ОПИСАНИЕ API

платформы АрхиГраф

Содержание

1. Назначение платформы АрхиГраф

Платформа АрхиГраф предназначена для хранения информационной модели, нормативно-справочной информации, основных и транзакционных данных организации. Она обеспечивает доступ прикладным программным компонентам программного как к данным, размещенных в хранилищах под ее управлением, так и к любой информации во внешних хранилищах.

Платформа АрхиГраф предназначена для использования в качестве ядра сложных прикладных автоматизированных систем, в том числе:

- ситуационных центров,
- интегрированных аналитических систем и логических витрин данных,
- систем построения отчетности и др.

Также АрхиГраф может выступать в роли корпоративного инструмента управления данными (Data Governance) в сложных, интегрированных инфраструктурах, включающих десятки бизнес-приложений.

АрхиГраф обеспечивает синхронизацию данных между различными приложениями, контроль прав доступа при обращении к информации, версионность модели и данных, извлечение данных из внешних источников в режиме логической витрины, подписку на уведомления об изменении модели и данных, логический контроль информации и многие другие функции.

Структура (модель) информации, с которой работает АрхиГраф, представляется в виде онтологической модели. Это позволяет приложениям-клиентам работать с машинно-читаемым представлением не только самих данных, но и их структуры – информационной модели. Онтологии допускают применение множественной (фасетной) классификации сущностей, наследование наборов атрибутов, множественность значений атрибутов и др.

2. Архитектура обмена

2.1. Форматы запросов

Принцип работы программного интерфейса АрхиГраф основан на обмене пакетами с другими программными системами. Поддерживаются запросы в двух основных форматах:

- JSON
- XML

Также возможно чтение данных платформы при помощи языков запросов SPARQL и GraphQL.

2.2. Транспорт

Транспорт XML или JSON пакетов может осуществляться несколькими способами:

- при помощи очередей RabbitMQ или Kafka (предпочтительный способ, и единственный возможный в высоконагруженных и высоконадежных архитектурах),
- REST-сервисов,
- протокола Websocket,
- web-интерфейса (в отладочных целях).

Форматы пакетов и последовательность обмена ими не зависят от способа передачи информации. В процессе обмена каждое приложение, желающее отправить запрос к АрхиГраф, должно сформировать сообщение в любом из поддерживаемых форматов, и отправить его в систему при помощи соответствующего транспорта. Платформа возвратит ответ при помощи того же транспортного протокола. Исходящий пакет будет возвращен в том же формате, в каком получен входящий.

3. Средства управления платформой АрхиГраф

АрхиГраф имеет собственную панель администрирования, имеющую следующий вид:

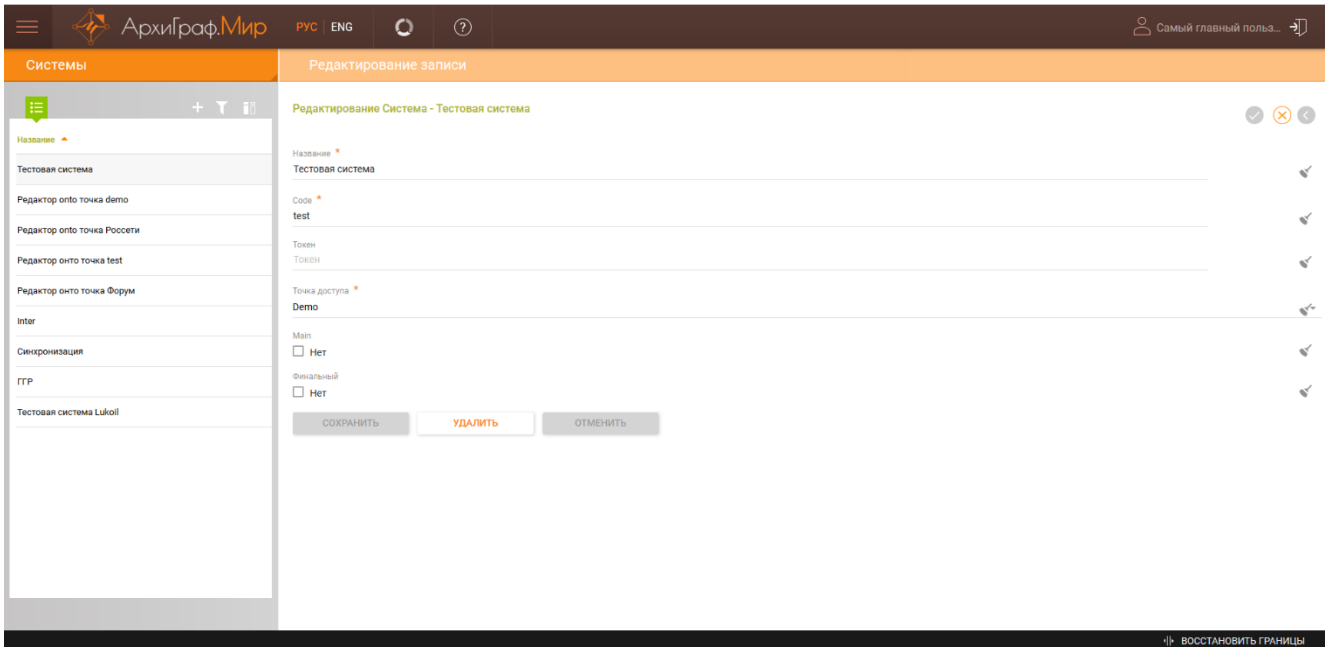

Рис. 1. Административная панель платформы АрхиГраф

Административная панель обеспечивает управление следующими настройками платформы:

- Хранилища данных
- Точки доступа (наборы данных)
- Информационные системы-клиенты
- Очереди обмена данными
- Подписки
- Константы
- Служебные классы и атрибуты, поддерживаемые платформой
- Подтверждение или отклонение запросов на редактирование

Управление структурой модели может выполняться аналитиком через визуальный онлайн-редактор онтологий АрхиГраф.Мир. Некоторые инструменты работы с данными платформы встраиваются также в систему АрхиГраф.СУЗ, и обеспечивают выполнение следующих функций:

- Интерфейс редактирования информационной модели позволяет создавать, редактировать, удалять классы, атрибуты и экземпляры объектов, просматривать их в виде деревьев или списков с возможностями сортировки и фильтрации, находить быстрым поиском по подстроке (части названия), задавать значения любым свойствам любых объектов (в т.ч. несколько значений каждому атрибуту, если это предусмотрено информационной моделью), а также прикреплять к ним файлы и пользовательские комментарии.
- АрхиГраф.СУЗ обеспечивает создание и применение правил поиска дубликатов объектов по условиям, конструируемым администратором. Специальный интерфейс предназначен для просмотра и слияния объектов-дубликатов (нормализации данных).
- Многопользовательская работа и разграничение прав. Инструмент разграничения прав доступа пользователей к содержимому информационной модели предоставляет администратору возможность выбрать группу пользователей, и установить для нее права доступа ко всем или некоторым классам иерархии информационной модели в один из следующих уровней: нет доступа, только чтение, редактирование с подтверждением, редактирование.
- Интерфейс редактирования информационной модели учитывает права доступа пользователей к модели, выводит сообщение об ошибке при попытке выполнить операцию, не разрешенную текущему пользователю в соответствии с его правами.
- При редактировании содержимого информационной модели пользователь с правами «редактирование с подтверждением» на какой-либо элемент данных имеет возможность создать запрос на изменение этой записи. Интерфейс модератора - пользователя с полными правами на редактирование - позволяет просмотреть заявки на изменение элементов данных, подтвердить или отклонить каждый запрос.
- Редактор поддерживает импорт/экспорт фрагментов модели через файлы Excel.
- Редактор протоколирует все изменения в модели, выполняемые разными пользователями, и предоставляет возможность просмотра их истории.

Для отладки и настройки работы с платформой АрхиГраф полезна также форма тестирования XML- и JSON HTTP-запросов, доступная по адресу /mdm в любой инсталляции MDM. Форма имеет такой вид:

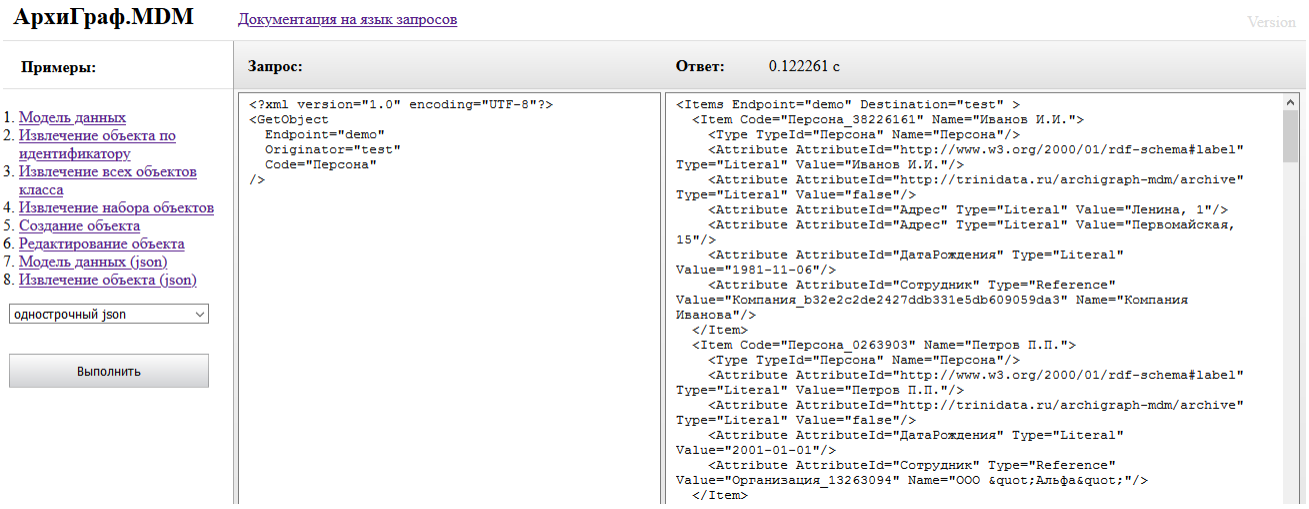

Рис. 2. Форма тестирования HTTP-запросов к платформе АрхиГраф

Справа расположены ссылки на примеры запросов. Нажатие на любую ссылку копирует текст запроса в поле «Запрос», где его можно изменить. Нажатие на кнопку «Отправить» выполняет обращение к платформе, ответ отображается в поле «Ответ».

Для выполнения HTTP-запросов к системе со стороны программных компонентов нужно отправить POST-запрос на URL /mdm, передав в параметре request тело XML- или JSON-запроса.

4. Основной API запросов и ответов АрхиГраф

4.1. Общие сведения

Пакеты в формате XML и JSON имеют схожую структуру.

Пример пакета в формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetDataSchema Endpoint="demo" Originator="test" />
```

Тот же пакет в формате JSON:

```
{"GetDataSchema": {"Endpoint":"demo", "Originator":"test"}}
```

Названия тегов и атрибутов не зависят от регистра.

Любые запросы, независимо от корневого тега, могут иметь следующие стандартные параметры:

Endpoint – код точки доступа[1], к которой относится запрос. Применим ко всем запросам за исключением запроса на получение списка точек доступа (GetEndpoints). Параметр не обязателен: если код точки доступа не передан, запрос будет отнесен к точке, определенной настройками системы как точка доступа по умолчанию. Если значение

[1] Точка доступа (endpoint) представляет собой обособленную часть логического пространства данных. Можно уподобить точки доступа отдельным базам данных в реляционной СУБД.

параметра передано во входном пакете, то это значение будет возвращено в корневом теге ответа в атрибуте Endpoint.

OperationId – идентификатор запроса. Применим ко всем запросам. Если OperationId передан во входном пакете, то его значение будет возвращено в атрибуте OperationId корневого тега ответа. Не обязательный, но необходим при обмене пакетами посредством очередей, так как позволяет сопоставить исходящий запрос с относящимся к нему ответом.

Originator – код запрашивающей информационной системы-клиента. Применим ко всем запросам. Если параметр передан в запросе, то его значение будет возвращено в атрибуте Destination корневого тега ответа. Информационная система с переданным кодом должна быть зарегистрирована в системе, и для нее должны быть настроены права доступа. В общем случае Originator не обязателен, но при запросе без указания системы будут применяться самые минимальные права доступа (права доступа для анонимных систем).

Token – применим ко всем запросам. Строка служит для обеспечения информационной безопасности: для идентификации системы проверяется соответствие кода системы, переданного в атрибуте Originator, с зарегистрированным для него токеном. Информационная система может быть зарегистрирована с пустым токеном, в этом случае параметр Token передавать не обязательно.

В случае ошибки при обработке любого запроса в ответ платформа возвращает пакет InvalidPackage.

Пакет **InvalidPackage** – сообщение о неверном запросе. Атрибуты пакета InvalidPackage:

Message - текст сообщения об ошибке

Пример пакета в формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<InvalidPackage Destination="test" Message="Object not found" />
```

Пример пакета в формате JSON:

```
{"InvalidPackage":{"Destination":"test", "Message":"Object not found"}}
```

4.2. Запросы для работы с точками доступа и хранилищами

4.2.1. Получение набора точек доступа

GetEndpoints – запрос на получение списка точек, доступ к которым разрешен для заданной информационной системы-клиента платформы.

Параметры

Не имеет собственных дополнительных параметров.

Пример запроса

В формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetEndpoints Originator="test" />
```

В формате JSON:

```
{"GetEndpoints":{"Originator":"test"}}
```

Ответ

Пакет **Endpoints**

Вложенные теги: *Endpoint*

Тег Endpoint содержит описание отдельной точки доступа.

Атрибуты тега Endpoint:
Code – код точки доступа. Используется в качестве значения атрибута Endpoint входящих пакетов;
Name - читаемое наименование точки доступа;
Default – является ли точкой доступа по умолчанию, значение true либо false.

Пример

В формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<Endpoints Destination="test">
  <Endpoint Code="demo" Name="Демонстрационная точка доступа" Default="true" />
</Endpoints>
```

В формате JSON:

```
{
   "Endpoints":
    {
    "Destination": "test",
    "Endpoint": [
        {
            "Code":"demo",
            "Name":"Демонстрационная точка доступа",
            "Default":"true"
        }
     ]
  }
}
```

4.2.2. Получение списка хранилищ

GetStorages – запрос на получение списка хранилищ с их характеристиками: набор попадающих классов, хранение истории, поддержка 4D-историйности, работа в режиме логической витрины данных (LDW), тип физического хранилища.

Параметры

В случае, если в запросе передан атрибут **EndpointCode**, будет возвращен список хранилищ, относящихся только к указанной точке доступа. Если код точки доступа не указан, будут возвращены хранилища всех точек, доступных информационной системе.

Для каждого хранилища возвращаются его параметры (код, наименование, реквизиты доступа и т.п.), а также список классов модели, объекты которого содержатся в указанном хранилище. По умолчанию будут возвращены лишь классы верхнего уровня, атрибут **WithSubclasses** позволяет получить полный перечень классов, с учетом наследуемых. Атрибут WithSubclasses принимает значения 0 и 1, значение по умолчанию 0.

Пример запроса

В формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetStorages EndpointCode="demo" Originator="test" WithSubclasses="1" />
```

В формате JSON:

```
{
  "GetStorages":{
     "EndpointCode":"demo",
     "Originator":"test",
     "WithSubclasses": "1"
  }
}
```

Ответ

Пакет **Storages**

Описание исходящего пакета Storages

Вложенные теги:

Storage – содержит описание хранилища данных

ObjectType – характеризует класс модели, объекты которого хранятся в данном хранилище

Тег Storage

Атрибуты тега Storage:

Code – код (идентификатор) хранилища в АрхиГраф
Name – читаемое наименование хранилища
Endpoint – код точки доступа, к которой относится хранилище
History – флаг, принимает значения 0 и 1. Показывает, запоминается ли для данного хранилища средствами АрхиГраф история изменения объектов, в нем содержащихся.
Timestamp – флаг, принимает значения 0 и 1. Определяет, поддерживает ли хранилище 4D-историйность данных. Такая поддержка обеспечивается, например, в HBase за счет хранения множества значений каждого атрибута объекта, причем каждое значение помечено timestamp.
Logic – флаг, принимает значения 0 и 1. Поддерживает ли хранилище работу с правилами логического вывода.
Type – тип физического хранилища (PostgreSQL, MongoDB и т.п.)

Атрибуты тега ObjectType:

Code – код АрхиГраф класса модели

Name – читаемое наименование класса

Пример ответа

В формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<Storages Destination="test">
  <Storage Code="storage1" Name="TDB для хранения модели" Endpoint="demo"
History="1" Timestamp="0" Logic="1" Type="Fuseki" />
  <Storage Code="storage2" Name="БД контрагентов" Endpoint="demo"
History="0" Timestamp="0" Logic="1" Type="MongoDB">
  <ObjectType Code="Контрагент" Name="Контрагенты" />
  </Storage>
</Storages>
```

В формате JSON:

```
{"Storages:
  {
    "Destination":"test",
    "Storage": [
      {"Code":"storage1", "Name": "TDB для хранения модели", "Endpoint":
"demo", "History":"1", "Timestamp":"0", "Logic":"1", "Type": "Fuseki"},
      {"Code":"storage2", "Name": "БД контрагентов", "Endpoint": "demo",
"History":"0", "Timestamp":"0", "Logic":"1", "Type":" MongoDB",
        "ObjectType": [
          {"Code":"Контрагент", "Name":"Контрагенты"}
        ]
      }
  ]
  }
}
```

4.2.3. Изменение параметров хранилища

UpdateStorage – запрос на редактирование изменяемых параметров хранилища.

Параметры

Вложенные теги: *Storage*.

Атрибуты тега Storage:

Code – обязателен, код изменяемого хранилища в АрхиГраф

Name – не обязателен, строка. Наименование хранилища.

History – не обязателен, 0 или 1. Отменить или установить ведение истории изменения объектов средствами АрхиГраф.

AddTypes – не обязателен, 0 или 1. Если передано значение 0 или атрибут не задан, то список классов, относящихся к хранилищу, будет заменен на список классов, переданных в текущем пакете в теге ObjectType. Если же передано значение 1, то список классов, относящихся к хранилищу, будет дополнен переданными классами.

DeleteTypes – не обязателен, 0 или 1. Если передано значение 1, то из списка классов, относящихся к хранилищу, будут исключены классы, переданные в пакете.

Вложенные теги: *ObjectType*

Атрибуты тега ObjectType:

Code – код АрхиГраф класса модели

Пример запроса

В формате XML:

```
<UpdateStorage Endpoint="demo" Originator="test">
  <Storage Code="storage2" AddTypes="1">
  <ObjectType Code="Подрядчик" />
  </Storage>
</UpdateStorage>
```

Ответ

OperationResults – пакет с информацией о результате выполнения операции. Подробнее описание пакета см. в разделе описания запроса UpdateObject.

4.2.4. Получение описания хранилища для класса

GetStoragesMapping – получить параметры физического хранения объектов класса (тип и реквизиты БД, коллекция/таблица, имена полей/столбцов соответствующих атрибутам). Может использоваться приложением для работы напрямую с хранилищем данных, например – для создания и выполнения заданий MapReduce и др.

Параметры

Вложенный тег *ObjectType* – классы модели, по которым требуется получить информацию

Атрибуты тега ObjectType:

Code – обязательный, код АрхиГраф класса модели

Пример запроса

В формате XML:

```
<GetStoragesMapping Endpoint="demo" Originator="test">
  <ObjectType Code="Контрагент" />
</GetStoragesMapping>
```

В формате JSON:

```
{"GetStoragesMapping": {
    "Endpoint":"demo",
    "Originator":"test",
    "ObjectType": [{"Code":"Контрагент"}]
  }
}
```

Ответ

Пакет **MapStorages**

Вложенные теги:

MapStorage – описание хранилища, содержащего объекты заданного класса,

MapAttribute – описание физического хранения значений атрибутов,

Target – только для атрибутов ссылочного типа – список классов модели, характеризующий область значения атрибута.

Атрибуты тега MapStorage:

ObjectType – класс модели, переданный в запросе
Code – код (идентификатор) хранилища в АрхиГраф
Name – читаемое наименование хранилища
Type – тип физического хранилища
Host, **Port**, **Login**, **Password**, **Database** – реквизиты подключения к БД
Table – коллекция/таблица, в которой хранятся объекты

Атрибуты тега MapAttribute (вложен в MapStorage):

AttributeId – идентификатор атрибута в модели
Field – поле/столбец для хранения значения атрибута в БД
Type – тип атрибута. Принимает значения Literal (литерал) либо Reference (ссылка на другой объект)
Datatype – тип для литеральных атрибутов. Принимаемые значения:

- xsd:string – строка
- xsd:boolean – флаг да/нет (true/false)
- xsd:integer – целое число
- xsd:double - число
- xsd:date - дата
- xsd:datetime - дата и время

Получить перечень всех литеральных типов, поддерживаемых платформой, можно запросив объекты специального класса archigraph:type.

Запрос для получения перечня литеральных типов в формате XML:

```
<GetObjectsGroup Code="archigraph:type" />
```

Запрос для получения перечня литеральных типов в формате json:

```
{"GetObjectsGroup":{"Code":"archigraph:type"}}
```

Атрибуты тега Target (вложен в MapStorage):

Code – код класса

Name – читаемое наименование класса

Пример ответа

В формате XML:

```
 <MapStorages Endpoint="demo" Destination="test">
  <MapStorage ObjectType="Контрагент" Code="storage2" "Name"="БД
контрагентов" Type="MongoDB" Host="12.12.12.123" Port="27017"
Database="demo" Table="clients">
```

```
    <MapAttribute AttributeId="ИНН" Field="inn" Type="Literal"
Datatype="xsd:string" />
    <MapAttribute AttributeId="Город" Field="city" Type="Reference">
    <Target TargetId="СправочникГород" Name="Города" />
  </MapAttribute>
  </MapStorage>
  <MapStorage ObjectType="СправочникГород" Code="storage1" Type="Fuseki"
Host="12.12.12.123" Port="8080" Table="demo" />
</MapStorages>
```

4.3. Запросы для работы с моделью

4.3.1. Получение информационной модели

GetDataSchema – запрос на получение структуры информационной модели.

Параметры

StartElement - стартовый класс интересующего фрагмента модели. Если параметр пропущен, возвращается содержимое всей модели.

WithoutSubClasses – необязательный атрибут, может принимать значения 0 и 1, значение по умолчанию 0. Имеет смысл только при указании StartElement. Если значение атрибута 0 или не задано, то возвращается описание модели как самого класса, заданного в атрибуте StartElement, так и всех его подклассов. Если же значение WithoutSubClasses=1, то возвращается описание только самого класса, указанного в StartElement.

WithoutInherited - необязательный атрибут, может принимать значения 0 и 1, значение по умолчанию 0. Если значение 0 или не задано, то описание класса содержит все атрибуты, применимые к объектам данного класса, включая атрибуты, унаследованные от родительских классов. Если же WithoutInherited=1, то для каждого класса будут возвращены только атрибуты, ассоциированные непосредственно с указанным классом, без наследуемых.

WithoutRangeInherited - необязательный атрибут, может принимать значения 0 и 1, значение по умолчанию 0. Если значение 0 или не задано, то описание области значений атрибутов-ссылок, включает полный перечень классов, включая подклассы. Если же WithoutInherited=1, то для области значений атрибута-ссылки будут возвращены только классы, ассоциированные непосредственно с атрибутом, без вложенных.

WithoutAttributes - необязательный атрибут, может принимать значения 0 и 1, значение по умолчанию 0. При задании WithoutAttributes будет возвращено описание классов без перечня атрибутов.

Пример запроса

В формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetDataSchema StartElement="Компании" WithoutSubClasses="1" />
```

Пример запроса в формате JSON:

```
{"GetDataSchema": {"StartElement":"Компании", "WithoutSubClasses": "1"}}
```

Ответ

DataSchema – пакет с описанием структуры информационной модели: перечнем всех описанных в ней классов, атрибутов и связей.

Атрибуты пакета DataSchema

StartElement – корневой элемент фрагмента модели, если был задан в запросе

Prefix – префикс названий элементов модели по умолчанию. Во всех остальных пакетах префикс не указывается, если он соответствует этому префиксу. Теоретически, в пределах одной модели данных могут использоваться разные префиксы. В этом случае идентификаторы элементов модели, имеющих другой префикс, будут иметь полный вид - http://.../.../[Имя элемента].

Тег *ObjectType* – передает информацию о каком-либо классе.

Атрибуты тега ObjectType:

Code – код АрхиГраф класса

Name – читаемое наименование класса

Archive – принимает значение true или false. Значением true помечаются устаревшие классы.

Тег *Parent* – передает информацию о том, что класс является потомком другого класса. Каждый класс может являться потомком любого числа других классов.

Атрибуты тега Parent:

ParentId – уникальный код класса-родителя

Тег *Attribute*

Передает информацию об атрибуте, присущем объектам каком-либо класса. Атрибуты наследуются классами рекурсивно, то есть если атрибут «ИНН» объявлен для класса «Юридические лица», он считается объявленным и для объектов вложенного в него класса «Компании».

Атрибуты тега Attribute

AttributeId – уникальный код атрибута

Name – читаемое наименование атрибута

Type – тип значения атрибута: Literal для атрибутов-литералов, Reference для атрибутов, хранящих ссылки на другие объекты.

DataType – используется для атрибутов-литералов, хранит тип значения. Указывается один из стандартных типов данных xsd: integer, string, date, dateTime, boolean, double.

MinCardinality – минимальное число значений этого атрибута для каждого объекта. Если MinCardinality =1, значит, атрибут обязателен.

MaxCardinality – максимальное число значений этого атрибута для каждого объекта. Атрибуты MinCardinality и MaxCardinalityмогут не указываться, тогда атрибут может принимать любое количество значений.

RangeIntersection – необязательный атрибут. Задается для свойств-ссылок в случае, если областью значений является пересечение нескольких классов (RangeIntersection="true"). По умолчанию, если атрибут RangeIntersection не указан, полагается, что область значения - объединение классов, указанных во вложенных тегах Target.

ClassSet – необязательный атрибут, может быть задан для свойств-литералов и свойств-ссылок. Используется, когда область применения свойства – пересечение нескольких классов. В этом случае в значении ClassSet указываются (через запятую) идентификаторы тех классов, к которым одновременно должен принадлежать объект, чтобы к нему было применимо описываемое свойство.

Archive – принимает значение true или false. Значение true соответствует устаревшим атрибутам, описание которых сохранено для обратной совместимости накопленных данных и текущей модели.

Тег *Target*

Указывает классы, объекты которых могут быть значениями для данного атрибута-ссылки.

Атрибуты тега Target:

TargetId – уникальный код класса

Name – читаемое наименование класса

Пример ответа

В формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<DataSchema StartElement="Компания" Prefix="http://some-model.ru/instance">
 <ObjectType Code="Компания" Name="Компании">
  <Parent ParentId="ЮридическиеЛица"/>
  <Attribute Type="Literal" AttributeId="Телефон" Name="телефон"
DataType="xsd:string" MinCardinality="1" />
 </ObjectType>
 <ObjectType Code="Персона" Name="Персоны">
  <Parent ParentId="Компания"/>
  <Attribute Type="Literal" AttributeId="ДатаРождения" Name="дата рождения"
DataType="xsd:date" MinCardinality="1" MaxCardinality="1"/>
  <Attribute Type="Reference" AttributeId="РаботаетВ" Name="работает в">
    <Target TargetId="Компания" Name="Компании" />
  </Attribute>
 </ObjectType>
</DataSchema>
```

В формате JSON:

```
{"DataSchema": {
 "StartElement":"Компания",
 "Prefix": "http://some-model.ru/instance",
 "ObjectType": [
  {
    "Code": "Компания",
    "Name": "Компании",
    "Parent": [ {"ParentId": "ЮридическиеЛица"} ],
    "Attribute": [
      {
        "Type": "Literal",
        "AttributeId": "Телефон",
        "Name": "телефон",
        "DataType": "xsd:string",
        "MinCardinality":1
      }
    ]
  },
  {
    "Code": "Персона",
    "Name": "Персоны",
    "Parent": [ {"ParentId": "Компания"} ],
    "Attribute": [
      {
        "Type": "Literal",
        "AttributeId": "ДатаРождения",
        "Name": "дата рождения",
        "DataType": "xsd:date",
        "MinCardinality":1,
        "MaxCardinality":1
      },
      {
        "Type": "Reference",
        "AttributeId": "РаботаетВ",
        "Name": "работает в",
        "Target": [ {"TargetId": "Компания", "Name": "Компании"} ]
      }
   ]
  }
  ]
 }
}
```

4.3.2. Получение информационной модели в компактной форме

GetDataSchemaCompact – запрос на получение структуры информационной модели в компактной форме

Параметры

StartElement – стартовый класс интересующего фрагмента модели. Если параметр пропущен, возвращается содержимое всей модели.

WithoutSubClasses – необязательный атрибут, принимает значения 0 и 1, значение по умолчанию 0. Имеет смысл только при указании StartElement. Если значение атрибута 0 или не задано, то возвращается описание модели как самого класса, заданного в атрибуте StartElement, так и всех его подклассов. Если же значение WithoutSubClasses=1, то возвращается описание только самого класса, указанного в StartElement.

WithoutInherited – необязательный атрибут, принимает значения 0 и 1, значение по умолчанию 0. Если значение 0 или не задано, то описание класса содержит все атрибуты, применимые к объектам данного класса, включая атрибуты, унаследованные от родительских классов. Если же WithoutInherited=1, то для каждого класса будут возвращены только атрибуты, ассоциированные непосредственно с указанным классом, без наследуемых.

Пример запроса

В формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetDataSchemaCompact StartElement="Компании"/>
```

В формате JSON:

```
{"GetDataSchemaCompact": {"StartElement":"Компании"}}
```

Ответ

Пакет **DataSchemaCompact -** структура модели данных в кратком формате

Пакет аналогичен описанному выше пакету DataSchema, за исключением того, что в нем перечень атрибутов не повторяется для каждого класса. Вместо этого набор атрибутов задается отдельно тегами AttributeDefinition, а для каждого класса указываются применимые к нему атрибуты тегами ApplicableAttribute (внутри тега ObjectType).

Тег *AttributeDefinition* – описывает атрибут, которым могут обладать объекты каких-либо классов модели.

Атрибуты тега AttributeDefinition:

AttributeId – уникальный код атрибута

Name – читаемое наименование атрибута

Type – тип значения атрибута: Literal для атрибутов-литералов, Reference для атрибутов, хранящих ссылки на другие объекты.

DataType – тип значения для атрибутов-литералов, один из стандартных типов данных xsd.

MinCardinality – минимальное число значений этого атрибута для каждого объекта. Если MinCardinality=1, значит, атрибут обязателен.

MaxCardinality – максимальное число значений этого атрибута для каждого объекта.

Archive – значение true соответствует устаревшим атрибутам модели.

Тег *Target*

Указывает классы, объекты которых могут быть значениями для данного атрибута-ссылки.

Атрибуты тега Target

TargetID – уникальный код класса

Name – читаемое наименование класса

Тег *ApplicableAttribute*

Передает информацию о том, что данный атрибут присущ объектам описываемого класса.

Атрибуты тега ApplicableAttribute

AttributeID – уникальный код атрибута.

Пример ответа

В формате XML:

```
<DataSchemaCompact Destination="test" Prefix="http://trinidata.ru/demo/"
StartElement="СправочникТиповыеРешения">
  <AttributeDefinition AttributeId="ОтноситсяКДисциплине" Name="Относится к
дисциплине" Archive="false" Type="Reference" >
    <Target TargetId="СправочникДисциплина" Name="Дисциплина" />
  </AttributeDefinition>
  <AttributeDefinition AttributeId="http://www.w3.org/2000/01/rdf-
schema#label" Name="Наименование" Type="Literal" DataType="xsd:string"
Archive="false" MaxCardinality="1" />
  <ObjectType Code="СправочникТиповыеРешения" Name="Типовые решения"
Archive="false" >
    <Parent ParentId="Справочники" />
    <ApplicableAttribute AttributeId="ОтноситсяКДисциплине" />
    <ApplicableAttribute AttributeId="http://www.w3.org/2000/01/rdf-
schema#label" />
  </ObjectType>
</DataSchemaCompact>
```

4.4. Запросы на получение информации об объектах

4.4.1. Получение информации об одном объекте

GetObject – запрос на получение конкретного объекта

Параметры

Code – код (идентификатор АрхиГраф) объекта, описание которого необходимо получить.

Пример запроса

В формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetObject Originator="test" Code="ИвановИИ" />
```

В формате JSON:

```
{"GetObject":{"Originator": "test", "Code": "ИвановИИ"}}
```

Ответ

Items – пакет с описанием объектов. Подробное описание пакета приведено ниже в описании запроса на получение набора объектов.

4.4.2. Получение информации о наборе объектов

GetObjectsGroup – запрос на получение всех объектов определенного класса или классов, удовлетворяющих заданным условиям фильтра.

Параметры

Запрос имеет два формата. В сокращенном формате запрос состоит из одного тега GetObjectsGroup, в атрибуте Code которого указывается конкретный тип объектов, который необходимо вернуть. Возвращаются все объекты, относящиеся к данному классу.

В полном формате тег GetObjectsGroup включает несколько вложенных тегов ObjectType, что позволяет выбрать экземпляры, относящиеся к нескольким классам (хотя бы к одному из перечисленных). Теги FilterGroup и Filter позволяют задать условия отбора. Описание их синтаксиса приведено ниже.

Тег *GetObjectsGroup*

Корневой тег запроса.

Атрибуты:

Code – идентификатор класса, объекты которого необходимо вернуть (при сокращенном формате запроса).

Все остальные теги и атрибуты используются только в полном формате запроса.

Атрибуты корневого тега:

ObjectTypeGroupOperation — может принимать значение and или or. Используется для обработки условий принадлежности объектов классам, задаваемых в тегах ObjectType. Атрибут не является обязательным; если он не указан, то это равносильно заданию значения or (логическое ИЛИ).

CombineGroups – необязательный атрибут, принимает значения and или or. Используется для объединения в группу несколько условий отбора, заданных тегами FilterGroup. Если атрибут не указан, это равносильно значению and (логическое И).

CombineGeometry – необязательный атрибут, принимает значения and или or. Используется для объединения в группу несколько условий отбора по координатам, заданных тегами Geometry. Если атрибут не указан, это равносильно значению or (логическое ИЛИ).

ReturnCount – необязательный атрибут, принимает значения 0 и 1, по умолчанию значение равно 0. В случае переданного значения атрибута 1 в ответном пакете вместо информации об объектах будет возвращено их количество (в атрибуте Count тега Items).

WithoutSubClasses – необязательный атрибут со значениями 0 и 1, по умолчанию считается равным 0. Если в атрибуте WithoutSubClasses передано значение 1, то в условиях отбора будут учитываться только объекты, принадлежащие непосредственно заданным классам. Если атрибут WithoutSubClasses равен 0 или не задан, то в формировании условия на отбор объектов учитываются так же все подклассы классов, переданных в тегах ObjectType и/ атрибуте Code тега GetObjectsGroup.

ReturnCodeOnly – необязательный атрибут, принимает значения 0 и 1, по умолчанию равен 0. В случае, если ReturnCodeOnly равен 1, в ответе вместо полной информации об объекте будут возвращены только идентификаторы объектов, удовлетворяющих условиях фильтров: для каждого объекта возвращается тег Item с единственным атрибутом Code и без вложенных тегов Type и Attribute.

Limit – целое число: ограничение на количество возвращаемых объектов. Не обязательный атрибут, значение по умолчанию 1000.

Offset – необязательный атрибут, по умолчанию равен 0. Целое число, смещение: количество пропускаемых в результате поиска объектов. Найденные объекты нумеруются с 0.

Тег *ObjectType*

Позволяет указать классы, к которым могут относиться результаты запроса.

Атрибуты:

Code – идентификатор класса, объекты которого необходимо вернуть.

В составе запроса может присутствовать несколько тегов ObjectType с разными значениями Code. Возвращаются объекты, относящиеся в зависимости от значения атрибута ObjectTypeGroupOperation тега GetObjectsGroup либо к хотя бы к одному из перечисленных классов (при ObjectTypeGroupOperation=«or», или если значение атрибута ObjectTypeGroupOperation не указано) либо ко всем перечисленным классам

(если значение ObjectTypeGroupOperation равно «and»). При этом учитывается вложенность классов: объекты, принадлежащие подклассам запрошенных классов, тоже возвращаются (если не передан атрибут WithoutSubClasses=1 в корневом теге запроса).

Тег *FilterGroup*

Объединяет в группу несколько условий отбора результатов запроса. За логику объединения отвечает атрибут CombineGroups корневого тега GetObjectsGroup. По умолчанию группы объединены логическим И.

Атрибуты тега FilterGroup

Operation – логическая операция, которой объединяются условия фильтра, заданные внутри группы. Принимает значения and и or.

Тег *Filter*

Описывает одно из условий отбора результатов запроса.

Атрибуты тега Filter

Attribute – идентификатора атрибута, к которому применяется условие.

Value – литерал, с которым сравнивается значение указанного атрибута у всех отбираемых результатов. Если атрибут является указателем на другой объект, то Value должен содержать идентификатор объекта.

Comparison – операция сравнения. Принимаемые значения:

- Equal (равно),
- Contains (содержит),
- More (больше),
- Less (меньше),
- MoreOrEqual (больше или равно),
- LessOrEqual (меньше или равно),
- NotEqual (не равно),
- Exists (значение задано),
- NotExists (значение атрибута не задано),
- iEqual (равно с точностью до регистра).

Variable – необязательный атрибут, содержит имя переменной, участвующей в условии (для многоуровневых запросов). Использование в фильтрах переменных позволяет строить достаточно сложные условия отбора, учитывающие цепочки связей, и налагать условия не только на свойства самого объекта, но на свойства связанных с ним объектов.

ByName – необязательный атрибут, принимает значения 0 и 1, значение по умолчанию - 0. Имеет смысл только для атрибутов-указателей. В случае, если ByName равно 1, в условии отбора для сравнения с Value используется не значение атрибута, а читаемое наименование объекта, на который атрибут указывает.

Тег *Geometry*

Позволяет вести поиск объектов по географическим координатам, сохраненным в формате GeoJSON. Применим не для всех данных, а только тех, хранилище которых допускает такого рода поиск. В случае, если хранилище не поддерживает геопоиск, условие всегда будет ложно. С остальными условиями ограничение на координаты объединяется логическим И. Если задано несколько тегов Geometry, то между собой они будут объединены в зависимости от значения атрибута CombineGeometry головного тега GetObjectsGroup – логическое И либо ИЛИ, по умолчанию – ИЛИ.

Атрибуты тега Geometry:

Type – тип географического объекта: точка(point), полигон (polygon).

Attribute – атрибут, хранящий координаты

Тег *Coordinates*

Задает координаты точек – границ полигона в поиске по GeoJSON

Атрибуты тега Coordinates

Longitude - географическая долгота – вещественное число из диапазона (-180,180]

Latitude – географическая широта – вещественное число из диапазона (-90,90]

Тег *Item*

Позволяет передать идентификаторы искомых объектов. Условия тега Item объединяются логическим И с остальными условиями: если какие-то из объектов с переданными идентификаторами не удовлетворяют ограничениям, заданным с помощью тегов ObjectType, FilterGroup, Geometry и атрибутов головного тега, то в ответе данные объекты возвращены не будут.

Атрибуты тега Item

Code – идентификатор объекта

Пример запроса с использованием тега Item: среди заданных идентификаторами сотрудников вернуть только тех, кто работает в компании Альфа:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetObjectsGroup>
  <ObjectType Code="Сотрудники"/>
  <FilterGroup Operation="And">
    <Filter Attribute="РаботаетВ" Value="Альфа" Comparison="Equal">
  </FilterGroup>
  <Item Code="ИвановИИ" />
  <Item Code="ПетровПП" />
  <Item Code="СидоровСС" />
</GetObjectsGroup>
```

Пример запроса с использованием тега Item в формате json:

```
{"GetObjectsGroup": {
    "ObjectType":[
    {"Code":"Сотрудник"}
  ],
    "FilterGroup":[
   {
```

```
      "Operation":"And",
      "Filter":[
        {
              "Attribute":"РаботаетВ",
              "Value":"Альфа",
              "Comparison":"Equal"
            }
      ]
    }
     ],
     "Item": [
     {"Code":"ИвановИИ"},
     {"Code":"ПетровПП"},
     {"Code":"СидоровСС"}
     ]
   }
}
```

Тег *Sort*

Задает сортировку найденных объектов по указанным атрибутам в порядке возрастания или убывания

Атрибуты тега Sort

AttributeId – идентификатор атрибута, по которому производится сортировка. Обязательный атрибут.

Direction – Направление сортировки – по возрастанию (значение атрибута ASC) либо убыванию (значение DESC). Атрибут не является обязательным, по умолчанию производится сортировка по возрастанию значений атрибута (ASC)

SortByName – необязательный атрибут, принимающий значения 0 и 1. Имеет смысл только при сортировке с по значениям атрибутов-указателей. В случае, если SortByName равно 1, сортировка производится не по значениям атрибута, а по читаемым названиям объектов, на которые ссылается атрибут.

Пример запроса с сортировкой - отсортировать происшествия по наименованию района, в котором оно произошло, а в рамках района – по дате в обратном порядке:

```
<GetObjectsGroup>
  <ObjectType Code="Происшествие"/>
  <FilterGroup Operation="And">
    <Filter Attribute="Дата" Value="2018-06-01" Comparison="More" />
  </FilterGroup>
<Sort AttributeId="ПроизошлоВРайоне" Direction="ASC" SortByName="1" />
<Sort AttributeId="Дата" Direction="DESC"/>
</GetObjectsGroup>
```

Пример запроса с сортировкой в формате JSON:

```
{"GetObjectsGroup": {
    "ObjectType":[{"Code":"Происшествие"}],
    "FilterGroup":[
   {
     "Operation":"And",
     "Filter":[
       {
             "Attribute":"Дата",
             "Value":"2018-06-00",
```

```
            "Comparison":"More"
          }
      ]
    }
     ],
   "Sort":[
       {
      "AttributeId":"ПроизошлоВРайоне",
          "Direction":"ASC",
          "SortByName":"1"
    },
       {
      "AttributeId":"Дата",
      "Direction":"DESC"
    }
  ]
   }
}
```

Тег *FieldSet*

Позволяет в ответе получить не все атрибуты объекта, а только необходимые.

Атрибуты тега FieldSet:

Exclude – принимает значения 0 и 1, по умолчанию равен 0. Если значение равно 1, то из списка атрибутов объекта в ответе исключаются указанные. Если атрибут равен 0 или не задан, то для объектов будут возвращены значения указанных атрибутов.

Тег *Field* – вложен в FieldSet

Описание атрибутов, которые требуется либо вернуть, либо исключить из ответа в зависимости от значения параметра Exclude.

Параметры тега Field:

AttributeId – идентификатор атрибута

Variable – не обязательный - идентификатор переменной, из которой берется возвращаемое значение. Используется для запросов с подзапросами.

SetVariable – не обязательный, идентификатор переменной, в которую следует положить значение атрибута. Используется для запросов с подзапросами.

Тег *Subrequest* – задает подзапрос

Допускает все те же параметры и теги, которые описаны для тега GetObjectsGroup. Помимо этого, может иметь необязательный атрибут SetVariable – если он задан, то в значение переменной с указанным именем попадут идентификаторы объектов, отобранных условием подзапроса.

Пример запроса

В сокращенной форме в формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetObjectsGroup Code="Компании" ></GetObjectsGroup>
Пример запроса в сокращенной форме в формате json:
{"GetObjectsGroup":{"Code":"Компании"}}
```

В полной форме в формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetObjectsGroup>
  <ObjectType Code="Компании"/>
  <ObjectType Code="Организации"/>
  <FilterGroup Operation="Or">
    <Filter Attribute="Название" Value="альфа" Comparison="equal">
    <Filter Attribute="Название" Value="бета" Comparison="contains">
  </FilterGroup>
</GetObjectsGroup>
```

В полной форме в формате JSON:

```
{"GetObjectsGroup":{
  "Originator":"test",
  "ObjectType":[
    {"Code":"Компании"},
    {"Code":"Организации"}
  ],
  "FilterGroup":[
    {
      "Operation":"Or",
      "Filter":[
      {"Attribute":"Название", "Value":"альфа", "Comparison":"equal"},
      {"Attribute":"Название", "Value":"бета", "Comparison":"equal"}
      ]
    }
  ]
 }
}
```

Пример запроса с использованием геометрии

В формате XML:

```
<GetObjectsGroup>
  <ObjectType Code="Происшествие"/>
  <FilterGroup Operation="And">
    <Filter Attribute="Дата" Value="2018-06-00" Comparison="More" />
  </FilterGroup>
   <Geometry Type="Polygon" Attribute="Координаты">
    <Coordinates Longitude="30.3837" Latitude = "59.8768" />
    <Coordinates Longitude="30.3837" Latitude = "59.9168" />
    <Coordinates Longitude="30.4639" Latitude = "59.9168" />
    <Coordinates Longitude="30.4639" Latitude = "59.8768" />
  </Geometry>
</GetObjectsGroup>
```

Поиск по координатам в формате JSON:

```
{"GetObjectsGroup": {
    "ObjectType":[{"Code":"Происшествие"}],
    "FilterGroup":[
   {
     "Operation":"And",
     "Filter":[
       {
             "Attribute":"Дата",
             "Value":"2018-06-00",
             "Comparison":"More"
         }
```

```
      ]
    }
     ],
     "Geometry": [
     {
       "Type":"Polygon",
       "Attribute":"Координаты",
           "Coordinates":[
             {"Longitude":"30.3837", "Latitude":"59.8768"},
             {"Longitude":"30.3837", "Latitude":"59.9168"},
             {"Longitude":"30.4839", "Latitude":"59.9168"},
             {"Longitude":"30.4839", "Latitude":"59.8768"}
           ]
     }
     ]
  }
}
```

Пример запроса с использованием FieldSet

В формате XML – вернуть для объектов класса Парк только значение атрибута Территория:

```
<GetObjectsGroup Code="Парк" Limit="20" Offset="0">
  <FieldSet Exclude="0">
          <Field AttributeId="Территория" />
  </FieldSet>
</GetObjectsGroup>
```

В формате JSON:

```
{"GetObjectsGroup": {
  "Code":"Парк",
  "Limit":"20",
  "Offset":"0",
  "FieldSet": [
    {
    "Exclude":"0",
        "Field": [
          {"AttributeId":"Территория"}
     ]
   }
    ]
  }
}
```

Пример запроса с подзапросом: найти всех сотрудников действующих компаний с фамилией Иванов. В ответе получить для каждого сотрудника его ФИО и название компании, в которой он работает:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetObjectsGroup>
  <ObjectType Code="Сотрудник"/>
  <FilterGroup Operation="And">
    <Filter Attribute="ФИО" Value="Иванов" Comparison="Contains">
    <Filter Attribute="РаботаетВ" Variable="?x" Comparison="Equal">
  </FilterGroup>
  <Subrequest SetVariable="?x">
    <ObjectType Code="Компания"/>
    <FilterGroup Operation="And">
     <Filter Attribute="Действующая" Value="true" Comparison="Equal" />
    </FilterGroup>
    <FieldSet>
```

```
      <Field AttributeId="ПолноеНаименование" SetVariable="?y" />
    </FieldSet>
  </Subrequest>
  <FieldSet>
    <Field AttributeId="ФИО" />
    <Field AttributeId="РаботаетВ" Variable="?y" />
  </FieldSet>
  <Sort AttributeId="ФИО" Direction="ASC" />
</GetObjectsGroup>
```

Ответ

Пакет **Items**

Пакет Items содержит информацию о конкретных объектах. Данный пакет может быть получен в качестве результата выполнения запросов GetObject, GetObjectsGroup, GetObjectHistory.

Пакет состоит из одного или нескольких элементов Item, каждый из которых передает информацию об одном объекте. В рамках одного пакета Items может прийти информация о нескольких взаимосвязанных сущностях разных типов, или о нескольких сущностях одного типа.

Каждый объект (сущность) характеризуется следующими параметрами:

- **Идентификатор** – глобальный код объекта, присвоенный системой АрхиГраф. Обычно выглядит как URI, то есть адрес вида http://some-domain.ru/prefix/object. Первая часть идентификатора, до object, называется префиксом - обычно она одинакова для всех элементов модели. Стандартный префикс может указываться, или пропускаться. Префикс, отличный от стандартного, указывается в любом случае. Последняя часть URI, object, является собственно уникальным кодом объекта.
- **Наименование** – стандартное свойство, присутствующее по умолчанию у всех элементов. Представляет собой читаемое название элемента.
- Набором **типов** – перечнем классов, к которым относится объект. Каждый объект может относиться к любому количеству классов. Принадлежность к классам рекурсивна, т. е. для АрхиГраф тот факт, что объект принадлежит к некому классу иерархии, означает, что он одновременно является членом всех вышележащих классов.
- Набором **атрибутов** – перечнем значений свойств данного объекта. Для свойств-литералов указывается непосредственное значение, для свойств-связей - уникальный идентификатор объекта, на которое ссылается свойство. В семантической модели каждое свойство каждого объекта может иметь столько значений, сколько разрешает информационная модель.

Пример пакета, иллюстрирующий его общую структуру:

```
<?xml version="1.0" encoding="UTF-8"?>
<Items Destination="test">
  <Item Code="ИвановИИ" Name="Иванов И.И.">
    <Type TypeId="Персона" Name="Персоны" />
    <Type TypeId="Сотрудник" Name="Сотрудники" />
    <Attribute Type="Literal" AttributeId="ДатаРождения" Value="1979-01-01"/>
    <Attribute Type="Reference" AttributeId="РаботаетВ" Value="ОООАльфа"/>
```

```
  </Item>
</Items>
```

Пример пакета Items в формате JSON:

```
{"Items": {
  "Destination":"test",
  "Item": [
    {
    "Code":"ИвановИИ",
    "Name":"Иванов И.И.",
    "Type": [
      {"TypeId":"Персона", "Name":"Персоны"},
      {"TypeId":"Сотрудник", "Name":"Сотрудники"}
    ],
    "Attribute": [
      {
        "Type":"Literal",
        "AttributeId":"ДатаРождения",
         "Value":"1979-01-01"
      },
      {
        "Type":"Reference",
        "AttributeId":"РаботаетВ",
        "Value":"ООООльфа"
       }
    ]
  }
    ]
   }
}
```

Тег *Item*

Передает информацию о конкретном объекте. Тег Item может быть либо возвращен в составе пакетов Items, SubscriptionItems, SubscriptionDeleteItems, либо получен в составе входящего пакета UpdateObject, содержащего запрос на редактирование/создание объекта. Ниже приводится описание атрибутов тега Item с уточнением, в каком случае он может указан.

Атрибуты тега Item

Code – уникальный код объекта

Name – читаемое имя объекта

LocalCode – временный код объекта, используемый в случае, когда тег Item передается в составе пакета UpdateObject, содержащего запрос на создание объекта. Значением атрибута является внутренний код объекта в той системе, в которой он был создан. Атрибут Code при этом передается пустым. В ответном пакете OperationResults система возвращает присвоенный объекту постоянный уникальный код, а также соответствие между ним и временным кодом.

OperationId – уникальный идентификатор операции на создание/изменение объекта. Используется при передаче тега Item в составе пакета UpdateObject. Генерируется приложением-отправителем, используется для получения информации о результате операции.

DomainIntersection – необязательный атрибут. Может использоваться только при передаче в Item данных о свойствах классов (литеральных и ссылочных). Значение DomainIntersection=true указывается, если областью применимости свойства является пересечение набора классов. По умолчанию в случае задания у свойства нескольких классов в атрибуте domain областью применимости свойства будет объединение этих классов; DomainIntersection в теге Item в этом случае не указывается.

RangeIntersection – необязательный атрибут, так же относится только к передаче данных о свойствах. RangeIntersection=true задается, если областью значений свойства-ссылки является пересечение классов; в случае области значений -объединения классов RangeIntersection не указывается.

IgnoreTypes – необязательный атрибут используется при передаче тега Item в составе пакета UpdateObject, принимает значения 0 или 1, значение по умолчанию 0. При указании атрибута равным 1 принадлежности объекта классам в тегах Type внутри тега Item можно не указывать; если же теги Type будут заданы, то их значения при обработке запросы будут проигнорированы - сохранится существующая принадлежность объекта классам. Имеет смысл только для случая редактирования объекта. Запрос на создание объекта с указанием IgnoreTypes=1 вернет сообщение об ошибке.

AddTypes – необязательный атрибут используется при передаче тега Item в составе пакета UpdateObject, принимает значения 0 или 1, значение по умолчанию 0. При задании AddTypes=1 классы, переданные в тегах Type будут добавлены к классам объекта (если они были заданы ранее). При AddTypes=0 и без указания IgnoreTypes=1 (по умолчанию) набор классов, которым принадлежит объект, будет заменен на набор классов, переданных в тегах Type.

CreateIfNotExists – необязательный атрибут, используется при передаче тега в составе UpdateObject. Если для тега Item передано значение атрибута CreateIfNotExists=1 и задан атрибут Code, но в системе нет атрибута с указанным идентификатором, то будет создан новый объект с идентификатором, значение которого равно значению атрибута Code. Значение CreateIfNotExists по умолчанию 0 - тогда, если объект с идентификатором из атрибута Code в системе отсутствует, будет возвращено сообщение об ошибке.

FullUpdate - необязательный атрибут, используется при передаче тега в составе UpdateObject. Принимает значение 0 и 1, значение по умолчанию 0. Если атрибут FullUpdate не задан или равен 0, то изменения затронут только те атрибуты, которые были переданы запросом. В противном случае объект будет обновлен полностью: у объекта будут удалены все существующие атрибуты, которые не переданы в запросе, за исключением атрибутов система-источник и код в системе источнике.

Date - тип дата и время - возвращается в случае, если Item содержит ответ на запрос получения состояния объекта на указанную дату (GetObject или GetObjectsGroup с дополнительным параметром Date).

Тег *Type* – вложен в Item

Передает информацию о том, каким классам принадлежит данный объект. Может встречаться несколько раз для одного объекта.

Атрибуты тега Type

TypeId – идентификатор класса, которому принадлежит объект.

Name – читаемое наименование класса. Имеет смысла только для исходящих пакетов. В случае, если значение атрибута Name передано в рамках запроса UpdateObject, оно будет проигнорировано.

Теги *Operation* – вложен в Item

Возвращается в случае получения ответа на запрос GetObjectHistory и содержит информацию об истории изменения принадлежности класса объекта.

Атрибуты тега Operation:

Date – тип дата и время – момент установки указанного значения

System – код информационной системы, произведшей изменение значения

OperationId – идентификатор операции, в результате которой было изменено значение (если был указан в запросе на изменение)

Вложенные теги для тега Operation: Set (один, несколько или ни одного в случае, если значение атрибута было очищено). Единственный атрибут тега Set – Value – содержит установленное значение.

Тег *Attribute* – вложен в Item

Передает информацию о значении какого-либо атрибута объекта. Может встречаться несколько раз для одного атрибута, если модель данных это позволяет.

Атрибуты тега Attribute

Type – тип значения атрибута: Literal, если значение представляет собой константу, Reference – если значение является уникальным идентификатором другого объекта, и LocalCodeReference, если ссылка осуществляется на другой объект по его временному коду. Последний вариант используется только в случаях, когда тег Item передается в составе пакета UpdateObject. При этом пакет может содержать запрос на создание сразу нескольких сущностей, связанных между собой. Значение LocalCodeReference используется для указания на то, что значение Value ссылается на другой объект, создаваемый в этом же запросе, по его временному коду.

AttributeId – идентификатор атрибута в модели данных.

Value – значение атрибута.

Name – читаемое наименование объекта, идентификатор которого содержится в значении атрибута. Возвращается только для атрибутов ссылочного типа. Будет проигнорировано в случае, если значение атрибута Name передано в рамках запроса UpdateObject.

Ignore – используется при передаче тега Item в составе пакета UpdateObject, и указывает, что система не должна изменять значение этого атрибута.

Empty – используется при передаче тега Item в составе пакета UpdateObject, его задание указывает, что значение атрибута должно быть очищено.

ExistingOnly – используется при передаче тега Item в составе пакета UpdateObject, указывает, что значение атрибута изменяется только в том случае, если оно ранее уже было установлено.

AddValue - используется при передаче тега в составе пакета UpdateObject, указывает, что если значение или значения для атрибута уже были заданы, то новое значение не перезаписывает существующие, а добавляется к ним. Имеет смысл для многозначных атрибутов.

4.5. Запросы на изменение данных

Для всех запросов на редактирование обязательно задание кода информационной системы в атрибуте Originator корневого тега.

4.5.1. Редактирование/создание объекта

UpdateObject – запрос на создание/изменение конкретного объекта или набора объектов, заданных своими идентификаторами.

Параметры

Пакет содержит теги Item, описание которых приводится в структуре пакета Items, содержащего информацию о каком-либо объекте. В одном пакете может быть передан запрос на изменение сразу нескольких объектов. Для каждого объекта передается свой тег Item. Тег Item в рамках пакета на изменение обязательно должен содержать значение атрибут OperationId – данный идентификатор будет указан в ответе и позволяет сопоставить запрос на редактирование конкретного объекта с результатом выполнения.

Пример запроса

На редактирование объекта в формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<UpdateObject Originator="test" OperationId="000004">
  <Item Code="Персона_1"   OperationId="0000041">
    <Type TypeId="Персона" />
    <Attribute Type="Literal" AttributeId="http://www.w3.org/2000/01/rdf-
schema#comment" Value="Лучший сотрудник" />
  </Item>
</UpdateObject>
```

Пример запроса на редактирование объекта в формате JSON:

```
{"UpdateObject": {
 "Originator":"test",
 "OperationId":"000004",
 "Item": [
    {
   "Code":"Персона_1",
   "OperationId":"0000041",
     "Type": [
     {"TypeId":"Персона"}
```

```
    ],
    "Attribute": [
     {
       "Type":"Literal",
       "AttributeId":"http://www.w3.org/2000/01/rdf-schema#comment",
           "Value":"Лучший сотрудник"
           }
     ]
     }
   ]
  }
}
```

Пример запроса на создание объекта в формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<UpdateObject Originator="test" OperationId="000004">
  <Item LocalCode="1298"   OperationId="0000041">
    <Type TypeId="Персона" />
    <Attribute Type="Literal" AttributeId="ФИО" Value="Хохолев Х.Х." />
  </Item>
</UpdateObject>
```

При редактировании объектов по умолчанию будут изменены только те атрибуты, которые были переданы в тегах Attribute. Отсутствующие в запросе атрибуты объекта сохранят свои значения. Если требуется оставить у объекта только переданные атрибуты и удалить все прочие, то необходимо указать флаг FullUpdate=1 в теге Item.

Для очистки значений конкретного атрибута используется флаг Empty в теге Attribute. Пример запроса, при котором происходит удаление всех значений атрибута Адрес:

```
<?xml version="1.0" encoding="UTF-8"?>
<UpdateObject Originator="test" OperationId="000004">
  <Item Code="Организация_1"   OperationId="0000041">
    <Type TypeId="Организация" />
    <Attribute Type="Literal" AttributeId="Адрес" Empty="1" />
  </Item>
</UpdateObject>
```

Если атрибут принимает множественные значения, то при редактировании будут удалены все старые значения атрибута и присвоены значения, переданные в запросе. Для добавления значений к списку уже заданных ранее значений необходимо использовать флаг AddValue для тега Attribute.

Пример использования AddValue. Пусть у объекта *Организация_1* значение атрибута УчаствуетВПроекте равно *Проект_1.* В результате выполнения запроса

```
<UpdateObject Originator="test" OperationId="000004">
  <Item Code="Организация_1"   OperationId="0000041">
    <Type TypeId="Организация" />
    <Attribute Type="Reference" AttributeId="УчаствуетВПроекте"
Value="Проект_2"/>
  </Item>
</UpdateObject>
```

атрибут УчаствуетВПроекте примет значение *Проект_2.* При выполнении же запроса

```
<UpdateObject Originator="test" OperationId="000004">
  <Item Code="Организация_1"   OperationId="0000041">
    <Type TypeId="Организация" />
```

```
    <Attribute Type="Reference" AttributeId="УчаствуетВПроекте"
Value="Проект_2" AddValue="1" />
  </Item>
</UpdateObject>
```

атрибуту УчаствуетВПроекте будет иметь два значения *Проект_1* и *Проект_2.*

Флаг Ignore тега Attribute означает, что данный атрибут не нужно учитывать при редактировании; теги Attribute с данным флагом будут пропущены при обработке пакета.

При редактировании объекта может быть использован флаг IgnoreTypes для тега Item. В этом случае вложенные теги Type можно передавать или не передавать – в любом случае объект сохранит принадлежность тем классам, которая у него была. Если же флаг IgnoreTypes не указан, то теги Type являются обязательными и произойдет присвоение объекту списка классов, переданных в запросе. При создании нового объекта атрибут Type является обязательным, а использование флага IgnoreTypes приведет к возвращению ошибки.

Другой флаг, влияющий на обработку принадлежности объекта классам - AddTypes. При задании AddTypes=1 для существующего объекта список классов, которым он принадлежит, будет дополнен списком переданных в запросе классов. Для нового объекта произойдет присвоение списка переданных классов так же, как без флага AddTypes. При одновременном задании значения 1 для флагов IgnoreTypes и AddTypes флаг AddTypes будет проигнорирован.

Ответ

Пакет **OperationResults** - пакет с информацией о результатах выполнения операции.

Вложенные теги: OperationResult – результат выполнения запроса для отдельного объекта.

Атрибуты тега OperationResult:

OperationId – идентификатор запроса, значение из тега Item входного пакета.

Result – может вернуть значения *success*, *error* либо *proposed*. Значение *success* вернется в случае успешного изменения объекта, значение *error* - в случае какого-либо сбоя, возникшего при выполнении запроса. В случае, если система, выполняющая запрос, имеет к изменяемому объекту права на редактирование с подтверждением, и произошло успешное добавление запроса на подтверждение в базу, в атрибуте Result будет возвращено значение *proposed*.

Message – текст сообщения об ошибке в случае сбоя либо поясняющий операцию текст в других случаях; например, при групповой операции Message будет содержать информацию о числе измененных или удаленных объектов.

Code – код АрхиГраф измененного/созданного объекта

LocalCode - код в системе источники объекта - в случае, если данное значение было передано в запросе.

Пример ответа

В формате XML

```
<OperationResults Destination="test" >
  <OperationResult Result="success" OperationId="0000041" Code="Персона_1"
LocalCode="1297" />
  <OperationResult Result="error" Message="Unknown code Персона_007"
OperationId="0000042" Code="Персона_007" />
</OperationResults>
```

Пример ответа в формате JSON:

```
{"OperationResults":  {
 "Originator":"test",
 "OperationId":"000004",
 "OperationResult":   [
    {
    "Result":"success",
    "Message":"Unknown code Персона_007",
      "OperationId":"0000041",
    "Code":"Персона_1",
    "LocalCode":"1297"
  },
    {
    "Result":"error",
    "Message":"Unknown code Персона_007",
      "OperationId":"0000042",
    "Code":"Персона_007"
  }
  ]
 }
}
```

Кроме того, после успешного выполнения операции АрхиГраф отправляет пакет SubscriptionItems, который получают все информационные системы, заинтересованные в сведениях об объектах такого типа. Формат пакета SubscriptionItems полностью повторяет формат пакета Items, описанного в предыдущем разделе. Для объектов в пакете SubscriptionItems передается полный набор их атрибутов, а не только измененные.

4.5.2.Удаление объекта

DeleteObject - запрос на удаление конкретного объекта, заданного идентификатором.

Параметры

Code – код АрхиГраф объекта.

Ответ

Пакет **OperationResults** - пакет с информацией о результатах выполнения операции. Кроме того, после успешного выполнения операции АрхиГраф отправляет информационным системам, подписанным на изменение объектов данного типа, пакет SubscriptionDeleteItems. Структура пакета идентична пакету Items, он содержит тег Item с описанием удаляемой сущности.

Пример запроса

В формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<DeleteObject Code="ИвановИИ" Originator="test" />
```

В формате json:

```
{"DeleteObject":{"Code":"ИвановИИ", "Originator":"test" }}
```

4.5.3. Редактирование набора объектов

UpdateObjectsGroup – запрос на изменение группы объектов.

Параметры

Запрос имеет два формата — полный и сокращенный, аналогичные форматам запроса GetObjectsGroup.

В сокращенном формате запрос состоит из тега UpdateObjectsGroup и одного или нескольких вложенных в него тегов Attribute, характеризующих те атрибуты объектов, которые будут изменены. Описание тега Attribute приводится в структуре пакета Items. Для тега UpdateObjectsGroup задается дополнительный атрибут Code – идентификатор класса модели. При выполнении запроса краткого формата будут изменены все объекты, относящиеся к указанному классу.

В полном формате в отличие от краткого тег UpdateObjectsGroup не имеет атрибута Code, а вместо этого включает в себя несколько вложенных тегов ObjectType, FilterGroup, Item, Geometry, позволяющих задать условие для отбора тех объектов, которые будут изменены. Синтаксис тегов ObjectType, FilterGroup, Item, Geometry приведен в структуре пакета GetObjectsGroup.

Ответ

Пакет **OperationResults** – пакет с информацией о результате выполнения операции. В тексте ответа будет возвращена информация о количестве измененных объектов.

Также АрхиГраф рассылает пакеты SubscriptionItems со всеми измененными в результате выполнения запроса объектами. Пакеты SubscriptionItems получат все информационные системы, имеющие право доступа к этим объектам.

Примеры запросов

Краткий формат:

```
<?xml version="1.0" encoding="UTF-8"?>
 <UpdateObjectsGroup Code="Компании" Endpoint="demo" Originator="test"
OperationId="0945ab454">
 <Attribute Type="Literal" AttributeId="ДействующаяКомпания" Value="true" />
</UpdateObjectsGroup>
```

Полный формат:

```
<?xml version="1.0" encoding="UTF-8"?>
<UpdateObjectsGroup Endpoint="demo" Originator="test"
OperationId="0945ab4fr4">
  <ObjectType Code="Компании"/>
  <ObjectType Code="Организации"/>
  <FilterGroup Operation="Or">
    <Filter Attribute="Название" Value="ООО Альфа" Comparison="Equal">
```

```
    <Filter Attribute="Название" Value="ООО Бета" Comparison="Equal">
  </FilterGroup>
  <Attribute Type="Literal" AttributeId="ДействующаяКомпания" Value=”false”
/>
</UpdateObjectsGroup>
```

4.5.4. Удаление набора объектов

DeleteObjectsGroup – запрос на удаление группы объектов.

Параметры

Запрос имеет два формата — полный и сокращенный, аналогичные форматам запроса GetObjectsGroup.

В сокращенном формате запрос состоит из тега DeleteObjectsGroup, имеющего дополнительный атрибут Code – идентификатор класса объектов. При выполнении запроса краткого формата будут удалены все объекты, относящиеся к указанному классу.

В полном формате тег DeleteObjectsGroup включает в себя несколько вложенных тегов ObjectType, FilterGroup, Item, Geometry, позволяющих задать условие для отбора тех объектов, которые будут изменены. Синтаксис тегов ObjectType, FilterGroup, Item, Geometry приведен в структуре пакета GetObjectsGroup.

Ответ

Пакет **OperationResults** – информация о результате выполнения операции. Также платформа рассылает пакеты SubscriptionDeleteItems со всеми удаленными в результате выполнения запроса объектами. Пакеты SubscriptionDeleteItems получат все информационные системы, заинтересованные в сведениях об объектах, удовлетворяющим условиям запроса.

Примеры запросов

Краткий формат:

```
<?xml version="1.0" encoding="UTF-8"?>
<DeleteObjectsGroup Code="Компании" Endpoint="demo"  Originator="test"
OperationId="0945ab454dc">
</DeleteObjectsGroup>
```

Полный формат:

```
<?xml version="1.0" encoding="UTF-8"?>
<DeleteObjectsGroup Endpoint="demo" Originator="test"
OperationId="0945ab454dcf7" CombineGroups="Or">
  <ObjectType Code="Компании"/>
  <ObjectType Code="Организации"/>
  <FilterGroup Operation="Or">
    <Filter Attribute="Название" Value="ООО Альфа" Comparison="Equal">
    <Filter Attribute="Название" Value="ООО Бета" Comparison="Equal">
  </FilterGroup>
</DeleteObjectsGroup>
```

4.6. Языковые версии данных

4.6.1. Получение списка поддерживаемых языков

GetLanguages – получить перечень языковых версий, поддерживаемых системой. Метод не имеет собственных параметров.

Примеры запросов

В формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetLanguages Endpoint="demo"  Originator="test" />
```

В формате json:

```
{"GetLanguages":{ "Endpoint":"demo", "Originator":"test"}}
```

Ответ

Пакет **LanguagesList** – список поддерживаемых языков.
Вложенные теги *Language* характеризуют конкретную языковую версию и имеет следующие атрибуты:
Code – код языковой версии в стандарте ISO 639.

Name – читаемое наименование языковой версии.

Default – является ли указанный язык языком по умолчанию, возвращаемое значение false или true.

Пример ответа

В формате XML:

```
<LanguagesList Endpoint="demo" Destination="system">
  <Language Code="RU" Name="Русский" Default="true" />
  <Language Code="EN" Name="English" Default="false" />
</LanguagesList>
```

В формате json:

```
{
  "LanguagesList":{
    "Endpoint": "demo", "Destination": "system",
    "Language": [
      {"Code": "RU", "Name": "Русский", "Default": "true"}
      {"Code": "EN", "Name": "English", "Default": "false"}
    ]
  }
}
```

4.6.2.Получение объектов с учетом языковых версий

Настройками MDM один из языков задан как язык по умолчанию. При запросе объектов если языковая версия не указана явно и для объекта есть данные на нескольких языках, то будут возвращены только данные версии языка по умолчанию.

Для запроса данных на других языках служит атрибут **Lang**, указываемый в корневом запросе. В качестве значения атрибута может быть указан код одного из языков, поддерживаемых системой, либо ключевое слово ALL, позволяющее получить по объекту все заданные для него языковые версии.

Примеры запросов

Получить данные по объекту на английском языке, запрос в формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetObject Endpoint="demo" Originator="system" Code="Персона" Lang="EN" />
```

В формате json:

```
{
  "GetObject":{
   "Endpoint":"demo",
    "Originator": "system",
    "Code": "Персона",
    "Lang": "EN"
  }
}
```

Получить полные данные по объекту, запрос в формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetObject Endpoint="demo" Originator="system" Code="Персона" Lang="ALL" />
```

Для объектов указанного класса получить значения атрибутов на английском языке, запрос в формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetObjectsGroup Endpoint="demo" Originator="system" Code="Персона" Lang="EN" />
```

В случае, если свойство объекта имеет значения на разных языках, языковая версия указывается в атрибуте Lang тега Attribute.

Пример ответа в формате XML, содержащего описание объекта, для свойства rdfs:label которого определены значения на двух языках:

```
<Items Endpoint="demo" Destination="system" Lang="ALL" >
  <Item Code="Персона" Name="Персона">
    <Type TypeId="http://www.w3.org/2002/07/owl#Class" Name="Класс"/>
    <Attribute AttributeId="http://www.w3.org/2000/01/rdf-schema#label"
Type="Literal" Value="Персона" Lang="RU"/>
    <Attribute AttributeId="http://www.w3.org/2000/01/rdf-schema#label"
Type="Literal" Value="Person" Lang="EN"/>
  </Item>
</Items>
```

Если язык данных совпадает с языком по умолчанию, атрибут Lang для него может не указываться. Пример ответа в случае, когда русский язык является языком по умолчанию:

```
<Items Endpoint="demo" Destination="system" Lang="ALL" >
  <Item Code="Персона" Name="Персона">
    <Type TypeId="http://www.w3.org/2002/07/owl#Class" Name="Класс"/>
    <Attribute AttributeId="http://www.w3.org/2000/01/rdf-schema#label"
Type="Literal" Value="Персона" />
```

```
    <Attribute AttributeId="http://www.w3.org/2000/01/rdf-schema#label"
Type="Literal" Value="Person" Lang="EN"/>
  </Item>
</Items>
```

Языковые версии применимы только к атрибутам-литералам строкового типа. Атрибуты других типов будут возвращены при любом запросе объекта. Для атрибутов-ссылок читаемое наименование ссылки будет возвращено на языке по умолчанию, если языковая версия в запросе не указана либо выбран вариант Lang=ALL. Если запрошены данные на конкретном языке и значение rdfs:label на этом языке для значения ссылки задано, в атрибуте Name тега Item будет возвращено оно. Если читаемого наименования на запрошенном языке у значения ссылки нет, то в атрибуте Name тега Item будет возвращено наименование на языке по умолчанию.

Пример описания объекта, полученного при запросе без указания атрибута Lang, язык по умолчанию – русский:

```
<Item Code="Персона_1" Name="Тимофеев Т.Т.">
  <Type TypeId="Персона" Name="Персона"/>
  <Attribute AttributeId="http://www.w3.org/2000/01/rdf-schema#label"
Type="Literal" Value="Тимофеев Т.Т."/>
  <Attribute AttributeId="ДатаРождения" Type="Literal" Value="2001-01-02"/>
  <Attribute AttributeId="Сотрудник" Type="Reference" Value="Компания_1" Name="
Иванов и партнеры"/>
</Item>
```

Пример описания объекта, полученного при запросе с заданием Lang=EN:

```
<Item Code="Персона_1" Name="Timofeev T.">
  <Type TypeId="Персона" Name="Person"/>
  <Attribute AttributeId="http://www.w3.org/2000/01/rdf-schema#label"
Type="Literal" Value="Timofeev T." Lang="EN"/>
  <Attribute AttributeId="ДатаРождения" Type="Literal" Value="2001-01-02"/>
  <Attribute AttributeId="Сотрудник" Type="Reference" Value="Компания_1"
Name="Ivanoff Company Inc"/>
</Item>
```

4.6.3. Редактирование языковых версий данных

При редактировании объектов язык, к которому относится указанное значение свойства, указывается в атрибуте Lang тега Attribute. Так же языковая версия может быть указана в атрибуте Lang тега Item, в этом случае она распространяется на все переданные в запросе свойства.

Примеры запросов

Присвоить для объекта читаемое наименование нескольких языках, на запрос в формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<UpdateObject Endpoint="demo" Originator="system"
  <Item Code="Персона" OperationId="1234">
    <Type TypeId="Проект" />
    <Attribute Type="Literal" AttributeId="http://www.w3.org/2000/01/rdf-
schema#label" Value="Персона" Lang="RU" />
    <Attribute Type="Literal" AttributeId="http://www.w3.org/2000/01/rdf-
schema#label" Value="Person" Lang="EN" />
```

```
 </Item>
</UpdateObject>
```

Присвоить для объекта значения нескольких атрибутов на английском языке, на запрос в формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<UpdateObject Endpoint="demo" Originator="system"
  <Item Code="Персона" OperationId="1234" Lang="EN">
    <Type TypeId="Проект" />
    <Attribute Type="Literal" AttributeId="http://www.w3.org/2000/01/rdf-
schema#label" Value="Person" />
    <Attribute Type="Literal" AttributeId="http://www.w3.org/2000/01/rdf-
schema#comment" Value="Class comment" />
 </Item>
</UpdateObject>
```

Языковые версии применимы только к атрибутам-литералам строкового типа. Если язык данных совпадает с языком по умолчанию, то его можно не указывать. Если языковая версия указана для свойства, тип которого не допускает данные на разных языках (числовой, дата и т.п.), то будет возвращена ошибка.

Пример ошибки, возвращаемой в случае попытки присвоить языковую версию свойству типа boolean:

Запрос:

```
<?xml version="1.0" encoding="UTF-8"?>
<UpdateObject Endpoint="demo" Originator="test" OperationId="000004">
  <Item  Code="Персона" OperationId="0000041">
    <Type TypeId="http://www.w3.org/2002/07/owl#Class" />
      <Attribute AttributeId="http://trinidata.ru/archigraph-mdm/archive"
Type="Literal" Value="false" Lang="EN"/>
  </Item>
</UpdateObject>
```

Ответ:

```
<OperationResults Endpoint="demo" Destination="test" OperationId="000004" >
  <OperationResult Result="error" Message="Language versions not allowed for
attribute &#39;http://trinidata.ru/archigraph-mdm/archive&#39;"
OperationId="0000041" Code="Персона"/>
</OperationResults>
```

4.7. Запросы для работы с подписками

4.7.1. Установить подписку

UpdateSubscription – подписать информационную систему на получение информации об изменении объектов заданного класса и/или на изменение модели самого класса.

Параметры

Вложенный тег: *Subscribe*

Атрибуты тега Subscribe:

Format – формат получаемых пакетов с информацией об объектах, значение XML либо JSON.

OperationId – не обязательная строка. Значение, которое будет передаваться в атрибуте OperationId пакетов SubscriptionItems и SubscriptionDeleteItems. Служит для различения пакетов, полученных по разным подпискам.

Delayed – значения 0 (используется по умолчанию) либо 1. Является ли рассылка отложенной. Если значение равно 0, то отправка пакета с измененными объектами будет производиться сразу после внесения изменений. В случае, если нет необходимости в немедленном отклике, можно использовать отложенную отправку, что снижает нагрузку на систему и не приводит к замедлению скорости обработки запросов на стороне АрхиГраф.

Objects – значение 0 (используется по умолчанию) или 1. Получать ли информацию об изменении индивидуальных объектов, принадлежащих классам, указанным в подписке

Model – 0 (используется по умолчанию) или 1. Получать ли информацию об изменении свойств самого класса или его атрибутов.

Active – не обязательный, по умолчанию равен 1 – подписка является активной (значение 1) или не активной (значение 0).

Host, **Port**, **Login**, **Password**, **Queue** – реквизиты очереди для получения пакетов об изменении сущностей. Если ни один из данных атрибутов не передан, но для системы уже есть зарегистрирована активная подписка, то будут использованы реквизиты существующей подписки.

Broker – необязательный атрибут, брокер сообщений. Поддерживаемые значения RabbitMQ и ApacheKafka. Если не указан, то будет использовано значение, заданное настройками MDM как значение брокера по умолчанию.

Тег *ObjectType* – вложен в тег Subscribe.

Перечисляет классы модели, на изменение объектов или свойств которых производится подписка.

Атрибуты тега ObjectType:

Code – идентификатор класса

Пример запроса

Запрос на добавление к существующим подпискам подписки на получение информации об объектах классов Контрагент и Сотрудник:

```
<UpdateSubscription Endpoint="demo" Originator="test">
 <Subscribe Format="json" Delayed="0" Objects="1" >
  <ObjectType Code="Контрагент" />
  <ObjectType Code="Сотрудник" />
</Subscribe>
</UpdateSubscription>
```

Ответ

Пакет **OperationResults** – информация о результате выполнения операции.

Чтобы подписаться на получение информации об изменении всех классов модели, в качестве класса в теге ObjectType указывается служебное слово __root__.

Пример запроса

Запрос на получение информации обо всех изменениях модели:

```
<UpdateSubscription Endpoint="demo" Originator="system">
 <Subscribe Format="json" Delayed="0" Objects="0" Model="1" Host="127.0.0.1"
Port="5672" Login="test" Password="test" Queue="test_subscribe" >
  <ObjectType Code="__root__" />
</Subscribe>
</UpdateSubscription>
```

4.7.2. Получить статус подписки

GetSubscription – для заданных классов получить информацию о том, подписана ли система на получение информации об изменении сущностей и свойства этой подписки. Если классы не заданы, будет возвращена информация обо всех подписках системы.

Параметры

Тег *ObjectType*

Атрибуты тега ObjectType:

Code – идентификатор класса

Пример запроса

В формате XML:

```
<GetSubscription Endpoint="demo" Originator="test">
  <ObjectType Code="Сотрудник" />
</GetSubscription>
```

Ответ

Пакет **Subscribes**

Вложенные теги

Subscribe – характеризуют отдельную подписку. Если для системы нет зарегистрированных подписок для указанного в запросе класса, то тег Subscribes не будет содержать вложенных тегов.

Атрибуты тега Subscribe:

Active – подписка является активной (значение 1) или не активной (значение 0);

Format – формат получаемых пакетов, XML либо JSON;

OperationId – значение для атрибута OperationId пакетов SubscriptionItems и SubscriptionDeleteItems;

Delayed – является ли рассылка отложенной;

Objects - получать ли информацию об изменении индивидуальных объектов;

Model – получать ли информацию об изменении модели класса;

Host, **Port**, **Login**, **Password**, **Queue**, **Broker** – реквизиты очереди для получения пакетов об изменении.

Вложенные теги: *ObjectType*

Атрибуты тега ObjectType:

Code – идентификатор класса
Name – читаемое наименование класса

Пример ответа

В формате XML:

```
<Subscribes Endpoint="demo" Destination="test">
  <Subscribe Format="json" OperationId="" Delayed="0" Objects="1" Model="0"
Host="12.12.12.123" Port="15672" Queue="MDM_IN" >
    <ObjectType Code="Контрагент" Name="Контрагенты" />
    <ObjectType Code="Сотрудник" Name="Сотрудники" />
  <Subscribe>
<Subscribes>
```

Подписки учитывают иерархию классов. Если Класс2 является подклассом для Класс1, то к объектам и свойствам класса Класс2 будут применены все подписки, оформленные для класса Класс1. При запросе состояния подписок класса Класс2 будет возвращено описание фактической подписки.

Пример запроса для подкласса:

```
<GetSubscription Endpoint="demo" Originator="test">
  <ObjectType Code="Класс2" />
</GetSubscription>
```

Ответ при запросе подписки подкласса:

```
<Subscribes Endpoint="demo" Destination="test">
  <Subscribe Format="json" [ … ]>
    <ObjectType Code="Класс1" Name="Класс 1" />
  <Subscribe>
<Subscribes>
```

4.7.3. Отменить подписку

DeleteSubscription – отменить подписку системы на получение информации по указанным классам.

Вложенные теги: *ObjectType*

Атрибуты ObjectType:

Code – обязательный, идентификатор класса

Не обязательные атрибуты:

Format – позволяет удалить только подписки, возвращающие данные в указанном формате JSON или XML. Если не задан, удаляются подписки всех форматов.

Delayed – позволяет удалить только немедленные (если передано значение 0) или отложенные (передано значение 1) подписки. Если не передан, удаляются подписки всех типов.

Objects – если передано значение 1, то будут исключены подписки на получение изменений объектов указанного класса.

Model – если передано значение 1, то будут удалены подписки на получение изменений модели

Если не передано ни Objects=1, ни Model=1, то будут удалены подписки как на получение изменений объектов, так и изменений модели.

Пример запроса

В формате XML:

```
<DeleteSubscription Endpoint="demo" Originator="test">
  <ObjectType Code="Сотрудник" />
</DeleteSubscription>
```

Ответ

Пакет **OperationResults** – информация о результате выполнения операции.

5. API работы с историйностью данных и модели

5.1. Работа с историей модели

5.1.1. Создание виртуальной точки доступа

CreateEndpoint – создать виртуальную точку доступа с состоянием модели на определенную дату.

Параметры

Date – момент времени в прошлом, состояние модели на который необходимо вернуть

Пример запроса

В формате XML:

```
<CreateEndpoint Endpoint="demo" Originator="test" Date="2019-01-01 00:00:00"
/>
```

Пример запроса состояния модели на 01.01.2019:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetDataSchema Endpoint="demo_20190101000000_24234sad34" Destination="test"
/>
```

Ответ

Пакет **OperationResults** – информация о результате выполнения операции. В случае успеха в атрибуте Code будет возвращен идентификатор созданной точки доступа. Этот идентификатор в дальнейшем можно использовать в качестве значения атрибута

Endpoint в запросах на получение модели данных (GetDataSchema, GetDataSchemaCompact).

Пример ответа

```
<OperationResults Endpoint="demo" Destination="test" >
  <OperationResult Result="success" Code="demo_20190101000000_24234sad34" />
</OperationResults>
```

5.1.2. Удаление виртуальной точки доступа

DeleteEndpoint – удалить виртуальную точку доступа.

Параметры

Code – код виртуальной точки доступа

Пример запроса

В формате XML:

```
<DeleteEndpoint Originator="test" Code="demo_20190101000000_24234sad34" />
```

Ответ

Пакет **OperationResults** – информация о результате выполнения операции.

Пример ответа

В формате XML:

```
<OperationResults Destination="test" >
  <OperationResult Result="error" Message="Точка доступа
demo_20190101000000_24234sad34 не найдена"
Code="demo_20190101000000_24234sad34" />
</OperationResults>
```

5.2. Работа с историей данных

5.2.1. Получить историю изменений элемента данных

GetObjectHistory – запрос на получение истории изменения объекта

Параметры

Code – код объекта, информацию о котором необходимо получить.

Пример запроса

В формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetObjectHistory Originator="test" Code="ИвановИИ" />
```

В формате JSON:

```
{"GetObjectHistory": {"Originator":"test","Code":"ИвановИИ"}}
```

Ответ

Пакет **Items** – описание объекта.

Пример ответа

В формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<Items Destination="test">
  <Item Code="ИвановИИ" Name="Иванов И. И.">
   <Type>
    <Operation Date="2019-01-02T16:20:00" System="test"
OperationId="918273645"/>
     <Set Value="Персона"/>
     <Set Value="Сотрудник"/>
    </Operation>
    <Operation Date="2019-01-01T00:00:00" System="test">
     <Set Value="Персона"/>
    </Operation>
   </Type>
   <Attribute Type="Literal" AttributeId="ФИО">
    <Operation Date="2019-01-01T00:00:00" System="test">
     <Set Value=" Иванов И.И."/>
    </Operation>
   </Attribute>
   <Attribute Type="Literal" AttributeId="ДатаРождения">
    <Operation Date="2019-01-01T00:00:00" System="test">
     <Set Value="1979-01-01"/>
    </Operation>
   </Attribute>
   <Attribute Type="Reference" AttributeId="РаботаетВ">
    <Operation Date="2019-01-03T00:00:00" System="test">
     <Set Value="ОООБета"/>
    </Operation>
    <Operation Date="2019-01-02T16:20:00" System="test"
OperationId="918273645">
     <Set Value="ОООАльфа"/>
    </Operation>
   </Attribute>
  </Item>
</Items>
```

Пример ответа в формате JSON:

```
{"Items": {
  "Destination":"test",
  "Item": [
    {
    "Code": "ИвановИИ",
    "Name": "Иванов И.И.",
    "Type": [
          {
              "Operation":[
                 {
                   "Date":"2019-01-02 00:00:00",
                   "System":"test",
                   "Operationid":"918273645",
                   "Set":[{"Value":"Персона"},{"Value":"Сотрудник"}]
                 },
                 {
                   "Date":"2019-01-01 00:00:00",
                   "System":"test",
                   "Set":[{"Value":"Персона"}]
                 }
```

```
                ]
            }
        ],
        "Attribute": [
            {
                "Type":"Literal",
                "AttributeId":"ФИО",
                  "Operation": [
                  {
                        "Date":"2019-01-01 00:00:00",
                        "System":"test",
                        "Set":[{"Value":"Иванов И.И."}]
                    }
                ]
            },
             {
                "Type":"Literal",
                "AttributeId":"ДатаРождения",
                  "Operation": [
                  {
                        "Date":"2019-01-01 00:00:00",
                        "System":"test",
                        "Set":[{"Value":"1979-01-01"}]
                    }
                ]
             },
             {
                "Type":"Reference",
                "AttributeId":"РаботаетВ",
                  "Operation": [
                  {
                        "Date":"2019-01-03T00:00:00",
                        "System":"test",
                        "Set":[{"Value":"ООООбета"}]
                    },
                  {
                        "Date":"2019-01-02T00:00:00",
                        "System":"test",
                        "Operationid":"918273645",
                        "Set":[{"Value":"ОООАльфа"}]
                    }
                ]
             }
        ]
     }
     ]
    }
}
```

5.2.2. Получение состояния элемента данных на определенную дату

Запросы GetObject и GetObjectsGroup могут быть выполнены с передачей дополнительного параметра Date (тип дата и время) для получения состояния объектов на заданную дату. При этом запрос GetObjectsGroup может исполняться только для хранилищ с нативной поддержкой 4D-историйности, а GetObject – для них и для хранилищ, для которых ведется история значений средствами АрхиГраф. В случае, если хранилище объекта не позволяет получить его состояние в прошлом, будет возвращен InvalidPackage с сообщением об ошибке.

Пример запроса

В формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetObject Originator="test" Code="ИвановИИ" Date="2019-01-01 00:00:00" />
```

Ответ

Пакет **Items** – описание объекта. Значение параметра Date будет возвращено в теге Item ответа.

Пример ответа

В формате XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<Items Destination="test">
  <Item Code="ИвановИИ" Name="Иванов И.И." Date="2019-01-01 00:00:00">
   <Type TypeId="Персона" Name="Персоны" />
   <Attribute Type="Literal" AttributeId="ФИО" Value="Иванов И.И." />
   <Attribute Type="Literal" AttributeId="ДатаРождения" Value="1979-01-01"
/>
  </Item>
</Items>
```

5.2.3. Первичная инициализация истории изменения

В случае, если первоначальное заполнение модели и хранилищ данных произошло не через запросы АрхиГраф, для корректной работы с историей изменений требуется первичная инициализация истории. Для этого существуют два служебных метода InitModelHistory и InitObjectsHistory.

InitModelHistory – метод для инициализации истории изменения модели данных. Его единственный и обязательный параметр Date – дата и время, которыми будут помечено первое внесение данных об классах и атрибутах информационной модели.

Пример запроса

В формате XML:

```
<InitModelHistory Endpoint="demo" Originator="system" Date="2019-01-
01T00:00:00" />
```

Пример запроса в формате JSON:

```
{"InitModelHistory": {"Endpoint":"demo", "Originator":"test", "Date":"2019-
01-01T00:00:00"}}
```

В ответ на запрос будет возвращен пакет **OperationResults**.

При повторном вызове метода с той же датой существующие записи о значениях элементов модели за указанную дату будут удалены и заполнены заново.

InitObjectsHistory – метод инициализации истории изменения объектов, принадлежащих указанным классам.

Параметры

Date – дата и время, которыми будет помечена первая история объектов. Обязательный атрибут.

WithoutHistoryOnly – не обязательный, принимает значение 0 и 1, по умолчанию значение 0. Если WithoutHistoryOnly=1, то история будет заполнена только для объектов, для которых она еще не заносилась. Если WithoutHistoryOnly=0 и для объектов уже есть запись за дату, указанную в атрибуте Date, эти записи будут удалены и заполнены заново.

Вложенные теги: ObjectType. Атрибут Code тегов ObjectType содержит код класса, историю для объектов которого требуется инициировать. Задание хотя бы одного тега ObjectType является обязательным.

Пример запроса

В формате XML:

```
<InitObjectsHistory Endpoint="demo" Originator="test" Date="2019-01-
01T00:00:00" WithoutHistoryOnly="1">
  <ObjectType Code="Персона" />
  <ObjectType Code="Компания" />
</InitObjectsHistory>
```

В формате JSON:

```
{"InitObjectsHistory": {
  "Endpoint":"demo",
  "Originator":"test",
  "Date":"2019-01-01T00:00:00",
  "WithoutHistoryOnly":1,
  "ObjectType":[ {"Code":"Персона"}, {"Code":"Компания"} ]
 }}
```

Ответ

Пакет **OperationResults**.

6. Программный интерфейс GraphQL

Интерфейс для тестирования GraphQL-запросов доступен по адресу /graphql в любой инсталляции платформы. Форма тестирования запросов имеет следующий вид:

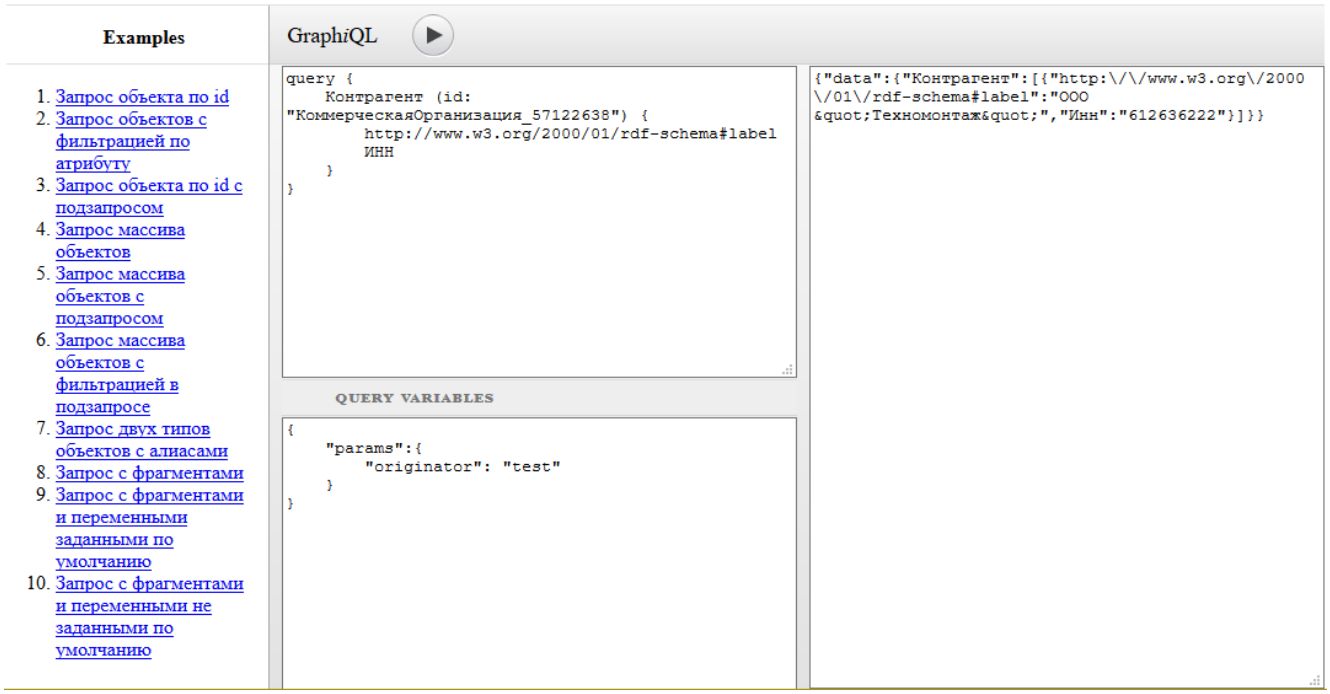

Рис. 3. Форма тестирования GraphQL-запросов

Слева расположены ссылки на примеры запросов, при нажатии на каждую из которых заполняются области запроса и его параметров в средней части страницы. После нажатия кнопки «Выполнить» в правой части страницы отображается ответ платформы.

Выполнить GraphQL-запрос со стороны программных компонентов можно также по адресу /graphql, передав в POST значения параметров query, variables и др. в соответствии с правилами формирования GraphQL POST-запросов.

Значения Endpoint и Originator необходимо передавать в параметрах запроса:

```
{
        "params": {
                "originator": "test",
                "endpoint": "demo"
        }
}
```

6.1. Запросы на чтение (query)

JSON-запрос query состоит из трех полей, из которых query и variables являются обязательными:

```
{query:"...", variables:{...}, operationName:"..."}.
```

В поле **query** содержится строка запроса, а в поле **variables** – используемые в нем переменные. В переменной **params**, необходимо перечислить переменные параметров запроса, поэтому нельзя использовать эту переменную в самом запросе.

GraphQL-ответ приходит тоже в формате JSON, в котором могут быть переданы поля с данными, возвращенными запросом, и ошибками:

```
{data:{...}, errors:[...]}
```

Поле **data** с результатом придет в той же структуре, в которой описан запрос **query**.

Приведем несколько примеров запросов query.

1) Запрос объекта (экземпляра класса Контрагент) по идентификатору – значению Code в АрхиГраф, уникального для каждого объекта, с возвратом значений двух атрибутов: http://www.w3.org/2000/01/rdf-schema#label и ИНН

```
{
  "query":"query {
        Контрагент (id: "КоммерческаяОрганизация_25134855") {
                http://www.w3.org/2000/01/rdf-schema#label
                ИНН
        }
   }",
  "variables": {
      "params": {
        "originator": "test"
}
   }
}
```

Ответ:

```
{
  "data":{
     "Контрагент":[
{
"http:\/\/www.w3.org\/2000\/01\/rdf-schema#label":"ООО
&amp;quot;Техномонтаж&amp;quot;",
"Инн":"612636222"
}
]
  }
}
```

2) Запрос объектов с фильтрацией по атрибуту ИмеетСтатусКонтрагента

```
{
  "query":"query {
    Контрагент (ИмеетСтатусКонтрагента: "СтатусКонтрагента_52370757"){
            http://www.w3.org/2000/01/rdf-schema#label
            ИНН
    }
   }",
  "variables": {
      "params": {
        "originator": "test"
}
   }
}
```

Ответ

```
{"data":{
"Контрагент":[
{
"http:\/\/www.w3.org\/2000\/01\/rdf-schema#label":"ООО
&amp;quot;Техномонтаж&amp;quot;",
"Инн":"612636222"
},
```

```
{
"http:\/\/www.w3.org\/2000\/01\/rdf-schema#label":"ОАО
&quot;Ленэнерго&quot;"
},
{
"http:\/\/www.w3.org\/2000\/01\/rdf-schema#label":"ООО &quot;Машин&quot;"
}
 ]
  }
}
```

3) Запрос объекта по идентификатору с подзапросом: извлекается объект с идентификатором КоммерческаяОрганизация_57122638, и объект, на который он ссылается свойством УчаствуетВПроекте.

```
{
  "query":"query {
            Контрагент (id: "КоммерческаяОрганизация_57122638") {
                    http://www.w3.org/2000/01/rdf-schema#label
                    ИНН
                    УчаствуетВПроекте {
                       http://www.w3.org/2000/01/rdf-schema#label
                        ИмеетСтатус
                    }
            }
  }",
  "variables": {
      "params": {
        "originator": "test"
}
   }
}
```

Ответ

```
{"data":{
"Контрагент":[
{
"http:\/\/www.w3.org\/2000\/01\/rdf-schema#label":"ООО
&quot;Техномонтаж&quot;",
"Инн":"612636222",
"УчаствуетВПроекте":[
{
"http:\/\/www.w3.org\/2000\/01\/rdf-schema#label":"Реконструкция клуба
&quot;Грибоедов&quot;",
"ИмеетСтатус":"ВыполненСОтставаниемСрока"
}
]
}
]
  }
}
```

4) Запрос набора объектов с подзапросом: извлекаются объекты класса Контрагент, а также объекты, связанные с ними свойством УчаствуетВПроекте.

```
{
  "query":"query {
    Контрагент {
            http://www.w3.org/2000/01/rdf-schema#label
            ИНН
            УчаствуетВПроекте {
                http://www.w3.org/2000/01/rdf-schema#label
```

```
                ИмеетСтатус
            }
    }
    }",
  "variables": {
      "params": {
        "originator": "test"
}
   }
}
```

5) Запрос массива объектов с фильтрацией в подзапросе: извлекаются объекты класса Контрагент, участвующие в проектах, имеющих статус с идентификатором ЗамороженИлиОтменен

```
{
  "query":"query {
    Контрагент {
            http://www.w3.org/2000/01/rdf-schema#label
            ИНН
          УчаствуетВПроекте(ИмеетСтатус: "ЗамороженИлиОтменен") {
               http://www.w3.org/2000/01/rdf-schema#label
              ИмеетСтатус
        }
    }
           }",
  "variables": {
      "params": {
        "originator": "test"
}
   }
}
6) Запрос двух типов объектов с алиасами: надежныйКонтрагент и
ненадежныйКонтрагент
{
  "query":"query {
        надежныйКонтрагент: Контрагент(ИмеетСтатусКонтрагента:
"СтатусКонтрагента_5350554") {
             http://www.w3.org/2000/01/rdf-schema#label
        }
        ненадежныйКонтрагент: Контрагент(ИмеетСтатусКонтрагента:
"СтатусКонтрагента_52370757") {
              http://www.w3.org/2000/01/rdf-schema#label
        }
}",
  "variables": {
      "params": {
        "originator": "test"
}
   }
}
7) Запрос с фрагментом - наборПолей
{
  "query":"query {
        надежныйКонтрагент: Контрагент(ИмеетСтатусКонтрагента:
"СтатусКонтрагента_5350554") {
              ...наборПолей
    }
        ненадежныйКонтрагент: Контрагент(ИмеетСтатусКонтрагента:
"СтатусКонтрагента_52370757") {
              ...наборПолей
    }
}
```

```
fragment наборПолей on Контрагент {
       http://www.w3.org/2000/01/rdf-schema#label
       ИНН
       УчаствуетВПроекте {
              http://www.w3.org/2000/01/rdf-schema#label
              ИмеетСтатус
       }
}",
  "variables": {
      "params": {
        "originator": "test"
}
   }
}
8) Запрос с фрагментом и переменными, заданными по умолчанию
{
  "query":"query участвующиеВПроектах($статус: String =
"ЗамороженИлиОтменен", $статусКонтрагента1: String =
"СтатусКонтрагента_5350554") {
           надежныйКонтрагент: Контрагент(ИмеетСтатусКонтрагента:
$статусКонтрагента1) {
                    ...наборПолей
           }
          ненадежныйКонтрагент: Контрагент(ИмеетСтатусКонтрагента:
"СтатусКонтрагента_52370757") {
                    ...наборПолей
           }
}

fragment наборПолей on Контрагент {
           http://www.w3.org/2000/01/rdf-schema#label
           УчаствуетВПроекте(ИмеетСтатус: $статус) {
                    http://www.w3.org/2000/01/rdf-schema#label
           }
}",
  "variables": {
      "params": {
        "originator": "test"
}
   }
}
```

9) Запрос с фрагментами и переменными, не заданными по умолчанию

```
{
  "query":"query участвующиеВПроектах($статус: String, $статусКонтрагента1:
String) {
           надежныйКонтрагент: Контрагент(ИмеетСтатусКонтрагента:
$статусКонтрагента1) {
                    ...наборПолей
           }
           ненадежныйКонтрагент: Контрагент(ИмеетСтатусКонтрагента:
"СтатусКонтрагента_52370757") {
                    ...наборПолей
           }
}

fragment наборПолей on Контрагент {
           http://www.w3.org/2000/01/rdf-schema#label
           УчаствуетВПроекте(ИмеетСтатус: $статус) {
                    http://www.w3.org/2000/01/rdf-schema#label
       }
```

```
    }",
  "variables": {
      "params": {
        "originator": "test"
},
"статус": "ЗамороженИлиОтменен",
"статусКонтрагента1": "СтатусКонтрагента_5350554"
   }
}
```

6.2. Запросы на изменение (mutations)

Изменение данных, хранящихся в платформе, осуществляется с помощью запроса mutation (аналог запроса UpdateObject в основном API платформы).

Логика построения запроса mutation повторяет структуру запроса query.

Ниже приведен пример запроса на добавление/изменение двух объектов класса Персона. В переменных $attributes1 и $attributes2 передаются параметры, с которыми нужно создать/изменить объекты. Если уже существуют объекты с указанными localCode, то объекты будут изменены, иначе созданы (кроме LocalCode, можно передать и постоянный код объекта, присвоенный платформой – Code).

```
{
  "query":"mutation ($attributes1: [Attribute], $attributes2: [Attribute]){
            first:UpdateObject(localCode: "000010002123_3", operationID:
"0000051", typeId: "Персона", attributes: $attributes1){
                    id
                     http://trinidata.ru/archigraph-mdm/LocalCode
            }
            second:UpdateObject(localCode: "000010002123_2", operationID:
"0000052", typeId: "Персона", attributes: $attributes2){
            id
                     http://trinidata.ru/archigraph-mdm/LocalCode
            }
  }",
  "variables": {
      "params":{
                "Endpoint": "demo",
                "originator": "test",
                "OperationID": 55555
        },
        "attributes1":[
        {
                    "attributeId" : "http://www.w3.org/2000/01/rdf-
schema#label",
                    "type" : "Literal",
                    "value": "Тимофеев Т.Т1"
                }
        ],
        "attributes2": [
                {
                    "attributeId" : "http://www.w3.org/2000/01/rdf-
schema#label",
                    "type" : "Literal",
                    "value": "Тимофеев Т.Т2"
                }
        ]
  }
}
```

В ответе в случае успешного выполнения возвращаются запрошенные идентификатор и localCode только что созданных/измененных объектов:


```
{"data":{
"first":[
{
"http:\/\/trinidata.ru\/archigraph-mdm\/LocalCode":"000010002123_3",
"id":"Персона_76aea552c22d05ecbfc78c13fc6d9da9"
}
],
"second":[
{
"http:\/\/trinidata.ru\/archigraph-mdm\/LocalCode":"000010002123_2",
"id":"Персона_f29c5c6559efeb6da6433bcf6231bc66"
}
]
}}
```


7. Точка доступа SPARQL

Платформа АрхиГраф поддерживает чтение данных при помощи стандартного интерфейса SPARQL endpoint. На текущий момент поддерживаются только запросы на чтение данных (SELECT), с некоторыми ограничениями – без использования подзапросов и агрегирующих функций. Интерфейс имеет экспериментальный статус.

Для выполнения запроса нужно обратиться по URL /mdm/[endpoint]/query?originator=[originator]&query=[запрос]. Для тестирования запросов можно использовать панель Fuseki, указав в поле SPARQL endpoint адрес /mdm/[endpoint]/query, например:

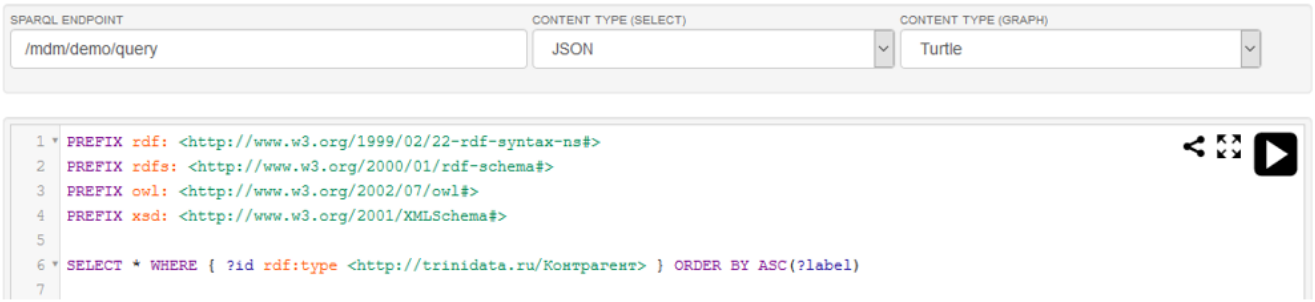

Рис. 4. Тестирование SPARQL endpoint платформы АрхиГраф

Ответ платформа возвращает в соответствии со спецификацией SPARQL, в виде JSON.