ООО «ТриниДата»

ТЕХНИЧЕСКАЯ ИНСТРУКЦИЯ

по работе с системой АрхиГраф.MDM

г. Екатеринбург, 2015

Оглавление

1. Архитектура обмена

Описываемый ниже механизм обмена предназначен, в первую очередь, для обмена основными данными с MDM-системой АрхиГраф.MDM. Аналогичный механизм может использоваться и для обмена информацией напрямую между программными компонентами: в этом случае необходима дополнительная настройка маршрутизации сообщений на уровне шины, которая основывается на типах их содержимого. В любом случае, принцип обмена базируется на использовании не зависящего от структуры данных, моделе-ориентированного формата. Для формирования и разбора такого формата со стороны каждого приложения необходимо анализировать модель данных, описание которой также предоставляется MDM-системой при помощи ее программного интерфейса.

Принцип работы программного интерфейса АрхиГраф.MDM основан на обмене XML-пакетами с другими программными системами. Транспорт пакетов может осуществляться несколькими способами:

- при помощи очередей MQ (предпочтительный способ, и единственный возможный в высоконагруженных и высоконадежных архитектурах),
- SOAP-сервисов, или
- прямых HTTP-запросов.

Форматы пакетов и последовательность обмена ими не зависят от способа передачи информации. В любом случае, запросы и ответы представляют собой XML-пакеты одинакового формата, для доставки которых могут использоваться различные транспорты.

Таким образом, в процессе обмена каждое приложение, желающее отправить запрос к АрхиГраф.MDM, должно сформировать XML-сообщение, и отправить его в MDM при помощи соответствующего транспорта. MDM-система возвратит ответ при помощи того же транспортного протокола.

В рекомендуемой конфигурации, когда обмен сообщениями происходит через MQ, для MDM и каждого приложения создается по две очереди: входящих и исходящих сообщений. В качестве менеджеров очередей могут использоваться как коммерческие, так и OpenSource решения. Мы рекомендуем использовать Apache ActiveMQ. Преимущество данной конфигурации состоит в том, что MDM-система, как и все остальные приложения, обрабатывает сообщения в асинхронном режиме. Благодаря этому не может возникнуть ситуации перегрузки того или иного сервера или приложения. Кроме того, такая схема облегчает кластеризацию MDM-системы.

Маршрутизацию сообщений соответствующим получателям обеспечивает корпоративная сервисная шина (ESB), или брокер сообщений (Message Broker). Эти компоненты отвечают за перемещение сообщений из исходящей очереди отправителя во входящие очереди получателей. Для выбора подходящего маршрута они используют информацию об отправителе, и анализируют содержимое пакета. Можно и напрямую включить в пакет сведения об адресате. В качестве ESB, работающей в связке с АрхиГраф.MDM, могут использоваться решения различных производителей, включая Open Source-продукты.

Архитектура программных компонентов, участвующих в обмене, при реализации описанного принципа будет выглядеть таким образом:

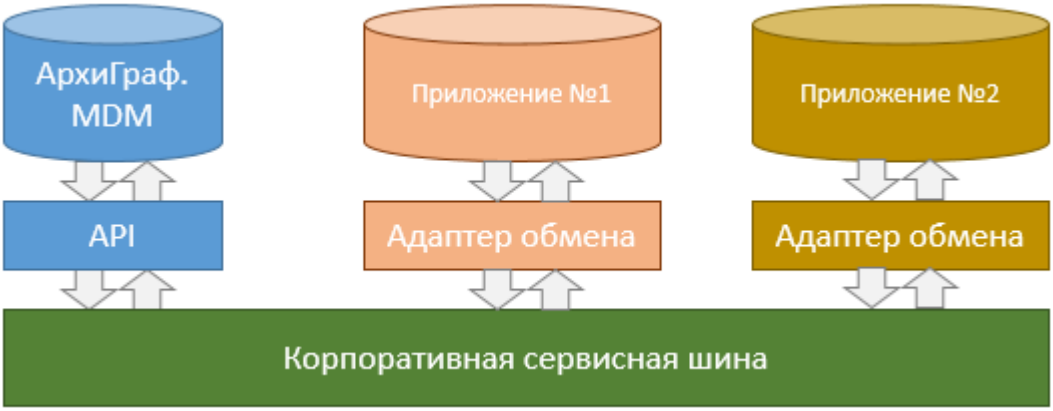

Рис. 1. Архитектура обмена с АрхиГраф.MDM

Последовательность прохождения каждого сообщения через программные компоненты архитектуры обмена выглядит так:

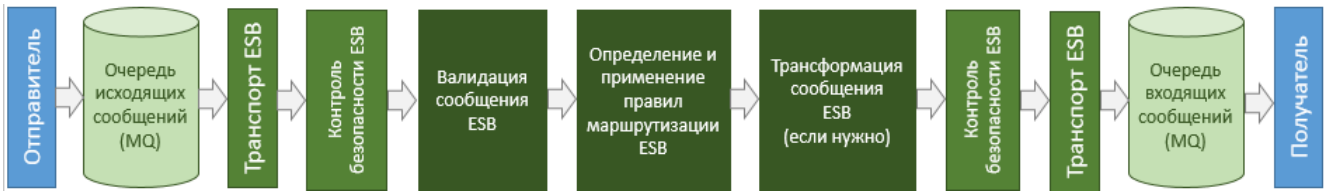

Рис. 2. Последовательность обработки сообщения программными компонентами при передаче между очередями

2. Запросы и ответы АрхиГраф.MDM

На следующей диаграмме показаны виды запросов, обрабатываемых АрхиГраф.MDM, и типы возвращаемых пакетов.

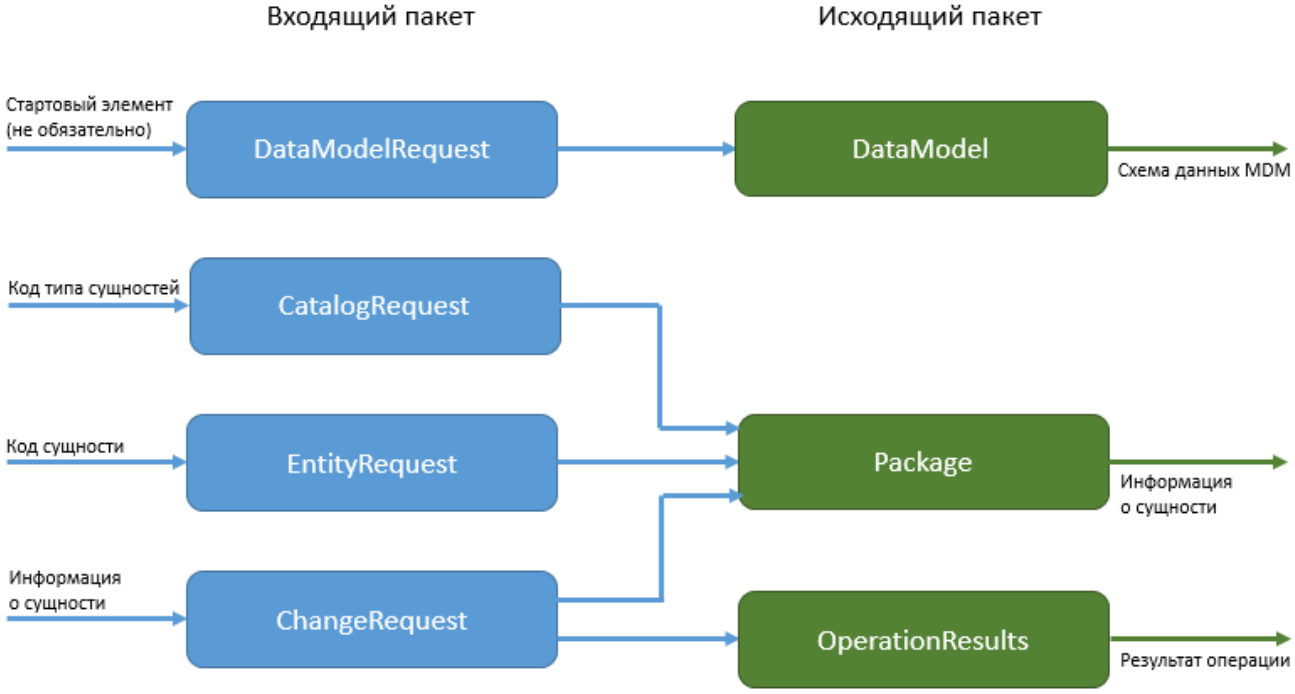

Рис. 3. Диаграмма запросов и ответов АрхиГраф.MDM.

В следующей таблице перечислено назначение каждого вида запросов и ответов.

Табл. 1. Перечень запросов и ответов АрхиГраф.MDM

Запросы	
DataModelRequest	запрос на получение структуры модели
CatalogRequest	запрос на получение всех объектов определенного класса
EntityRequest	запрос на получение конкретного объекта
ChangesRequest	запрос на изменение конкретного объекта
Ответы	
DataModel	структура модели данных
Package	информация о конкретных объектах
OperationResults	результат запроса на выполнение операции
InvalidPackage	сообщение об ошибочном запросе

Далее мы рассмотрим структуру каждого из перечисленных пакетов. Для ее понимания, необходимо знакомство с основными концепциями онтологического моделирования: класс, атрибут, индивидуальный объект (экземпляр). Получить необходимые сведения на эту тему можно в нашем методическом пособии «Введение в семантическое моделирование».

DataModelRequest

запрос на получение структуры информационной модели

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

Ответ: DataModel - пакет с описанием структуры информационной модели: перечнем всех описанных в ней классов, атрибутов и связей.
В случае ошибки при обработке запроса, в ответ на этот и другие пакеты MDM возвращает пакет InvalidPackage.

Пример:

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

CatalogRequest

запрос на получение всех объектов определенного класса

Параметры: Code - наименование класса, список объектов которого необходимо получить.

Ответ: Package - пакет с описанием сущностей указанного класса.

Пример:

```
<?xml version="1.0" encoding="UTF-8"?>
<CatalogRequest Code="Компании" ></CatalogRequest>
```

EntityRequest

запрос на получение конкретного объекта

Параметры: Code - код объекта, описание которого необходимо получить.

Ответ: Package - пакет с описанием сущностей указанного класса.

Пример:

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

ChangesRequest

запрос на изменение конкретного объекта

Параметры: Пакет содержит теги Item, описание которых приводится в структуре пакета Package, 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.

Ответ: OperationResults - пакет с информацией о результатах выполнения операции. Кроме того, после успешного выполнения операции MDM отправляет пакет Package, который получают все информационные системы, заинтересованные в сведениях об объектах такого типа.

Пример:

```
<?xml version="1.0" encoding="UTF-8"?>
<ChangesRequest Originator="ERP" >
[теги Item]
</ChangesRequest>
```

Атрибут Originator содержит идентификатор информационной системы-отправителя, и может использоваться ESB для маршрутизации сообщения, а также для контроля прав доступа на выполнение операции (см. следующий раздел).

Теперь опишем структуру возвращаемых пакетов.

DataModel

структура модели данных

Пакет состоит из множества тегов ObjectType, каждый из которых описывает какой-либо класс информационной модели.

Пример пакета, иллюстрирующий его общую структуру:

```
<?xml version="1.0" encoding="UTF-8"?>
<DataModel StartElement="Клиенты" Prefix="http://some-model.ru/instance">
<ObjectType Code="Компании" Name="Компании">
  <Parent ParentID="ЮридическиеЛица"/>
   <Attribute Type="Literal" AttributeID="Телефон" Name="телефон"
   DataType="xsd:string" MinOccurence="1" MaxOccurence=""/>
</ObjectType>
<ObjectType Code="Персоны" Name="Персоны">
```

```
  <Attribute Type="Literal" AttributeID="ДатаРождения" Name="дата рождения"
  DataType="xsd:date" MinOccurence="1" MaxOccurence="1"/>
  <Attribute Type="Reference" AttributeID="РаботаетВ" Name="работает в">
      <Target TargetId="Компании"/>
  </Attribute>
</ObjectType>
</DataModel>
```

Тег DataModel

Корневой тег модели данных.

Атрибуты:

- 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:

- Code - уникальный код класса
- Name - читаемое наименование класса

Тег Parent

Передает информацию о том, что класс является потомком другого класса. Каждый класс может являться потомком любого числа других классов.

Атрибуты:

- 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:

- 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.
- MinOccurence - минимальное число значений этого атрибута для каждого объекта. Если MinOccurence=1, значит, атрибут обязателен.
- MaxOccurence - максимальное число значений этого атрибута для каждого объекта. Атрибуты MinOccurence и MaxOccurence могут не указываться, тогда атрибут может принимать любое количество значений.

Тег Target

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

Атрибуты:

- TargetID - уникальный код класса

Package

информация о конкретных объектах

Пакет состоит из одного или нескольких элементов Item, каждый из которых передает информацию об одном объекте. В рамках одного пакета Package 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 MDM. 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 MDM 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"?>
<Package>
  <Item Code="ИвановИИ" Name="Иванов И.И." OperationID="0945ab454dcf5">
    <Type TypeId="Персоны"/>
    <Type TypeId="Сотрудники"/>
    <Attribute Type="Literal" AttributeId="ДатаРождения" Value="1970-01-01"/>
    <Attribute Type="Reference" AttributeId="РаботаетВ" Value="ООООльфа"/>
    <Attribute Type="Reference" AttributeId="ИмеетАдрес" Ignore="true"/>
```

```
  </Item>
</Package>
```

Тег Package

Используется для группировки информации о нескольких объектах

Атрибуты:

- Desintation - коды систем-получателей. Может использоваться ESB для маршрутизации сообщений.

Тег Item

Передает информацию о конкретном объекте.

Атрибуты:

- Code - уникальный код объекта
- Name - читаемое имя объекта
- LocalCode - временный код объекта, используемый в случае, когда тег Item передается в составе пакета ChangesRequest, 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 MDM-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 ChangesRequest. Генерируется приложением-отправителем, используется для получения информации о результате операции.

Тег Type

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

Атрибуты:

- TypeId - идентификатор класса, которому принадлежит объект.

Тег Attribute

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

Атрибуты:

- 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 ChangesRequest. 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 (см. подробности в следующем разделе).

- AttributeID - имя атрибута по схеме данных.
- Value - значение атрибута.
- Ignore - используется при передаче тега Item в составе пакета ChangesRequest, и указывает, что MDM-система не должна изменять значение этого атрибута.

OperationResults

результат запроса на выполнение операции

Пакет состоит из одного или нескольких элементов OperationResult, характеризующих результат выполнения операции.

Пример пакета, иллюстрирующий его общую структуру:

```
<?xml version="1.0" encoding="UTF-8"?>
<OperationResults Destination="E-Commerce">
  <OperationResult OperationID="af3459889459" Code="af88811f-deb4-483d-a05f-
  dbd2f21a7bea" LocalCode="2" Result="Успех" Message="Объект успешно создан"/>
  <OperationResult OperationID="564dbaf455" LocalCode="12345" Result="Ошибка"
  Message="Ошибка создания объекта"/>
</OperationResults>
```

Тег OperationResults

Группирует информацию о выполнении пакета операций.

Атрибуты:

- Destination - коды систем-получателей. Может использоваться ESB для маршрутизации сообщений.

Тег OperationResult

Передает информацию о результатах выполнения одной из операций в составе пакета.

Атрибуты:

- OperationID - уникальный идентификатор операции, переданный в теге Item запроса ChangeRequest.
- Code - постоянный код, присвоенный созданному объекту в результате обработки запроса на создание.
- LocalCode - временный код объекта, переданный системой-отправителем. При обработке пакета, система-отправитель должна сохранить у себя постоянный код объекта, присвоенный MDM, используя временный код, чтобы найти объект.
- Result - текстовое сообщение о типе результата операции ("Успех" или "Ошибка").
- Message - развернутое текстовое сообщение о результате операции.

InvalidPackage

сообщение о неверном запросе

Сообщает системе-отправителю, что она отправила неверный запрос.

Пример пакета, иллюстрирующий его общую структуру:

```
<?xml version="1.0" encoding="UTF-8"?>
<InvalidPackage Destination="E-Commerce"/>
```

Тег InvalidPackage

Атрибуты:

- Destination - коды систем-получателей. Может использоваться ESB для маршрутизации сообщений.

3. Авторизация и маршрутизация

АрхиГраф.MDM по умолчанию работает в «доверенном» режиме, в котором не контролируются права доступа к информационной модели, и любое обращающееся приложение может выполнять любые виды запросов. В режиме с контролем безопасности администратор должен настроить перечень программных компонентов, которые могут иметь доступ к АрхиГраф.MDM. После этого он может указать набор классов, к которым каждое из зарегистрированных приложений может обращаться для чтения или изменения. Настройка прав доступа на уровне классов распространяется как на сам класс, так и на все его подклассы, и индивидуальные объекты, являющиеся их членами. Поскольку каждый объект может одновременно являться членом нескольких классов, при определении прав доступа выбирается наиболее строгий вариант. В частности, это позволяет реализовать разные уровни прав доступа к определениям классов (структуре информационной модели), и экземплярам (содержанию модели), назначив более строгие права для класса owl:Class, к которому относятся все определения классов.

В режиме с контролем безопасности все виды запросов, перечисленных в предыдущем разделе, должны быть снабжены информацией об отправителе. Отправитель с таким именем должен быть зарегистрирован в системе АрхиГраф. Атрибут для именования отправителя называется Originator, и должен быть включен в первый, корневой тег пакета. Приведем в качестве примера пакет Package:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package Originator="1C">
  <Item Code="ИвановИИ" Name="Иванов И.И.">
  </Item>
</Package>
```

Кроме того, атрибут Originator должен всегда указываться в запросе ChangesRequest, независимо от режима работы MDM. Проверка того, что значение атрибута Originator соответствует фактическому отправителю, выполняет шина.

Если в режиме с контролем безопасности MDM получит запрос, в котором не указан отправитель – она применит права для анонимных приложений-корреспондентов, и ответит на запрос в соответствии с ними.

Допускается также добавлять в заглавный тег информацию о получателе сообщения, которая должна быть обработана шиной. Для этого используется атрибут Destination, в значении которого указываются идентификаторы систем-получателей – через пробел, если их несколько. Приведем пример пакета Package с указанием систем-получателей:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package Destination="1C E-Commerce">
  <Item Code="ИвановИИ" Name="Иванов И.И.">
  </Item>
</Package>
```

В атрибут Destination сообщений OperationResults и InvalidPackage MDM-система всегда добавляет отправителя сообщения ChangesRequest.

4. Обработка и формирование сообщений со стороны приложений-клиентов

Для обработки и формирования сообщений со стороны приложений-клиентов должны существовать адаптеры информационного обмена, работающие по стандартному алгоритму, который изложен ниже. Имеется стандартная библиотека таких адаптеров для различных платформ; для обсуждения возможности их использования можно обратиться к разработчику MDM-системы, компанию «ТриниДата».

1. В первую очередь, на стороне адаптера должен быть реализован парсер модели данных. Необходимо уложить содержимое пакета DataModel в собственную базу данных. Как минимум, нужно выделить оттуда типы сущностей (классы), и атрибуты, присущие каждому из них.
2. Необходимо реализовать механизм настройки мэппинга (сопоставления) сущностей информационной модели с элементами локального хранилища данных. Такими элементами могут выступать таблицы базы данных и их столбцы, или объекты программной платформы (например, Справочники для 1С), и их атрибуты. В любом случае, интерфейс настройки мэппинга должен обладать следующими возможностями:
   - Сопоставление классов информационной модели таблицам/объектам платформы;
   - Сопоставление атрибутов информационной модели столбцам таблицы/атрибутам объектов платформы;
   - Задание нестандартной процедуры-обработчика на уровне атрибута информационной модели;
   - Задание нестандартной процедуры-обработчика на уровне класса (опционально).

   В частности, для атрибутов, которые являются ссылками на объекты других типов, желательно предусмотреть два варианта хранения связи: в атрибуте объекта (если они связываются один к одному), или в таблице-связке (если объекты связываются один ко многим). При использовании связей один ко многим в информационной модели будет задано, что атрибут, выражающий связь, может иметь множество значений. Подробности реализации этой схемы приведены далее. В любом случае, при настройке связей между объектами, необходимо также указать имена полей, хранящих код MDM и локальный код

в таблице, на которую ссылаются связи (эти имена могут иметь и стандартные значения – например, id и mdm_code.

Настройки мэппинга, которые администратор системы выполняет в этом интерфейсе, сохраняются в локальном хранилище данных приложения, и используются разборе входящих и формировании исходящих сообщений.

3. Блок обработки входящих сообщений Package должен реализовывать следующую логику для каждого элемента Item, содержащегося в пакете (в случае стандартной процедуры обработки, т.е. если не задана процедура специального обработчика):
   - Определить элемент локального хранилища, куда должны сохраняться объекты данного класса. Если объект является членом нескольких классов, вся обработка выполняется для него несколько раз – по одной итерации цикла для каждого класса. Проверить наличие прав доступа на изменение объектов такого типа.
   - Выполнить поиск объекта в хранилище по глобальному идентификатору. Глобальный идентификатор должен содержаться в свойстве, которое одинаково называется для всех элементов локального хранилища – например, mdm_code. Если объект найден, происходит переход к процедуре его обновления, если не найден – к процедуре создания.
   - В случае создания объекта, необходимо добавить в хранилище новую запись, присвоить ей значение mdm_code, и затем перейти к процедуре обновления.
   - Процедура обновления проходит по всем атрибутам объекта, поступившим в составе пакета, и для каждого из них формирует часть запроса на обновление. Эта операция выполняется только для тех атрибутов, для которых найден соответствующий элемент локального хранилища данных – в противном случае, значение атрибута игнорируется. Если в пакете отсутствует значение для какого-либо атрибута, имеющего мэппинг, и у этого атрибута уже есть значение в локальной системе – оно остается без изменений. Очищаются значения только тех атрибутов, для которых пришло пустое значение (при этом в базах данных необходимо присваивать полям записи NULL, если значение атрибута пусто).
   - При сохранении изменений должны отключаться обработчики, регистрирующие журнал изменений для отправки (см. следующий пункт), или журнал должен очищаться после их срабатывания. Это необходимо для исключения отправки эхо-сообщений в адрес MDM.
   - Каждую итерацию цикла, связанную с сохранением какой-либо сущности, необходимо заключать в конструкцию try … catch. При возникновении исключения, система должна записать информацию о нем в свой журнал событий, приложив исходный XML-код пакета, и продолжить обработку.
   - Если при сохранении изменений обнаружена ссылка на другой объект, который еще не известен системе, она должна выполнить следующие действия: определить тип объекта, на который сделана ссылка, по модели данных; создать пустой объект такого типа, и присвоить ему mdm_code, равный значению ссылки; отправить запрос EntityRequest с кодом этого объекта. В результате, через некоторое время система получит информацию об этом объекте, и целостность данных восстановится. Данная возможность предназначена для автоматического восстановления после сбоев обмена.
   - При запуске задания обработки входящих сообщений по расписанию следует принять меры для того, чтобы не возникла ситуация, когда два экземпляра этого задания работают одновременно. Для этого можно ввести флаг блокировки в

локальном хранилище данных – при его обнаружении, вновь запущенный экземпляр задания должен прекратить работу. Флаг блокировки должен автоматически сбрасываться по истечении разумного промежутка времени, чтобы исключить его «зависание» в случае возникновения не обработанного исключения в коде адаптера.

- В общем случае, необходимо обрабатывать ситуацию, когда для одного атрибута приходит несколько значений. Приложению разрешается использовать только первое значение, если его модель данных не позволяет хранить несколько значений для атрибутов, но в этом случае приложение не сможет отправлять запросы на изменение значения атрибутов (см. далее). В то же время, множества значений могут иметь мэппинг на подчиненную таблицу в приложении. В этом случае идентификаторы значений в подчиненной таблице не играют никакой роли. Чаще всего такая ситуация используется на практике для связывания объектов разных типов, т.е. для атрибутов, чьи значения имеют тип Reference. В этом случае приложение может хранить таблицу-связку для каждой пары типов объектов, или – лучше – иметь одну универсальную таблицу, которая описывает связи любых объектов между собой.

4. Блок формирования исходящих сообщений Package должен реализовывать следующую логику:
   - При помощи механизма отслеживания изменений (триггеры на таблицах БД, процедуры-обработчики для платформенных решений) формировать журнал изменений в записях тех контейнеров локального хранилища данных, которые имеют мэппинг с информационной моделью.
   - С определенной периодичностью (раз в несколько минут) выполнять обработку журнала, формируя исходящие пакеты ChangesRequest с информацией об изменениях. Сущности одного типа, или нескольких взаимосвязанных типов – например, клиент и его адреса – могут отправляться одним пакетом. Сущности разных типов желательно отправлять разными пакетами, чтобы упростить диагностику возможных проблем, а также потому, что на типе содержимого могут быть основаны правила маршрутизации.
   - Для формирования пакета выполняется преобразование, обратное описанному в предыдущем пункте. При этом, если передаваемый элемент не имеет присвоенного значения mdm_code – он идентифицируется локальным кодом, значение которого помещается в атрибут LocalCode тега Item. Ссылки на такой элемент возможны только в рамках того же пакета ChangesRequest, и осуществляются путем указания локального кода объекта в значении атрибута тега Attribute. Таким образом, например, пакет для создания двух сущностей – клиента и его адреса – будет выглядеть так:

```
<?xml version="1.0" encoding="UTF-8"?>
<ChangesRequest Originator="1C">
  <Item LocalCode="abc123" Name="ООО \"Альфа\"">
    <Type TypeId="Компания"/>
  </Item>
  <Item LocalCode="def456" Name="Ленина 10-59">
    <Type TypeId="Адрес"/>
    <Attribute Type="LocalCodeReference"
AttributeId="ПринадлежитКомпании" Value="abc123"/>
  </Item>
</Package>
```

Если по какой-то причине элемент, на который имеется ссылка с отправляемой сущности, еще не имеет MDM-кода, и эти сущности нельзя поместить в один пакет – нужно задержать отправку сущности до тех пор, пока код элемента не станет известен. Для этого можно поместить запись об элементе в конец журнала изменений, чтобы она обработалась при следующем запуске процедуры отправки изменений.

Повторная отправка одной и той же сущности, не имеющей MDM-кода – например, если сущность была отредактирована сразу после создания, до получения постоянного кода – является штатной ситуацией, и корректно обрабатывается MDM-системой.

При редактировании тех же сущностей, которым уже присвоены коды MDM, пакет может выглядеть так:

```
<?xml version="1.0" encoding="UTF-8"?>
<ChangesRequest Originator="1C">
  <Item Code="prefix:afebb4394" Name="ООО \"Альфа\"">
    <Type TypeId="Компания"/>
  </Item>
  <Item Code="prefix:fbdc54ad5" Name="Ленина 10-59">
    <Type TypeId="Адрес"/>
    <Attribute Type="Reference" AttributeId="ПринадлежитКомпании"
Value="prefix:afebb4394"/>
  </Item>
</Package>
```

Возможна и смешанная ситуация, когда одна сущность в пакете создается, другая – редактируется.

- В пакете необходимо указать значения всех атрибутов отправляемой сущности, для которых есть мэппинг в данной системе. Если какой-либо атрибут имеет пустое значение, необходимо передать и его. MDM-система при обработке запроса на обновления изменит значения только пришедших атрибутов; то есть, если в MDM определены и другие атрибуты той же сущности, для которых не установлен мэппинг в данном приложении, эти атрибуты останутся неизменными.
- Если какой-либо атрибут имеет множественное значение согласно схеме, а в локальной системе используется только первое из них – приложение не может изменять значения этого атрибута. При отправке нужно поместить в соответствующий тег Attribute флаг Ignore="true". Если же в системе определен способ хранения множественных значений – нужно отправить их все. MDM-система при выполнении операции сначала удалит все имеющиеся значения этого атрибута, а затем создаст новые.
- Операция удаления сущностей выполняется путем присвоения им специального атрибута, имя которого определяется конкретной моделью данных (например, archive). Получив запись с таким атрибутом, система должна удалить ее, или пометить на удаление. Аналогично, при удалении записи в локальной системе, необходимо отправить запрос ChangesRequest, в котором удаленной (или помеченной на удаление) записи присвоен атрибут archive.
- Сформировав пакет, необходимо поместить его во внутреннюю очередь отправки, и после этого удалить обрабатываемую запись из журнала изменений.
- Очередь отправки также должна обрабатываться с определенной периодичностью. Обработка заключается в помещении каждого очередного сообщения в исходящую

очередь, или вызове программного интерфейса MDM другим способом. Эта очередь необходима для того, чтобы исходящие сообщения не были потеряны при временном разрыве связи с MDM.

5. В ответ на каждый пакет ChangesRequest поступает пакет OperationResults. Этот пакет необходимо обработать следующим образом:
   - Если в очередном теге OperationResult не содержится сообщения об ошибке – нужно найти элемент заданного типа по его LocalCode, и сохранить в его свойствах значение постоянного кода – mdm_code.
   - Если поступило сообщение об ошибке, необходимо поместить запись об этом в журнал событий, с указанием кода ошибки, а также вида и кода исходной операции. Желательно сохранить и сам исходный пакет – если адаптер ведет лог исходящих пакетов.
6. При поступлении ответа InvalidPackage, необходимо проинформировать администратора системы.