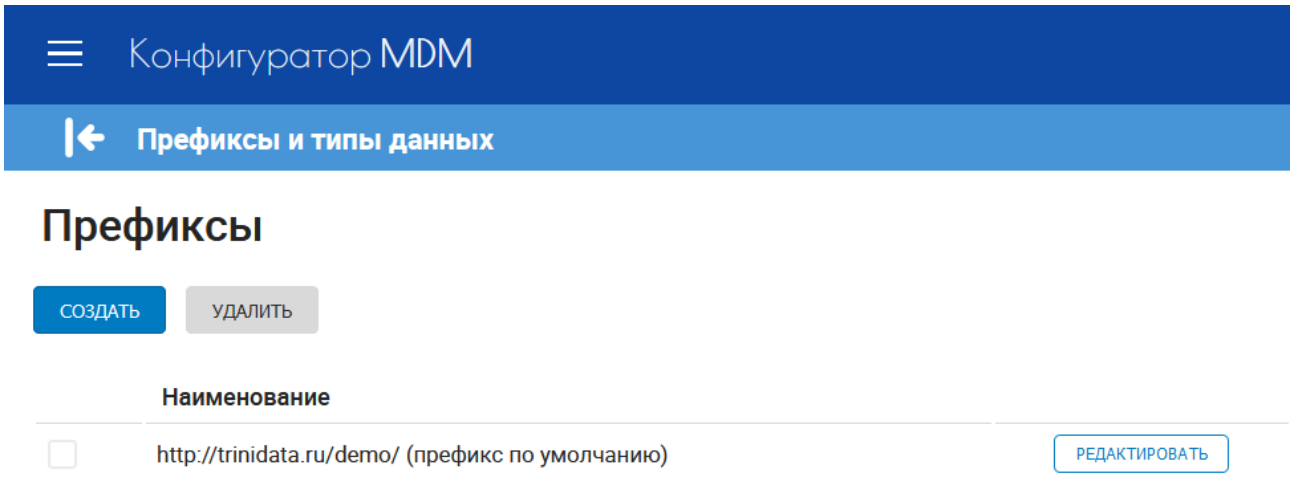

Рис. 3. Список префиксов, определенных для точки доступа

Кнопки «Создать», «Удалить», «Редактировать» предназначены для выполнения операций с элементами списка префиксов. Первый префикс в списке является префиксом по умолчанию для данной точки доступа. Это означает, что MDM будет сокращать этот префикс при работе через API. Например, если префикс по умолчанию – http://trinidata.ru/demo/, то идентификаторы элементов

онтологии http://trinidata.ru/demo/Персона и Персона будут эквивалентны при обращении к API MDM и при работе в редакторе АрхиГраф.Мир.

Остальные префиксы имеют значение только для АрхиГраф.Мир. Редактор онтологии предлагает выбрать один из этих префиксов при создании любого элемента онтологии.

Настройка типов данных заключается в указании наименования и идентификатора для каждого типа. Например, можно указать наименование «Any URI» и идентификатор xsd:anyURI – после этого можно будет присваивать литеральным атрибутам онтологии такой тип данных.

Нажатие на кнопку «**Специальные атрибуты**» в списке точек доступа открывает страницу, на которой можно управлять набором атрибутов, определенных вне онтологии. Эта возможность полезна, например, для создания атрибутов, значениями которых могут обладать классы или свойства. Вид диалогового окна создания специального атрибута показан на следующем рисунке.

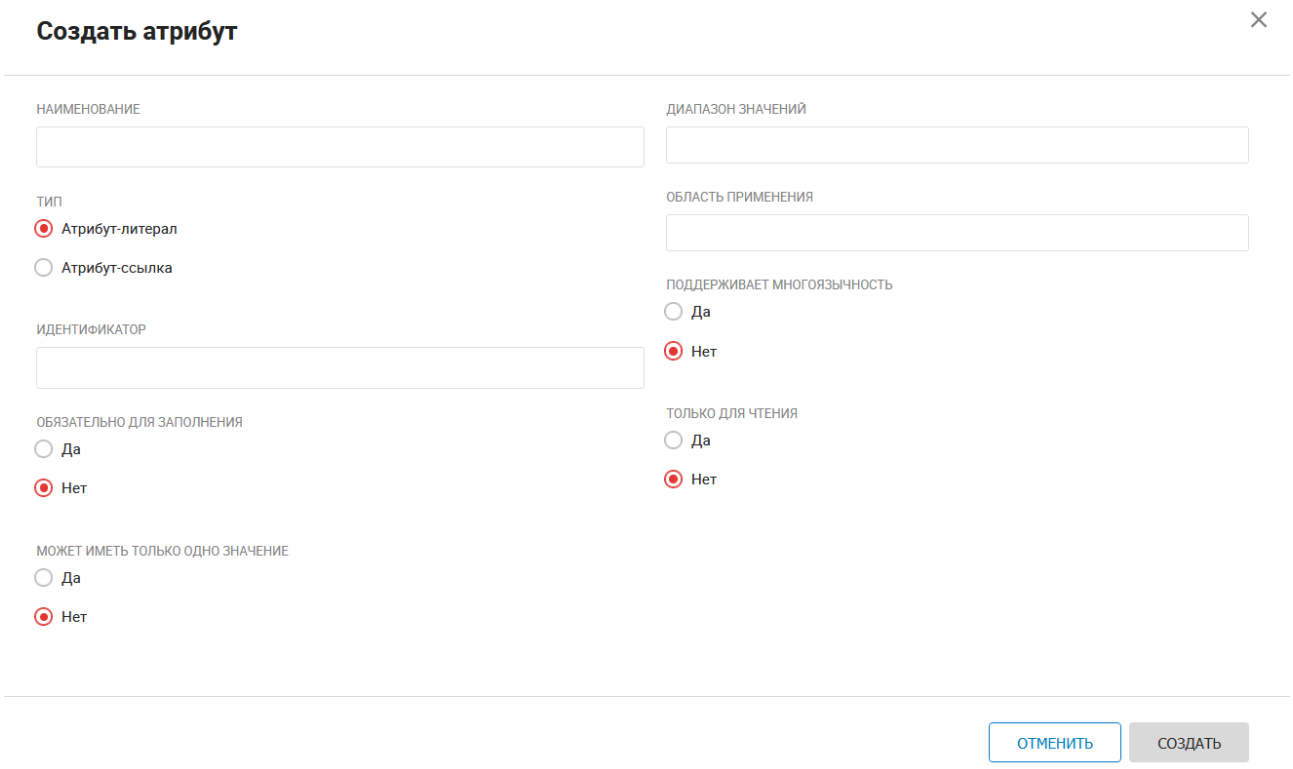

Рис. 4. Диалоговое окно создания специального атрибута

Поля свойств атрибута имеют следующий смысл:

- Наименование – читаемое название атрибута

- Тип – атрибут-литерал или атрибут-ссылка, owl:DatatypeProperty или owl:ObjectProperty.
- Идентификатор – полный URI, идентификатор создаваемого атрибута.
- Область применения – класс или тип сущности, к которому применим атрибут (например, выберите owl:Class, чтобы создать атрибут, применимый ко всем классам онтологии).
- Диапазон значений – для атрибутов-ссылок это один или несколько классов онтологии, из числа объектов которых выбираются значения для данного свойства. Для атрибутов-литералов это один из доступных типов данных xsd.
- Поддерживает многоязычность, Обязательно для заполнения, Может иметь только одно значение (для каждого объекта), Только для чтения – дополнительные свойства атрибута.

Настройка хранилищ

Хранилище – это таблица или коллекция базы данных, в которой размещается информация об объектах определенного класса(-ов) онтологии. Чтение и запись в хранилища выполняет MDM, они не должны быть доступны напрямую для других программных компонентов.

Хранилища настраиваются отдельно для каждой точки доступа, переход к их настройке происходит нажатием кнопки «Хранилища» в разделе «Точки доступа». Список хранилищ точки доступа имеет такой вид:

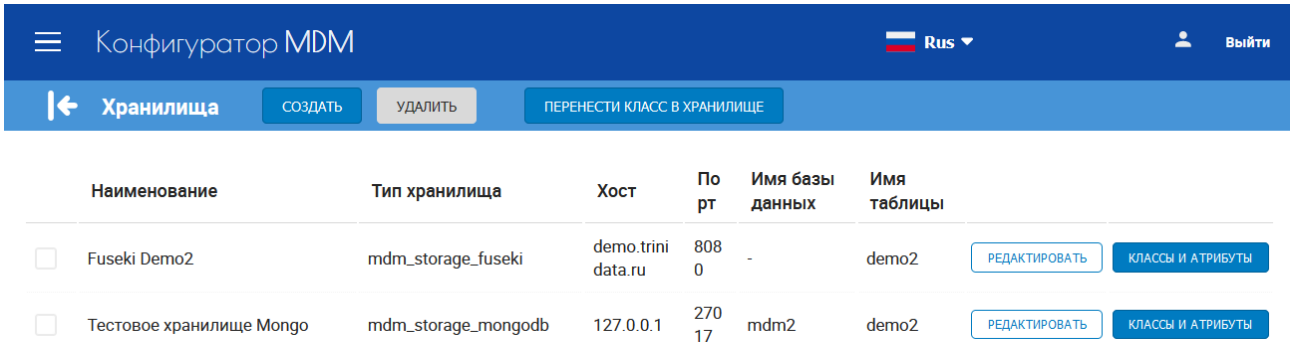

Рис. 5. Список хранилищ для точки доступа

В списке отображаются основные свойства каждого хранилища, а также кнопки «Редактировать» и «Классы и атрибуты». Диалоговое окно создания/редактирования хранилища выглядит так:

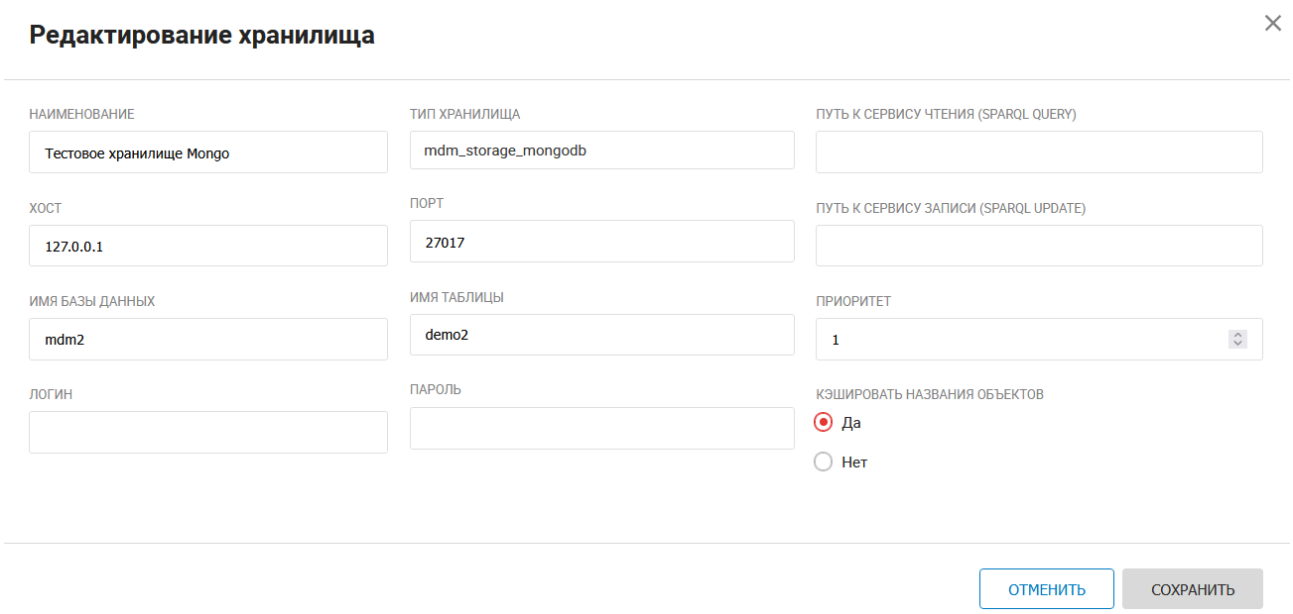

Рис. 6. Окно создания/редактирования хранилища

Свойства хранилища:

- Наименование – читаемое название для отображения в списке
- Тип хранилища – выбирается один из типов, поддерживаемых MDM. Фактически это выбор внутреннего «драйвера» для подключения к базе данных определенного типа.
- Хост, порт, логин, пароль, имя базы данных, имя таблицы – реквизиты подключения к СУБД.
- Путь к сервису чтения/записи – специфичные реквизиты для случая, когда хранилище является точкой доступа SPARQL. Значения этих полей обычно имеют вид «/fuseki/[dataset]/query» и «/fuseki/[dataset]/update» (для Apache Fuseki). В этом случае поле «имя базы данных» надо оставить пустым, а в поле «имя таблицы» указать имя датасета.
- Приоритет – число, определяющее порядок, в котором MDM ищет объект по идентификатору в хранилищах в случае, если класс объекта не известен. Хранилища с наибольшим значением приоритета просматриваются первыми.
- Кэшировать названия объектов – флаг, определяющий, что в данном хранилище для ссылок на другие объекты сохраняются читаемые названия этих объектов, а не только идентификаторы. Кэширование названий существенно ускоряет извлечение данных из хранилищ.

После того, как основные свойства хранилища настроены, нужно указать, объекты каких классов будут в нем храниться, и как свойства объектов будут спроецированы в структуру данных хранилища. Для этого предназначена вкладка «Классы и атрибуты», которая имеет следующий вид:

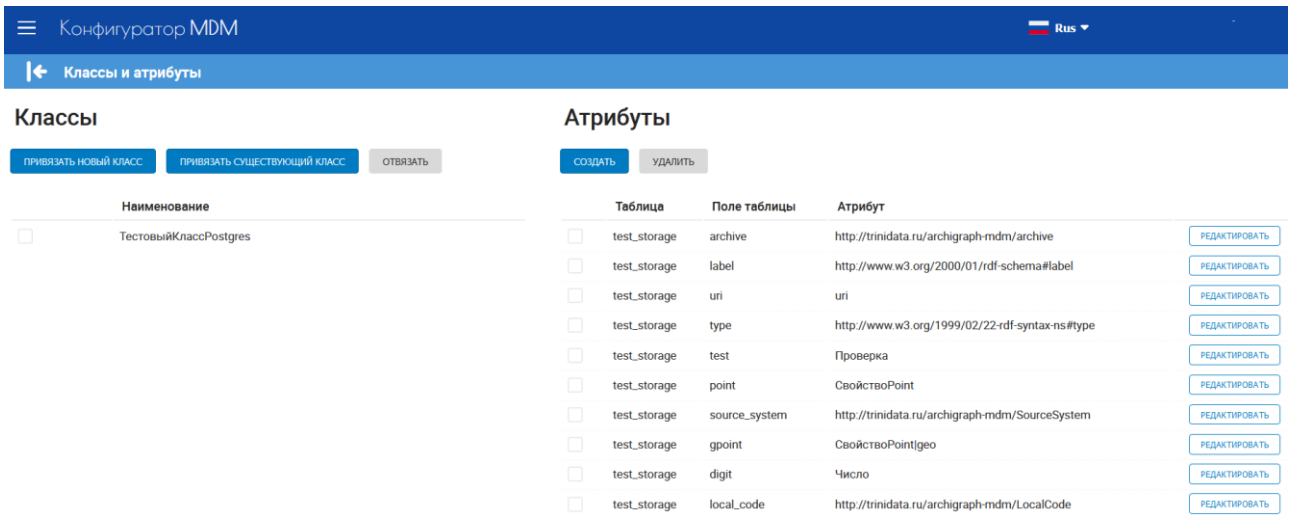

Рис. 7. Вкладка «Классы и атрибуты» в свойствах хранилища

Кнопка «Привязать существующий класс» позволяет выбрать класс онтологии и привязать его к данному хранилищу. Кнопка «Создать новый класс» позволяет создать новый класс, не переходя в интерфейс АрхиГраф.Мир, и сразу привязать его. К хранилищу можно привязать любое количество классов, но мы рекомендуем делать это только в случае, если объекты данных классов имеют похожие наборы свойств. В этом случае можно создать в хранилище индексы по столбцам/элементам коллекций, которые хранят значения свойств, по значениям которых наиболее часто будут выбираться объекты из хранилища.

В области «Атрибуты» выполняется настройка соответствия свойств онтологической модели столбцам/элементам коллекции хранилища. Диалоговое окно настройки выглядит так:

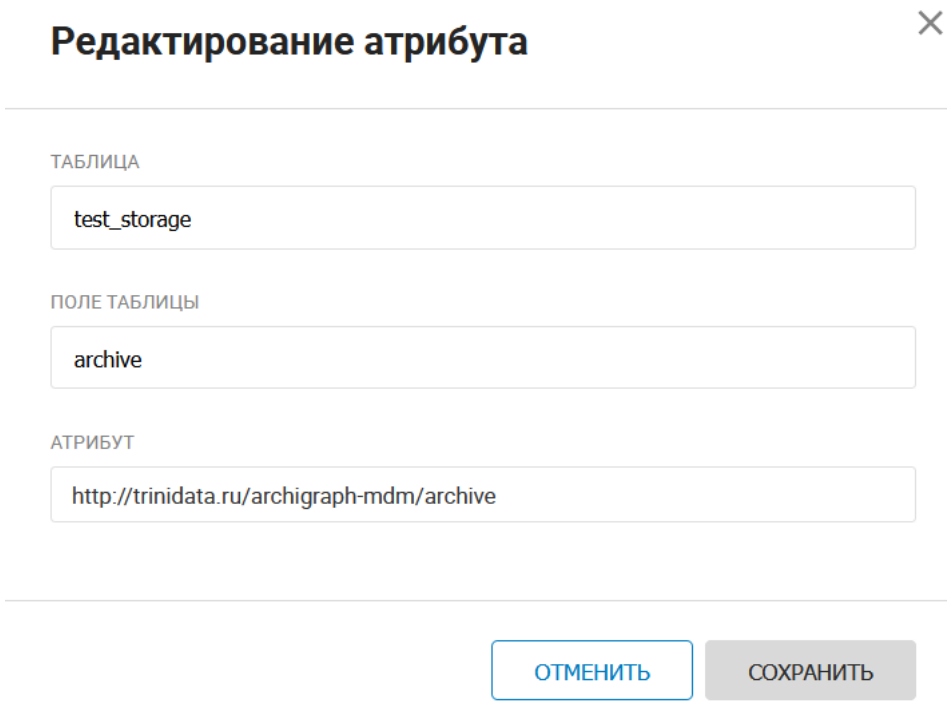

Рис. 8. Окно редактирования способа хранения атрибута онтологии в хранилище

Для хранилищ MongoDB и различных видов точек доступа SPARQL настраивать способ хранения атрибутов не требуется. Для хранилищ типа postgresql_jsonb и postgresql всегда должны быть создан следующий минимальный набор столбцов в таблице и правил соответствия:

Поле таблицы	Тип поля	Атрибут онтологии
uri	character varying(255) NOT NULL (первичный ключ)	uri
data	jsonb (этот столбец нужен только для хранилища postgresql_jsonb)	data
type	character varying(255)[]	http://www.w3.org/1999/02/22-rdf-syntax-ns#type
archive	character varying(8)	http://trinidata.ru/archigraph-mdm/archive
name	character varying(512)	http://www.w3.org/2000/01/rdf-schema#label
local_code	character varying(255)	http://trinidata.ru/archigraph-mdm/LocalCode
source_system	character varying(255)	http://trinidata.ru/archigraph-mdm/SourceSystem
is_defined_by	character varying(255)[]	http://www.w3.org/2000/01/rdf-schema#isDefinedBy
duplicate	character varying(255)[]	http://trinidata.ru/archigraph-mdm/duplicate

Кроме этих столбцов, могут быть созданы и столбцы для хранения любых других свойств объектов. Для хранилища типа postgresql создание таких столбцов обязательно, для хранилища типа postgresql_jsonb – опционально: для него имеет смысл создавать столбцы для тех свойств, по значениям которых в Postgres нужно построить индексы для ускорения выборки данных. Значения всех свойств в любом случае будут храниться в столбце data.

Автоматическая настройка хранилищ

Кнопка «Перенести класс в хранилище» на странице «Хранилища» позволяет выполнить операцию создания и настройки хранилища типа postgresql или postgresql_jsonb в полуавтоматическом режиме. Последовательность действий после нажатия на эту кнопку такова.

1. Появляется диалог, в котором пользователь вводит наименование хранилища, выбирает класс модели и тип хранилища.

2. Пользователь вводит реквизиты хранилища: хост, порт, имя базы, имя таблицы, логин, пароль, приоритет, флаг «Кэшировать названия объектов».

Если нужно сразу скопировать или перенести уже существующие в MDM объекты данного класса в создаваемое хранилище, пользователь отмечает переключатели «Копировать объекты подклассов» и/или «Удалить объекты из исходного хранилища».

3. Отображается диалог «Выберите столбцы для индексирования», в котором выводится список всех атрибутов, применимых к объектам данного класса. Пользователь может отметить галочками те атрибуты, по которым нужно создать индексы.

4. Создается таблица в хранилище и индексы по тем полям, которые отметил пользователь, а также по всем полям, которые в таблице присутствуют по умолчанию.

7. Создаются необходимые настройки MDM, сбрасывается кэш MDM.

8. Выполняется копирование или перенос объектов в новое хранилище, если была установлена такая опция.

Настройка маршрутов согласования

В разделе «Маршруты согласования» производится настройка последовательностей подтверждения изменений, которые пользователи вносят в атрибуты объектов модели.

Маршрут согласования создается для точки доступа.

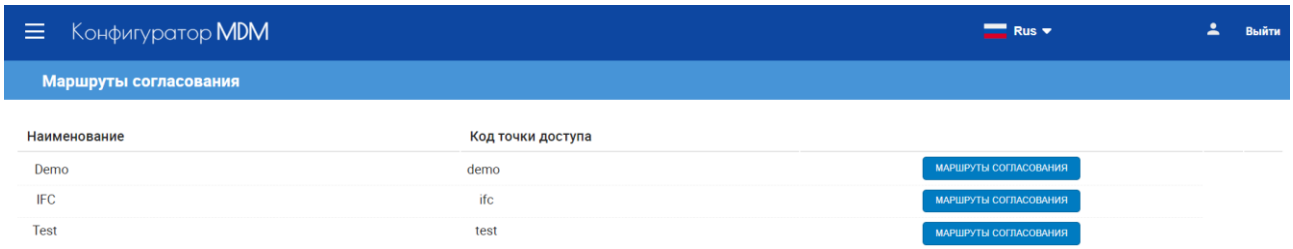

Рис. 9. Список точек доступа с маршрутами согласования

В диалоге создания/редактирования маршрута согласования нужно указать следующие свойства:

- Наименование – читаемое название маршрута

- Класс(ы) объектов – классы объектов, для которых определен маршрут согласования
- Применять при создании/изменении/удалении – признаки использования маршрута для соответствующих действий с объектами класса
- Активен – позволяет временно отключить маршрут для объектов данного класса.

Для созданного маршрута необходимо выбрать участников, которые должны согласовывать изменения объектов указанного класса через АрхиГраф.Мир. Для перехода к редактированию структуры маршрута нужно нажать кнопку «Маршрут».

Участники могут объединяться в группы, если от них требуется проводить согласование в параллельном режиме – для этого нужно нажать кнопку «Добавить группу». В качестве участников маршрута нажатием на кнопку «Добавить сущность» могут быть добавлены:

Информационная система – клиент АрхиГраф.MDM. В этом случае согласование сможет выполнять любой пользователь, работающий с MDM от имени соответствующей системы.

Логин – пользователь, работающий с АрхиГраф.Мир и авторизованный с указанным логином KeyCloak.

Имя пользователя – пользователь, работающий с АрхиГраф.Мир и авторизованный в KeyCloak, причем его учетная запись имеет указанное имя.

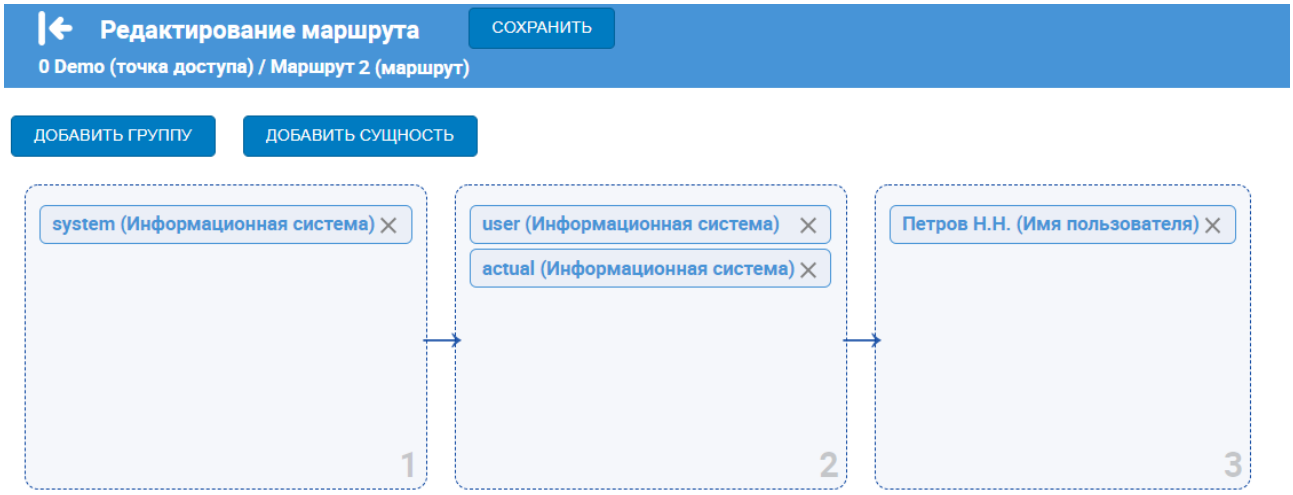

Рис.10. Редактирование маршрута согласования

После создания маршрута его участников можно перемещать между группами путем перетаскивания, удалять участников нажатием на крестик в

блоке с именем участника. По окончании редактирования маршрута его необходимо сохранить.

Настройка систем-клиентов

В разделе «Системы-клиенты» конфигуратора MDM выполняется настройка учетных записей, под которыми автоматизированные системы подключаются к API MDM. На стартовой странице этого раздела отображается список точек доступа. После выбора точки доступа отображается список зарегистрированных для нее систем-клиентов, который имеет такой вид:

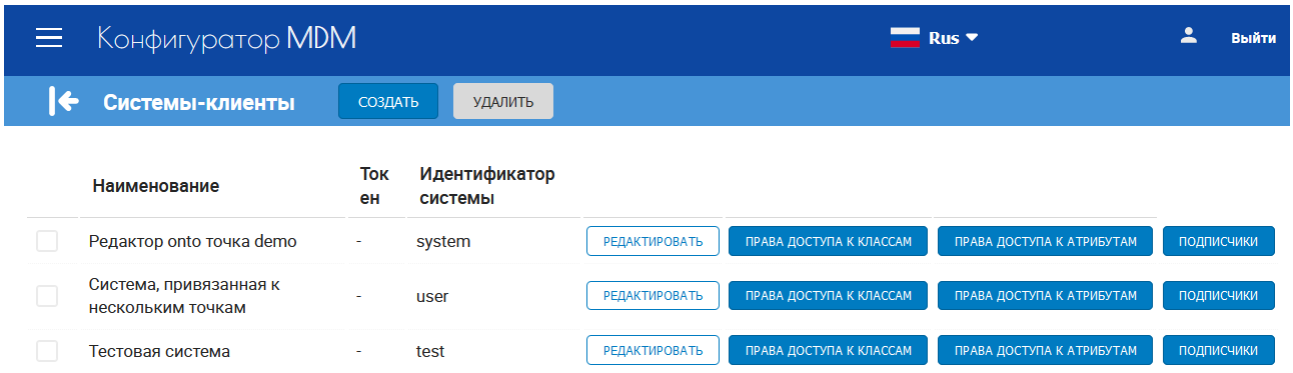

Рис. 11. Список систем-клиентов для точки доступа

В диалоге создания/редактирования системы нужно указать следующие свойства:

- Наименование – читаемое название
- Идентификатор системы – используется при обращении через API, указывается в поле Originator всех запросов
- Токен – опционально используется для дополнительного подтверждения аутентичности обращающейся системы, может передаваться в поле Token в обращениях к API.

По умолчанию каждая система-клиент имеет неограниченный доступ ко всем данным в точке доступа. Эти права можно ограничить. При нажатии на кнопку «**Права доступа к классам**» отображается список классов, к которым системе-клиенту разрешен доступ. Нажатие на кнопку «Создать» открывает диалоговое окно, в котором необходимо выбрать класс и установить переключатели:

- Чтение – система-клиент может читать объекты данного класса
- Запись – система-клиент может создавать, изменять, удалять объекты данного класса

- Подтверждение – система-клиент может создавать запросы на создание, изменение, удаление объектов данного класса, которые должен подтвердить или отклонить пользователь, работающий от имени системы с более высокими правами доступа.

Если все три переключателя установить в положение «Нет», данной системе будет запрещен доступ ко всем объектам выбранного класса.

Подклассы наследуют уровень доступа, установленный для надкласса.

На странице «**Права доступа атрибутов**» аналогичным образом настраиваются права доступа системы-клиента к значениям свойств объектов онтологии:

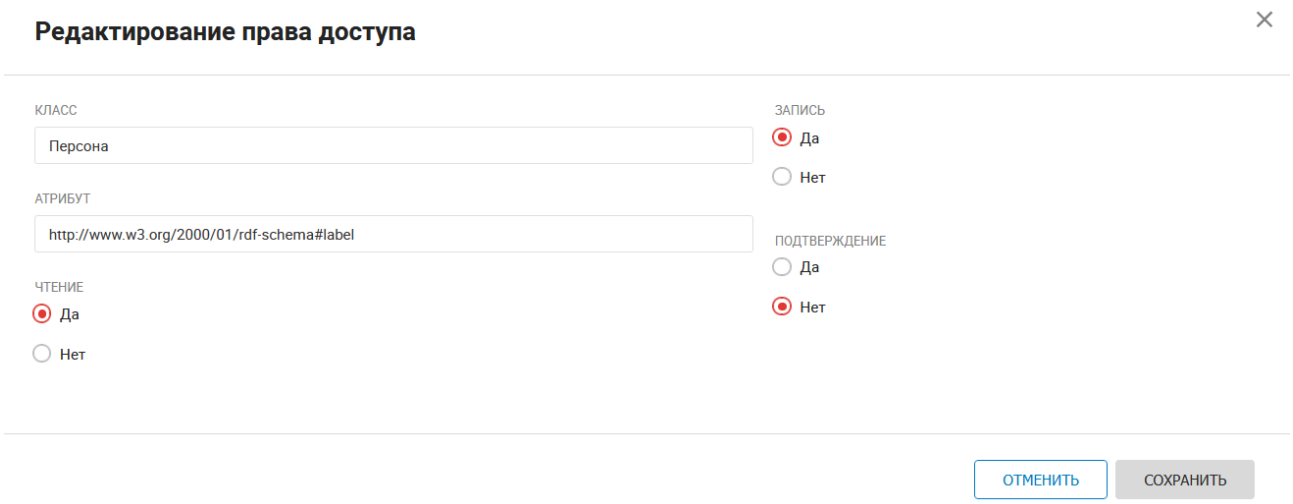

Рис. 12. Диалог настройки прав доступа к свойствам объекта онтологии

Возможно, например, настроить права доступа какой-либо системы к объектам класса «Персона» как «Запись», но для атрибута rdfs:label установить доступ «Чтение». Тогда система-клиент сможет изменять значения всех атрибутов Персоны, кроме rdfs:label, который будет доступен ей только на чтение.

Права доступа к атрибутам настраиваются в контексте класса; таким образом, системе-клиенту может быть доступен только для чтения атрибут rdfs:label объектов класса Персона, но доступен на запись атрибут rdfs:label объектов класса Контрагент.

На третьей вкладке, «**Подписчики**», можно просмотреть и настроить набор подписок для данной системы-клиента (установить подписки может и само приложение с помощью API MDM). На стартовом экране этой вкладки отображаются реквизиты очереди, через которую система-клиент хочет

получать от MDM по подписке информацию о создании/изменении/удалении объектов определенных классов.

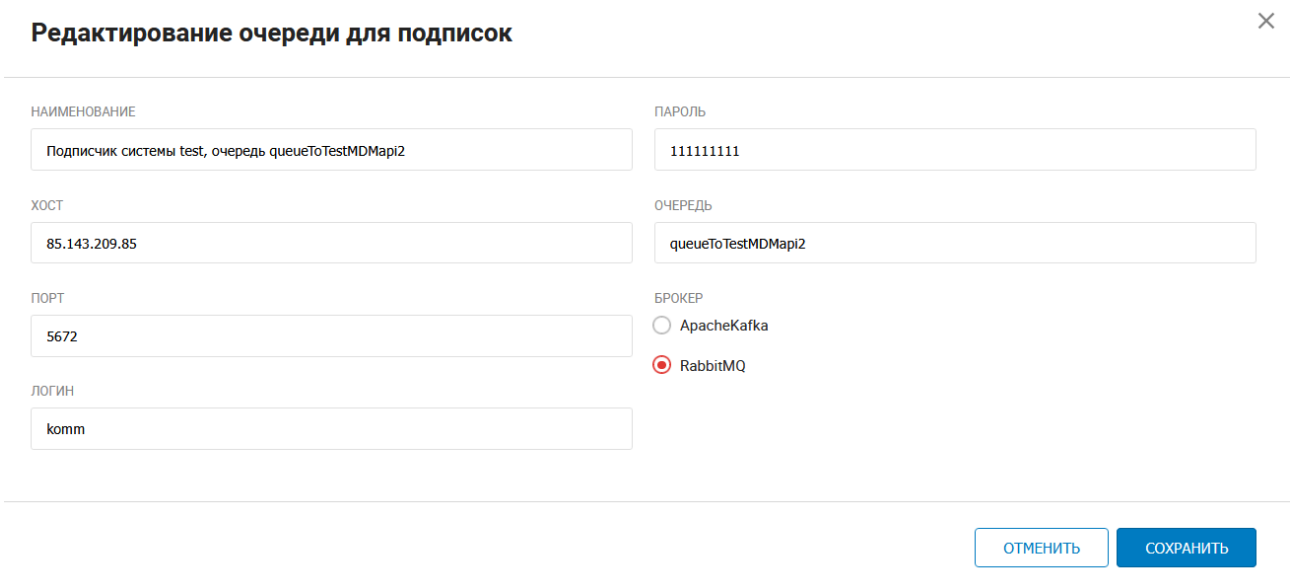

Рис. 13. Реквизиты подписки

Здесь нужно ввести читаемое название конфигурации, выбрать один из поддерживаемых типов очередей обмена сообщениями (RabbitMQ или Kafka), а также указать реквизиты подключения к очереди: хост, порт, логин, пароль, имя очереди/топика.

Затем нужно нажать кнопку «Подписки». Будет отображен список классов, информация об изменении объектов которых поступает системе-клиенту через эту очередь. Окно добавления/редактирования подписки имеет такой вид:

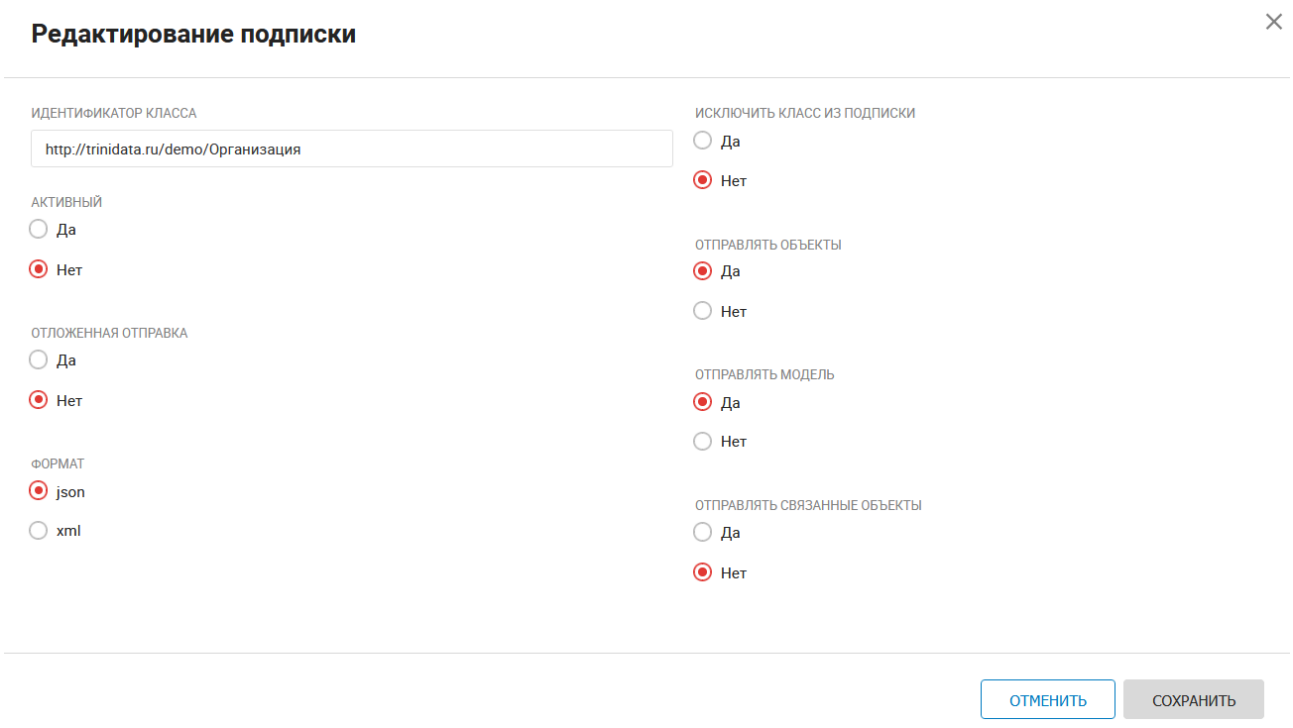

Рис. 14. Окно свойств подписки на определенный класс

В этом окне необходимо выбрать идентификатор класса онтологии, а затем установить переключатели:

- Активный – позволяет временно отключить подписку на объекты данного класса
- Отложенная отправка – если установлен, то отправка сообщения происходит не в момент изменения объекта в MDM, а пакетами с определенной периодичностью
- Формат отправляемых сообщений – XML или JSON
- Исключить класс из подписки – возможен вариант, когда устанавливается подписка на надкласс, а затем исключается подписка на один или несколько его подклассов
- Отправлять объекты – если установлен, то MDM отправляет уведомления об изменении объектов выбранного класса
- Отправлять модель – если установлен, то MDM отправляет уведомления об изменении модели (набора применимых атрибутов, свойств самого класса) выбранного класса
- Отправлять связанные объекты – если установлен, то MDM отправляет не только сам изменившийся объект, но и связанные с ним объекты.

На вкладке «Условия и группы» можно настроить логические условия, по которым происходит отбор объектов, отправляемых по подписке (если условия не настроены, то отправляются любые измененные объекты класса). Пояснения по настройке условий даны на самой странице конфигуратора.

На вкладке «Связанные объекты» можно указать, объекты каких именно классов должны отправляться вместе с измененным объектом, если установлен переключатель «Отправлять связанные объекты». Также нужно выбрать свойство, которым измененный объект ссылается на связанный объект, который нужно отправить.

Настройка прослушиваемых очередей

АрхиГраф.MDM может получать запросы от систем-клиентов через очереди в асинхронном режиме, и отправлять ответы на эти запросы также через очереди. В разделе «Прослушиваемые очереди» конфигуратора можно настроить пары таких очередей. Интерфейс раздела имеет такой вид:

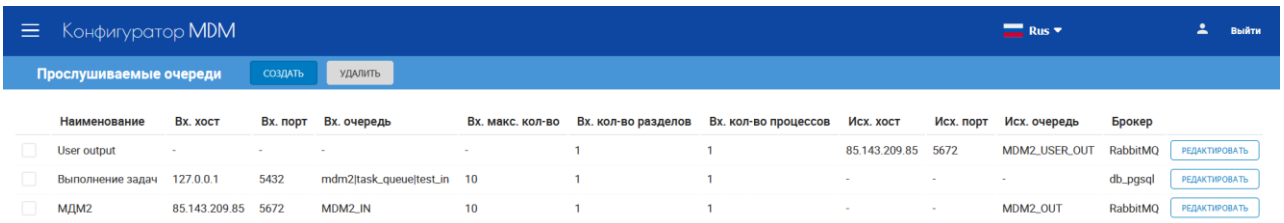

Рис. 15. Настройка прослушиваемых очередей

Для каждой пары очередей указываются следующие настройки:

Наименование – читаемое название для отображения в интерфейсе

Брокер – тип менеджера очередей, RabbitMQ или Kafka

Вх./Исх. хост, порт, логин, пароль, очередь – реквизиты для подключения MDM к входящей и исходящей очереди

Вх. макс. кол-во, Вх./Исх. кол-во разделов, кол-во процессов – настройки очередей, определяющие режим чтения и отправки сообщений, в том числе количество процессов MDM для обработки сообщений определенной очереди.

Настройка констант

Раздел «Константы» конфигуратора MDM позволяет настроить значения переменных, определяющих общие настройки работы системы. Раздел имеет следующий вид:

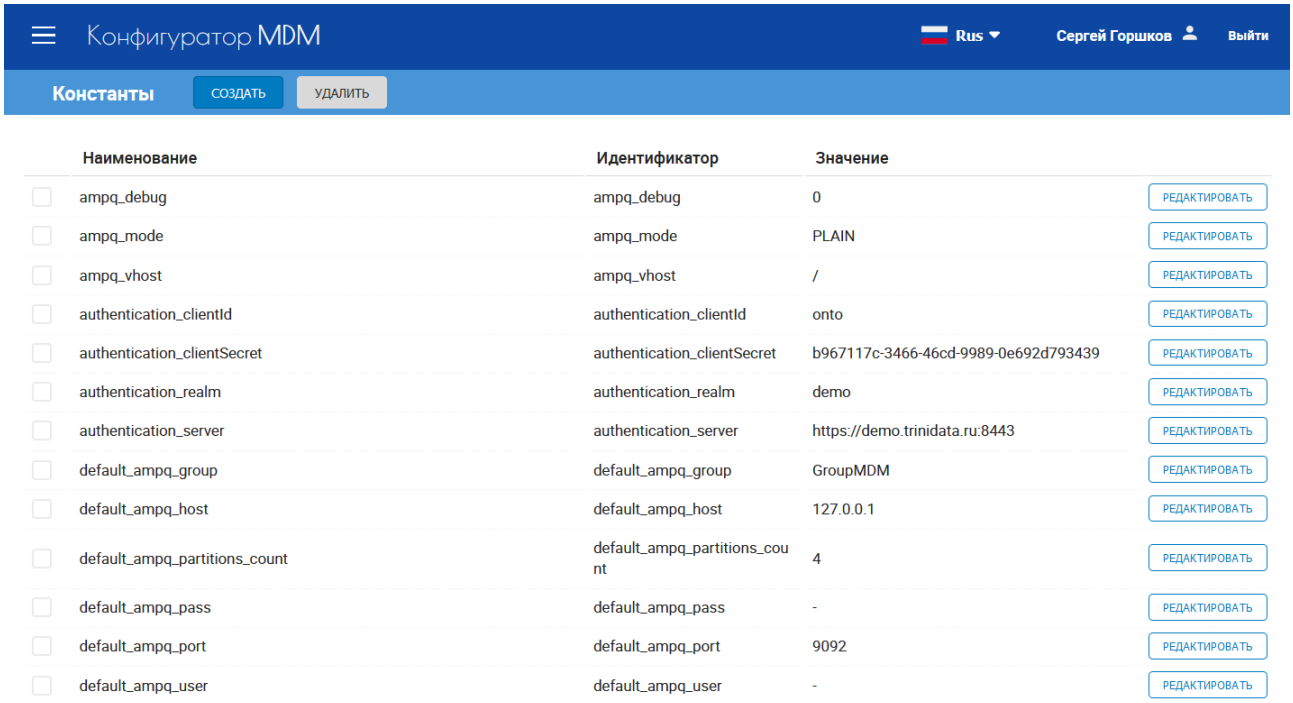

Рис. 16. Раздел «Константы» конфигуратора MDM

Настройка АрхиГраф.MDM с помощью утилиты mdmctl

Утилита служит инструментом настройки параметров МДМ, таких как точка доступа, хранилище, система, классы, языки и т. д.

Во всех командах есть необязательный параметр --debug yes, который выводит sql запрос. На вспомогательные запросы это не распространяется.

Вывод помощи:

```
mdmctl help
```

Выводит краткую справку по доступным командам.

Точка доступа (endpoint)

Создание точки доступа

Команда:

```
mdmctl add endpoint code_endpoint
```

где:

«code_endpoint» - код создаваемой точки доступа. Обязательный параметр создает точку доступа с указанным кодом, пустым именем и английским языком по умолчанию.

Необязательные параметры:

--name - задает имя точки доступа для удобства идентификации. Так же с именем через знак "@" идет код языка, если язык не указывается, по умолчанию будет установлен английский.

--prefix — задает префикс для точки доступа

--root — задает корень точки доступа

--storage — задает хранилище данных. В этом параметре передается имя хранилища без указания языка.

--model_storage - задает хранилище модели. В этом параметре передается имя хранилища без указания языка.

Пример набора параметров для создания точки доступа:

```
mdmctl add endpoint Demo --name "Точка Demo"@ru --prefix
http://trinidata.ru/demo/ --root http://trinidata.ru/demo
```

Удаление точки доступа

Команда:

```
mdmctl delete endpoint code_endpoint
```

где:

code_endpoint - код удаляемой точки доступа.

Точка доступа не удаляется безвозвратно, а помечается специальным атрибутом и считается удаленной для системы.

Обновление точки доступа

Команда:

```
mdmctl update endpoint code_endpoint
```

где:

code_endpoint - код обновляемой точки доступа.

Без дополнительных параметров ничего не произойдет. Для изменения определенных параметров необходимо задействовать хотя бы один дополнительный параметр со значением.

Дополнительные параметры:

--name — задает имя для точки доступа. Если точек доступа с таким именем несколько (один код и одно имя, но разные языки) обновятся все точки доступа, сменится только имя. Если задаем имя и язык (например: mdmctl update endpoint demo-point@en --name demo) обновится имя только для точки доступа с кодом языка «en».

--prefix — меняет префикс для точки доступа

--root — меняет корень для точки доступа

--model_storage — меняет хранилище моделей для точки доступа

--storage — меняет хранилище данных для точки доступа

--code — меняет код точки доступа. Если код задан с языком (например mdmctl update endpoint demo@en --code Demo1) заменит код только только для точки доступа, с сочетанием код @язык.

Обновление префикса по умолчанию, альтернативный вариант

Команда:

```
mdmctl update prefix_default code_endpoint new_prefix
```

где:

code_endpoint - код обновляемой точки доступа,

new_prefix — новый префикс по умолчанию для выбранной точки доступа.

Просмотр списка точек доступа

Команда**:**

`mdmctl show endpoint` – выведет свойства всех точек доступа

`mdmctl show endpoint Demo` - покажет свойство точки доступа с кодом «Demo»

Просмотр префикса по умолчанию для точки доступа

Команда:

```
mdmctl show prefix_default Demo
```

покажет свойство точки доступа с кодом «Demo».

Хранилище (storage)

Создание хранилища

Команда:

`mdmctl add storage storage_type storage_name`

где:

storage_type – тип хранилища (варианты: fuseki, allegrograph, blazegraph, mongodb, mongo, postgresql, postgresql_jsonb)

"storage_name" – имя хранилища.

Необязательные параметры:

--host — имя сервера

--port — порт на сервере

--login — логин для доступа к базе

--password — пароль для доступа к базе (нельзя использовать восклицательный знак и кавычки (и одинарные, и двойные))

--query — путь к SPARQL-сервису чтения данных

--update — путь к SPARQL-сервису изменения данных

--dbname — имя базы данных, которая используется как хранилище

--table — указывает конкретную таблицу с данными

--history — если true, то определяет, что хранилище сохраняет историю изменения свойств объектов

--logic — если true, то определяет, что хранилище поддерживает логические операции

Примеры команд с параметрами:

Для Fuseki:

```
mdmctl add storage fuseki "Хранилище модели Demo" --host 127.0.0.1 --port 8080 --login mdm --password test --query /fuseki/demo/query --update /fuseki/demo/update
```

В этой команде fuseki – тип создаваемого хранилища, "Хранилище модели Demo" – его название, 127.0.0.1 – IP-адрес хоста, 8080 – порт, mdm – логин для HTTP-авторизации, test – пароль для HTTP-авторизации, /fuseki/demo/query – путь к SPARQL-сервису чтения данных, /fuseki/demo/update – путь к SPARQL-сервису изменения данных.

Для Postgres:

```
mdmctl add storage postgresql_jsonb "Хранилище данных Demo Postgres" --host 127.0.0.1 --login mdm --password test --dbname mdm_data --table test --history true --logic true
```

--dbname mdm_data – таким образом задается имя базы данных, которая используется как хранилище , --table test – указывает конкретную таблицу с данными, --history true – определяет, что хранилище сохраняет историю

изменения свойств объектов, --logic true – определяет, что хранилище поддерживает логические операции.

Удаление хранилища

Команда:

```
mdmctl delete storage "Хранилище данных Demo Postgres"
```

где:

"Хранилище данных Demo Postgres" — пример имени удаляемого хранилища.

Присоединение хранилища данных к точке доступа

Команда:

```
mdmctl bind storage "Хранилище данных Demo Postgres" Demo
```

где:

"Хранилище данных Demo Postgres" — пример имени хранилища

Demo — пример кода точки доступа.

Присоединение хранилища моделей к точке доступа

Команда:

```
mdmctl bind model_storage "Хранилище модели Demo" Demo
```

где:

"Хранилище модели Demo" — пример имени хранилища

Demo — пример кода точки доступа.

Отвязывание хранилища от точки доступа

Команда:

```
mdmctl unbind storage "Хранилище данных Demo Postgres" Demo
```

где:

"Хранилище модели Demo" — пример имени хранилища

Demo — пример кода точки доступа.

Обновление хранилища

Команда:

```
mdmctl update storage "storage_name"
```

где:

"storage_name" — имя хранилища.

Дополнительные параметры:

--name — новое имя хранилища (без @ и кода языка)

--host — имя сервера

--port — порт на сервере

--login — логин для доступа к базе

--password — пароль для доступа к базе

--query_path — путь к SPARQL-сервису чтения данных

--update_path — путь к SPARQL-сервису изменения данных

--dbname — имя базы данных, которая используется как хранилище

--tab — указывает конкретную таблицу с данными

--history — если true, то определяет, что хранилище сохраняет историю изменения свойств объектов

--logic — если true, то определяет, что хранилище поддерживает логические операции

Для обновления необходимо добавить хотя бы один дополнительный параметр со значением

Пример команды:

```
mdmctl update storage "Хранилище модели Demo"  --name "Хранилище данных
demo" --dbname stor_demo
```

изменит хранилище с именем "Хранилище модели Demo". Изменится имя на "Хранилище данных demo" и имя базы данных станет "stor_demo".

Удаление хранилища

Команда:

```
mdmctl delete storage "Хранилище данных Demo Postgres"
```

где:

"Хранилище данных Demo Postgres" — пример имени хранилища.

Просмотр списка хранилищ

Команда:

```
mdmctl show storage
```

покажет все существующие хранилища

```
mdmctl show storage "Хранилище данных Demo Postgres"
```

покажет свойства хранилища с именем "Хранилище данных Demo Postgres"

Языки

Создание языка

Команда:

```
mdmctl add language it "Italiano"
```

где:

it – пример кода языка (итальянский)

"Italiano" – пример наименования языка.

Просмотр списка языков

```
mdmctl show language
```

выводит информацию по всем языкам

```
mdmctl show language ru
```

выводит информацию по языку с кодом ru.

Удаление языка

```
mdmctl delete language it
```

где:

it – пример кода языка.

Прослушиваемые очереди

Создание прослушиваемой очереди в настройках MDM

Команда:

```
mdmctl add queue
```

Дополнительные параметры:

--name — имя прослушиваемой очереди

--in — название входящей очереди

--out — название исходящей очереди

--input_host — хост входящей очереди

--output_host — хост исходящей очереди

--input_port — порт входящей очереди

--output_port — порт исходящей очереди

--input_login — логин входящей очереди

--output_login — логин исходящей очереди

--input_password — пароль входящей очереди

--output_password — пароль исходящей очереди

--broker — брокер очередей (например, RabbitMQ)

Для обновления необходимо добавить хотя бы один дополнительный параметр со значением

Пример команды

```
mdmctl add queue --name demo --in SYNCH_QUEUE_IN --out SYNCH_QUEUE_OUT --
input_host 127.0.0.1 --input_port 5672 --broker RabbitMQ --input_login
admin --input_password mypass --output_host 127.0.0.1 --output_port 5672 -
-output_login admin --output_password mypass
```

Если не задать имя оно формируется из параметров --in и --out, например, из команды выше получилось бы "SYNCH_QUEUE_IN to SYNCH_QUEUE_OUT"

Просмотр прослушиваемых очередей

Команда:

```
mdmctl show queue
```

выводит свойства всех очередей.

```
mdmctl show queue demo
```

выводит свойство очереди с именем demo

Обновление очереди

Команда:

```
mdmctl update queue "queue_name"
```

где:

"**queue_name**" имя очереди.

Дополнительные параметры:

--name — имя прослушиваемой очереди

--input_queue — название входящей очереди

--out_queue — название исходящей очереди

--input_host — хост входящей очереди

--output_host — хост исходящей очереди

--input_port — порт входящей очереди

--output_port — порт исходящей очереди

--input_login — логин входящей очереди

--output_login — логин исходящей очереди

--input_password — пароль входящей очереди

--output_password — пароль исходящей очереди

--broker — брокер очередей (например, RabbitMQ)

--input_max_count — максимальное количество для входящих сообщений в очереди

Пример команды:

```
mdmctl update queue test --name "Синхронизация контуров" --broker RabbitMQ
```

Изменит очередь с именем "test". Изменится имя на «Синхронизация контуров» и брокер очередей заменится на RabbitMQ.

Удаление прослушиваемой очереди

Команда (пример):

```
mdmctl delete queue --in SYNCH_QUEUE_IN --out SYNCH_QUEUE_OUT --input_host
127.0.0.1
```

Дополнительные параметры:

--in — имя входящей очереди

--out — имя исходящей очереди

--input_host — хост исходящей очереди. Этот параметр желательно указывать, чтобы не удалить случайно лишнюю очередь.

Константы

Установка значения константы

Команда:

```
mdmctl set constant log_folder "/../log/"
```

первый параметр – идентификатор константы,

второй – ее значение.

Дополнительный параметр:

--name — задает имя для новой константы

При выполнении команды создается или изменяется константа. Если команда указана без ключа

--name обновляется существующая константа.

Просмотр значений констант:

Команда:

```
mdmctl show constant
```

выводит значения всех констант.

```
mdmctl show constant log_folder
```

выводит значение конкретной константы log_folder.

Системы-клиенты MDM

Создание системы (Originator’a)

Команда:

```
mdmctl add system "Имя_системы" system_code
```

где:

"Имя_системы" — отображаемое имя системы

system_code — код системы

Дополнительные параметры:

--main — информационный флаг, если 1 то система ассоциирована с редактором онтологий. По умолчанию 0.

--final - по умолчанию 0, если задать 1, то система, получив пакет по подписке дальше рассылать не будет, даже если есть подписчики на объект такого класса. Системы, заведенные для синхронизации, имеют final=1.

Пример команды:

```
mdmctl add system "Демо-клиент" demo --token democlient --main 1 --final 0
```

Будет создана система с именем "Демо-клиент", кодом demo токеном democlient, будет ассоциирована с редактором онтологий и признаком того, что от этой системы подписчики будут получать пакеты на обновление.

Предоставление доступа Originator'у к точке доступа:

Команда:

```
mdmctl bind system code_system code_endpoint
```

где:

code_system — код системы

code_endpoint — код точки доступа.

Пример команды:

```
mdmctl bind system test demo
```

свяжет систему test с точкой доступа demo

Установка прав Originator'а на доступ к объектам класса

Команда:

```
mdmctl allow endpoint_code system_code entity_code read write confirm
```

где:

endpoint_code — код точки доступа

system_code — код системы

entity_code — код класса

read — 1 если разрешено, 0, если запрещено

write — 1 если разрешено, 0, если запрещено

confirm — 1 если разрешено, 0, если запрещено

Параметры вводятся через пробел.

Пример команды:

```
mdmctl allow Demo demo "Событие" 1 1 0
```

Будут установлены права системы (Originator'а) на доступ к объектам класса «Событие» чтение, запись, без подтверждения.

Установка прав Originator'а на доступ к атрибутам объектов класса

Команда:

```
mdmctl allow attr endpoint_code system_code class entity_code read write confirm
```

где:

endpoint_code — код точки доступа

system_code — код системы

class — код класса

entity_code — код атрибута

read — 1 если разрешено, 0, если запрещено

write — 1 если разрешено, 0, если запрещено

confirm — 1 если разрешено, 0, если запрещено

Параметры вводятся через пробел.

Пример команды:

```
mdmctl allow attr Demo demo "Событие" "связаноСКартой" 1 1 0
```

Будут установлены права системы (Originator’а) на доступ к атрибуту "связаноСКартой" объектов класса «Событие» чтение, запись, без подтверждения.

Просмотр списка Originator’ов

Команда:

```
mdmctl show system
```

просмотр всех свойств

```
mdmctl show system demo
```

просмотр свойств только для системы demo.

Просмотр прав Originator’а на классы:

Команда:

```
mdmctl show rights
```

просмотр всех заданных прав

```
mdmctl show rights demo
```

просмотр заданных прав для точки доступа demo

Просмотр прав Originator’а на атрибуты классов:

Команда:

```
mdmctl show rights attr
```

просмотр всех заданных прав на атрибуты класса

```
mdmctl show rights attr demo
```

просмотр заданных прав на атрибуты класса для точки доступа demo

Обновление системы

Команда:

```
mdmctl update system system_code
```

где:

system_code — код обновляемой системы

Дополнительные параметры:

--name — новое имя системы

--token — токен системы

--code — новый код системы

--endpoint — код точки доступа

--main — 1, если система ассоциирована с редактором онтологий, 0, если нет

--final — 1, если система конечная(принимает пакеты обновлений от другой системы), 0, если нет

Для обновления необходимо добавить хотя бы один дополнительный параметр со значением

Пример команды:

```
mdmctl update system test --name "Тестовая система" --endpoint demo
```

Задаст для системы новое имя "Тестовая система" и свяжет ее с точкой доступа "demo"

Удаление системы (Originator'a)

Команда:

```
mdmctl delete system demo
```

Управление распределением данных между хранилищами

Привязка классов к хранилищам

Команда:

```
mdmctl bind class class_code storage_name endpoint_code
```

где:

"class_code" - код класса. Можно использовать краткую форму записи, если он имеет префикс по умолчанию для данной точки доступа, и полную форму (в виде URI) в остальных случаях (пример: краткая форма: rdfs:label, полная форма: http://www.w3.org/2000/01/rdf-schema#label).

"storage_name" - имя системы

endpoint_code - код точки доступа

Пример команды:

```
mdmctl bind class "Событие" "Хранилище данных Demo Postgres" Demo
```

Просмотр списка классов, привязанных к хранилищам для точки доступа

Команда:

```
mdmctl show class
```

покажет все классы

```
mdmctl show class "endpoint_code"
```

покажет только классы для точки доступа "endpoint_code"

Пример команды:

```
mdmctl show class demo
```

покажет классы для точки доступа "demo"

Отвязывание класса от хранилища

Команда:

```
mdmctl unbind class class_code "storage_name" endpoint_code
```

где:

"class_code" - код класса.

"storage_name" - имя системы

"endpoint_code" - код точки доступа

Пример команды:

```
mdmctl unbind class "Событие" "Хранилище данных Demo Postgres" Demo
```

Отвяжет класс "Событие" от хранилища "Хранилище данных Demo Postgres" и точки доступа "Demo".

Управление хранением значений свойств в хранилищах

Создание мэппинга свойства

Команда:

```
mdmctl bind property attribute_name storage_name table_name field_name
```

где:

"attribute_name" - идентификатор свойства в онтологической модели

"storage_name" - название хранилища

"table_name" - название таблицы базы данных

"field_name" - имя соответствующего поля в этой таблице

Пример команды:

```
mdmctl bind property "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" "Хранилище данных Demo Postgres" test type
```

Создаст мэппинг атрибута "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" в хранилище с именем "Хранилище данных Demo Postgres"

Для привязки столбца с идентификатором сущности используется следующая команда:

```
mdmctl bind property uri "storage_name" table_name uri
```

Просмотр мэппинга свойств

Команда:

```
mdmctl show property
```

просмотр всех свойств

```
mdmctl show property "table_name"
```

просмотр свойств только для таблицы "table_name"

Удаление свойства из мэппинга

Команда:

```
mdmctl unbind property "attribute_name" "storage_name" table_name
```

где:

"attribute_name" - идентификатор свойства в онтологической модели

"storage_name" - название хранилища. Если хранилища для записи нет, ставим ""

"table_name" - название таблицы базы данных

Пример команды:

```
mdmctl unbind property "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" "Хранилище данных Demo Postgres" test
```

Удалит мэппинг для атрибута "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" в хранилище с именем Хранилище данных Demo Postgres"

Управление свойствами, определенными вне онтологии (стандартными свойствами)

Создание свойства в точке доступа

Команда:

```
mdmctl add property "property_uri" endpoint_code
```

где:

"property_uri" - URI создаваемого свойства (например: owl:sameAs)

"endpoint_code" - код точки доступа

Необязательные параметры:

--name — имя объекта с кодом языка через знак "@". Одновременно можно задавать несколько имен на разных языках.

--domain - область применения свойства

--range - диапазон значений свойства

--singular - равно true, если свойство может принимать единственное значение для каждого объекта

--mandatory - равно true, если свойство должно обязательно иметь хотя бы одно значение для каждого объекта

--multilang - равно true, если значения свойства могут быть заданы на разных языках

--readonly - равно true, если значение свойства доступно только для просмотра и не может быть изменено.

Пример команды:

```
mdmctl add property owl:sameAs Demo --name "Эквивалентен"@ru --name "Equivalent"@en --domain owl:NamedIndividual --range owl:NamedIndividual --singular true --mandatory false --multilang true --readonly false
```

Создаст свойство owl:sameAs в точке доступа Demo с именами Эквивалентен и Equivalent на русском и английском языке с областью применения owl:NamedIndividual, диапазоном значений owl:NamedIndividual, свойство будет принимать единственное значение для каждого объекта, не обязательно для каждого объекта, свойство будет мультиязычное и может быть изменено.

Просмотр списка свойств для точки доступа

Команда:

```
mdmctl show field
```

выводит все свойства для всех точек доступа

```
mdmctl show field endpoint_code
```

выводит свойства только для точки доступа endpoint_code

Удаление свойства из точки доступа

Команда:

```
mdmctl delete property "property_uri" endpoint_code
```

где:

"property_uri" - URI свойства (например: owl:sameAs)

"endpoint_code" - код точки доступа

Требования к квалификации персонала, осуществляющего сопровождение АрхиГраф.MDM

Операции по администрированию АрхиГраф.MDM должны выполняться системным администратором, имеющим опыт в администрировании Linux-систем, знакомым с принципами работы языков Java, PHP. Для настройки обмена между АрхиГраф.MDM с приложениями-клиентами (схема организации обмена описана в Руководстве пользователя) необходимо знание принципов и

технологий брокера сообщений (Message Broker), корпоративной сервисной шины (Enterprise Service Bus). Для поддержки модели данных АрхиГраф.MDM необходима квалификация аналитика/инженера по данным, а также знакомство с семантическими технологиями. Все необходимые сведения о них можно получить в нашем методическом пособии «Введение в онтологическое моделирование», http://trinidata.ru/files/SemanticIntro.pdf.

Структура лог-файлов

Каждый файл содержит лог за сутки. Названия файлы логов имеют вид:

- mdm_log_YYYYmmdd.log – логи входящих запросов.
- mdm_errorlog_YYYYmmdd.log – лог произошедших ошибок.

Столбцы файла лога запросов:

- дата и время запроса,
- идентификатор запросившей системы (если в запросе задан код системы и система с таким кодом есть),
- код запросившей системы (если найден в запросе),
- корневой тег запроса (например, GetObjectsGroup или DataModelRequest),
- ключевой атрибут запроса (Code в GetObjectsGroup и GetObject).

Столбцы файла с логом ошибок:

- дата и время запроса,
- текст ошибки,
- идентификатор запросившей системы (если задан код и система найдена),
- код запросившей системы (если задан в запросе),
- корневой тег запроса.
- ключевой атрибут запроса.

Те же логи могут сохраняться в базе PostgreSQL в таблице mdm_log. Оба лога (ошибок и осуществленных запросов) пишутся в одну таблицу, различаются они значением поля type: 1 для ошибок и 2 для запросов. В отличие от текстового лога, в базу пишется так же полный текст запроса. Структура таблицы mdm_log:

- date - дата и время запроса,
- type — тип лога, 1 (для ошибки) или 2,
- comment — текст ошибки

- xml — поступивший запрос
- system — идентификатор запросившей системы (если система задана и найдена)
- system_code — код запросившей системы (если задан)
- request - корневой тег запроса
- code - ключевой атрибут запроса (Code в GetObjectsGroup и GetObject)

По вопросам, связанным с функционированием АрхиГраф.MDM, для решения возникающих проблем, можно обращаться в службу поддержки компании ТриниДата по телефону: +7 343 2 110 256.