Санкт-Петербург

Руководство разработчика Luxms BI

Версия 2.8

Август 2018

Оглавление

Глава 1

Протокол обмена данными Luxms BI Sync API

1.1 Термины и определения

1.1.1 VCP

Visual Control Point, см. ТВК

1.1.2 ТВК

Точка Визуального Контроля. Координаты точки в многомерном пространстве измеряемых показателей вместе с её значением. Другими словами, это характеристики измеряемого значения показателя вместе с самим значением. Например, фраза "значение показателя ‘средняя температура‘, измеренное ‘20 декабря 2015 года‘ в ‘Преображенской больнице‘ и равное 36 градусам" описывает одну ТВК. Эта ТВК имеет одно значение, равное 36-и градусам, и три характеристики, как представлено в таблице:

Характеристика	Значение характеристики	Описание
Метрика	Средняя Температура	Метрики отвечают на вопрос `Что?` Каждая метрика имеет единицу измерения (размерность). Например: км/ч, рубли, штуки и т.д.
Место взятия замера	Преображенская больница	Места (объекты контроля) отвечают на вопрос `Где?` Объекты контроля могут иметь географические координаты: широту и долготу, а также границы.
Время взятия замера	20 декабря 2015 года	Временные метки (периоды) отвечают на вопрос `Когда?` Периоды задают временной интервал, в течение которого верно значение ТВК. Периоды имеют дату начала и стандартизованную длительность, например: день, неделя, месяц и т.д.

1.2 Введение

Протокол Luxms BI Sync API разработан для поддержки четырёх типов операций с данными Luxms BI:

- создание (**C**reate)
- получение (**R**etrieve)
- изменение (**U**pdate)
- удаление (**D**elete)

Этот набор операций получил название `CRUD` и поддерживается всеми хранилищами данных в том или ином виде.

Протокол Luxms BI Sync API разработан с использованием элементов архитектурного стиля `REST`. Для обмена данными между клиентом и сервером используется формат `JSON`.

Перед тем как подавать запросы по протоколу Luxms BI Sync API, необходимо пройти аутентификацию в системе.

Для загрузки ТВК в Luxms BI, нужно предварительно подготовить/узнать значения характеристик этих Точек Визуального Контроля.

Внимание! Для корректного копирования примеров из настоящего руководства в консоль, рекомендуется использовать `Acrobat Reader`. Другие программы для чтения PDF файлов не поддерживают формат PDF в полном объёме и работают некорректно при копировании текста.

1.3 Перед началом работы

Предполагается, что примеры будут запускаться в терминале UNIX-подобной системы. Для запуска примеров из этой главы необходимо настроить переменную окружения Shell `luxmsbi_url`.

Например:

```
export luxmsbi_url='https://127.0.0.1:8080'
```

Также, нужно получить у системного администратора логин и пароль для доступа к датасетам на сервере Luxms BI. Если требуется не только читать данные, но и изменять их, то необходимо, чтобы у пользователя был соответствующий уровень доступа к датасету.

1.4 Структуры данных Luxms BI

Luxms BI хранит ТВК (значения показателей) в виде 3-х мерного OLAP куба. Другими словами, каждое значение показателя характеризуется тремя ключами (идентификаторами). Ключи позволяют ответить на вопросы "Что?", "Где?", "Когда?" по отношению к значению показателя. Для хранения дополнительной информации о характеристиках `Что?-Где?-Когда?` используются таблицы-справочники.

Идентификатор	Вопрос	Таблица-справочник	Ключ в таблице значений
Идентификатор метрики	Что?	metrics	metric_id
Идентификатор локации	Где?	locations	loc_id
Идентификатор периода	Когда?	periods	period_id

Упрощённая схема данных представлена на рисунке:

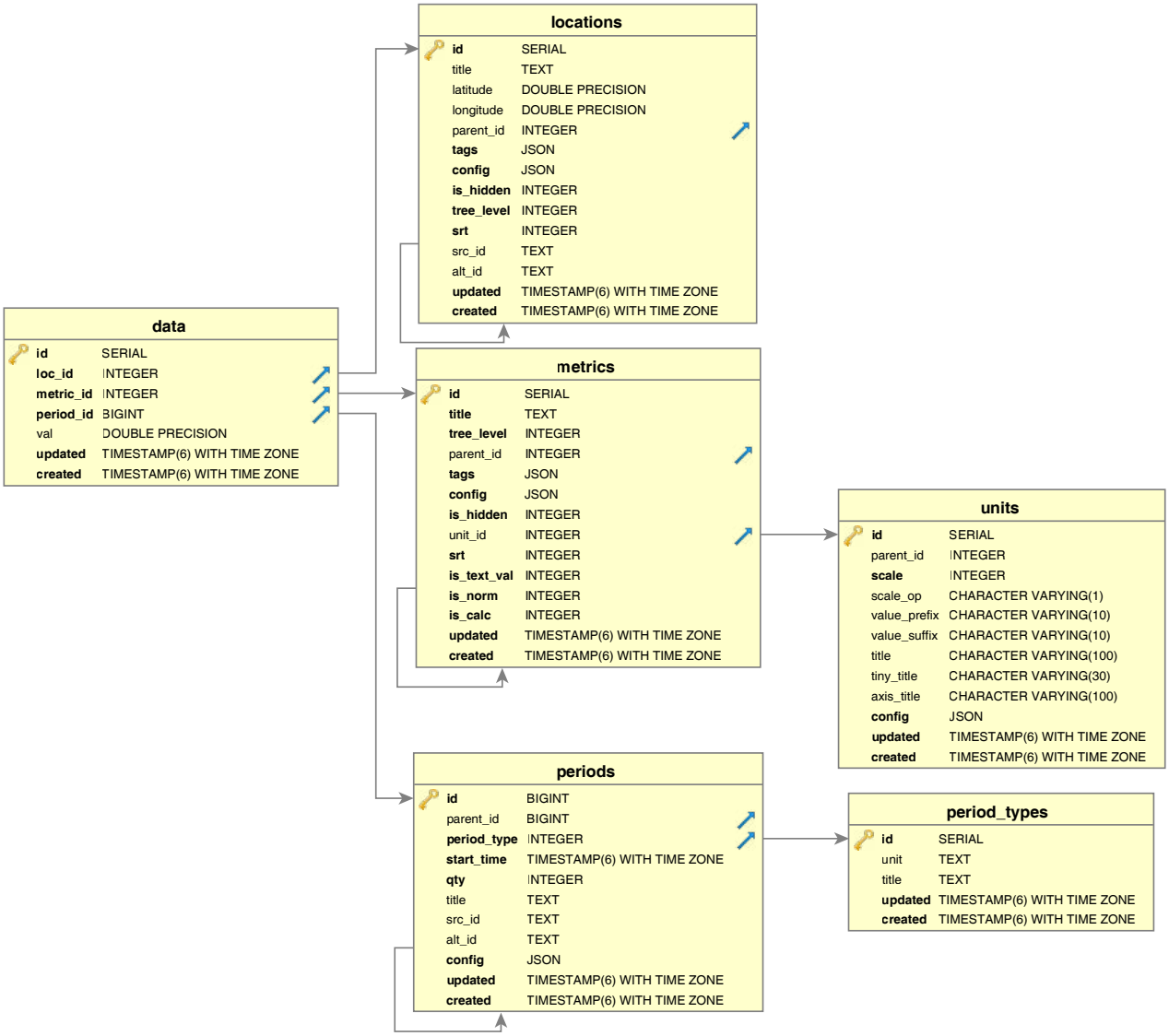

Рис. 1.1 Упрощенная схема данных LuxmsBI

1.4.1 Справочник метрик

Метрики отвечают на вопрос `Что?` по отношению к значению показателя. Другими словами, метрики характеризуют что именно измеряет то или иное значение. Подходящими названиями для метрик будут "Доходы", "Средняя температура", "Совокупная установленная мощность".

Метрики в Luxms BI организованы в иерархию, при этом количество уровней вложенности технически не ограничено.

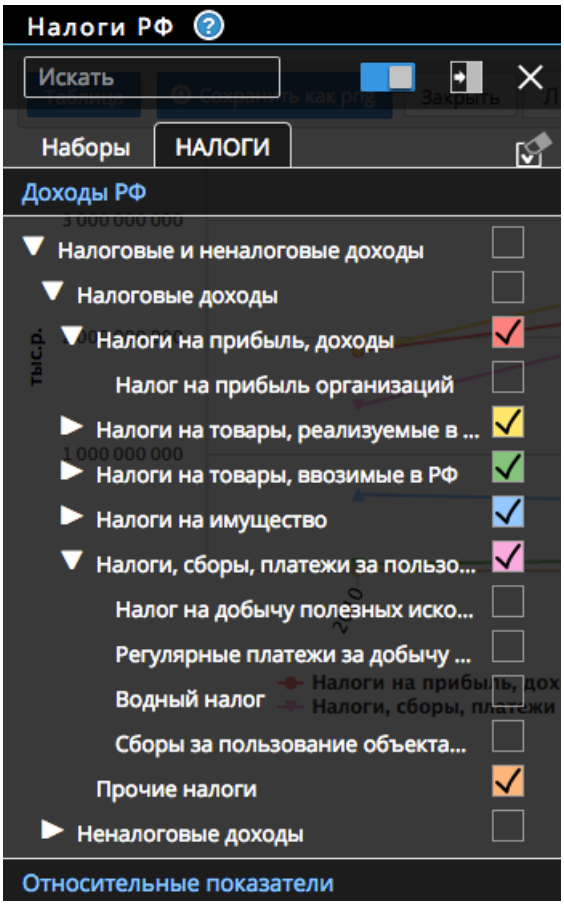

Рис. 1.2 Пример иерархии метрик в Luxms BI

Значения родительской метрики чаще всего являются агрегатами значений своих дочерних метрик. Например, родительская метрика "Доходы" может быть суммой метрик "Процентные доходы", "Операционные доходы" и "Прочие доходы".

Основные поля, характеризующие метрики указаны в таблице:

Имя поля	Значение	Тип поля	Пример JSON
id	Идентификатор метрики	Целое	`{"id": 11}`
title	Название метрики	Строка	`{"title": "Доходы"}`
tree_level	Уровень метрики в дереве. У корня дерева `tree_level=0`	Целое	`{"tree_level": 1}`
parent_id	Ключ родительской метрики. У корня `parent_id=null`	Строка	`{"parent_id": null}`
is_hidden	Признак скрытия метрики в UI	1 или 0	`{"is_hidden": 0}`
unit_id	Ключ единицы измерения из таблицы units	Целое	`{"unit_id": 2}`
srt	Порядковый номер для сортировки в UI	Целое	`{"srt": 10}`

Примечание: Перечень полей может отличаться от описанных в документации и зависит от установлен-

ной версии Luxms BI. Недокументированные поля следует игнорировать во всех запросах Luxms BI Sync API. Изменение значений недокументированных полей может привести к неожиданным побочным эффектам.

1.4.1.1 Справочник единиц измерений

В описании метрик используется поле `unit_id`, которое задаёт единицу измерения для значений. Например, это может быть килограмм, рубль или км/ч. Используемые в датасете единицы измерения хранятся в таблице `units`.

Если поле `unit_id` у метрики установлено в `null`, то пользователь не сможет выбрать эту метрику в панели метрик, а значит, не сможет вывести эту метрику на график.

Основные поля, характеризующие единицы измерения указаны в таблице:

Имя поля	Значение	Тип поля	Пример JSON
id	Идентификатор единицы измерения	Целое	`{"id": 1}`
title	Название единицы измерения для административного интерфейса	Строка	`{"title": "килограммы"}`
value_prefix	Префикс, отображаемый в UI перед значением	Строка	`{"value_prefix": "$"}`
value_suffix	Суффикс, отображаемый в UI после значения	Строка	`{"value_suffix": "%"}`
tiny_title	Название единицы измерения, используемое в легенде	Строка	`{"tiny_title": "шт."}`
axis_title	Название единицы измерения, используемое для подписи осей графиков	Строка	`{"axis_title": "килограммы"}`

1.4.2 Справочник локаций

Локации отвечают на вопрос `Где?` по отношению к значению показателя. Другими словами, локации характеризуют где именно получено то или иное значение. Подходящими названиями для локаций будут "Южный филиал", "Невский район", "г. Москва".

Локации в Luxms BI организованы в иерархию, при этом количество уровней вложенности технически не ограничено.

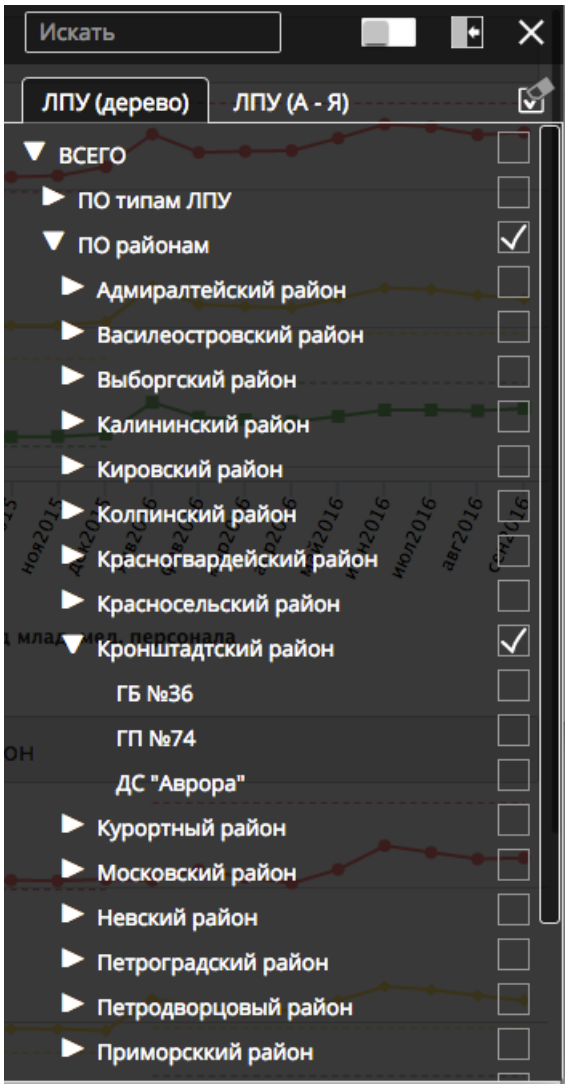

Рис. 1.3 Пример иерархии локаций в Luxms BI

Значения родительской локации чаще всего являются агрегатами значений своих дочерних локаций. Например, родительская локация "г. Санкт-Петербург" может быть суммой значений локаций "Невский район", "Центральный район" и "Кировский район".

Основные поля, характеризующие локации указаны в таблице:

Имя поля	Значение	Тип поля	Пример JSON
id	Идентификатор локации	Целое	`{"id": 11}`
title	Название локации	Строка	`{"title": "офис"}`
tree_level	Уровень локации в дереве. У корня дерева `tree_level=0`	Целое	`{"tree_level": 1}`
parent_id	Ключ родительской локации. У корня `parent_id=null`	Строка	`{"parent_id": null}`
is_hidden	Признак скрытия локации в UI	1 или 0	`{"is_hidden": 0}`
latitude	Долгота в десятичных градусах	Рациональное	`{"latitude": 37.61556}`
longitude	Широта в десятичных градусах	Рациональное	`{"longitude": 55.75222}`

Имя поля	Значение	Тип поля	Пример JSON
srt	Порядковый номер для сортировки в UI	Целое	`{"srt": 10}`

Примечание: Перечень полей может отличаться от описанных в документации и зависит от установленной версии Luxms BI. Недокументированные поля следует игнорировать во всех запросах Luxms BI Sync API. Изменение значений недокументированных полей может привести к неожиданным побочным эффектам.

1.4.3 Справочник периодов

Периоды отвечают на вопрос `Когда?` по отношению к значению показателя. Другими словами, локации характеризуют когда именно получено то или иное значение. Подходящими названиями для периодов будут "Первый квартал 2016 года", "12 часов", "март 2011г.".

Основные поля, характеризующие периоды указаны в таблице:

Имя поля	Значение	Тип поля	Пример
id	Идентификатор периода	64 битное целое	`{"id"↩ : "337719944274378750"}`
title	Название периода	Строка	`{"title": "1кв. 2015"}`
start_time	Дата начала периода	Строка/Дата SQL	`{"start_time" : "2015-12-01"}`
period_type	Тип периода, характеризующий длительность	Целое	`{"period_type":6}`

Периоды характеризуются временем начала периода `start_time` и длительностью `period_type`. В Luxms BI можно использовать 8 стандартных типов периодов, от секунды до года, как описано в разделе Справочник типов периодов.

Внимание! Идентификатор периода представляет собой 64 битное целое число, которое не может быть корректно представлено средствами JavaScript. Протокол Luxms BI Sync API передаёт идентификаторы периодов в виде строки.

1.4.3.1 Справочник типов периодов

Идентификатор типа периода используется для кодирования стандартной длительности периодов в Luxms BI.

Используются следующие типы периодов:

идентификатор типа периода	длительность периода
1	секунды
2	минуты
3	часы
4	дни

идентификатор типа периода	длительность периода
5	недели
6	месяцы
7	кварталы
8	годы

1.5 Аутентификация

Для начала работы с Luxms BI Sync API необходимо подать `POST` запрос на аутентификацию с указанием имени пользователя и пароля.

URL: `/api/auth/login`

Параметры запроса:

Имя параметра	Тип параметра	Пример
username	строка	username=developer
password	строка	password=devpass

Пример запроса:

```
curl -k -v \
--header 'Content-type: application/x-www-form-urlencoded' \
--data 'username=admin&password=abcd' \
-X POST "${luxmsbi_url}/api/auth/login"
```

Внимание! При запуске примеров, скопированных из этого документа, убедитесь, что Вы выполнили шаги, описанные в разделе Перед началом работы.

Пример ответа при успешной аутентификации:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Set-Cookie: LuxmsBI-User-Session=97f948ed-58ae-414c-b6fe-58e8105a29f4; ↩
expires=Thu, 03 Nov 2016 09:55:25 GMT; path=/;

{"key":"ACCESS_GRANTED","type":"MSG","message":"Доступ разрешен."}
```

Для корректной пары логин/пароль возвращается HTTP статус `200` и в заголовке `Set-Cookie` возвращается идентификатор сессии (cookie). Предоставленный cookie необходимо использовать в последующих запросах к сервису Luxms BI Sync API.

Полученный при успешной авторизации `cookie` действителен в течение суток с момента получения.

В теле HTTP ответа передаётся сообщение об успешной аутентификации в формате JSON.

При ошибке авторизации (логин и/или пароль неверны, доступ пользователю запрещен) возвращается HTTP статус `401`.

Пример ответа при ошибке аутентификации:

```
HTTP/1.1 403 Forbidden
Content-Type: application/json; charset=utf-8

{"key":"WRONG_PASSWORD_OR_USER_NAME", "type":"ERR", "message":"Неверный логин
 и/или пароль."}
```

В теле HTTP ответа передаётся сообщение об ошибке аутентификации в формате JSON.

1.6 Авторизация

Для авторизации запроса необходимо передать на сервер полученный на стадии аутентификации cookie. Это можно сделать с помощью HTTP заголовков Cookie или LuxmsBI-User-Session. Если в запросе указаны оба заголовка, то проверяется только заголовок Cookie, а заголовок LuxmsBI-User-Session игнорируется.

Для запуска примеров из этой главы необходимо присвоить переменной окружения Shell luxmsbi_cookie значение cookie, полученное в процессе аутентификации.

Например:

```
export luxmsbi_cookie='4350779b-5a72-4523-a87a-61b59295b18d'
```

1.6.1 Авторизация с помощью заголовка Cookie:

Для авторизации необходимо передать на сервер стандартный заголовок Cookie, как описано в rfc6265.

Пример заголовка:

```
Cookie: 4350779b-5a72-4523-a87a-61b59295b18d
```

1.6.2 Авторизация с помощью заголовка LuxmsBI-User-Session:

Для авторизации необходимо передать на сервер заголовок LuxmsBI-User-Session

Пример заголовка:

```
LuxmsBI-User-Session: 4350779b-5a72-4523-a87a-61b59295b18d
```

1.7 Структура URL

Точкой входа для запросов является URL путь `/api/db/`. Далее следует идентификатор датасета и имя таблицы, с которой будет работать запрос. Символ точки `.` используется в качестве разделителя между идентификатором датасета и именем таблицы.

В Luxms BI Sync API используется два типа URL :

1. `/api/db/${dataset}.${table}`
2. `/api/db/${dataset}.${table}/${object_id}`

сегмент URL	Значения	Пример
`${dataset}`	GUID, имя схемы или id датасета	**`4ebc64b0-39ea-4b62-a28d-722724daaa0a`** или **`ds_180`** или **`12`**
`${table}`	имя таблицы	**`locations`** или **`data`**
`${object_id}`	идентификатор записи в таблице	**`192`** или **`param1`**

Тип операции задаётся методом HTTP запроса в соответствие с таблицей:

HTTP метод	Тип операции	Наличие HTTP Body в запросе	Тип URL
GET	Получение объектов	нет	1 или 2
POST	Создание объектов	да	1
PUT	Редактирование объектов	да	2
DELETE	Удаление объектов	нет	2

1.8 Кодировка данных

Luxms BI Sync API использует формат JSON и кодировку UTF-8 в запросах и ответах.

1.9 Получение информации

1.9.1 Получение идентификаторов объектов из URL

1.9.1.1 Идентификатор набора данных

Набор данных в Luxms BI Sync API идентифицируется тремя способами:

1. целочисленный id датасета, уникальный для одного сервера Luxms BI.
2. текстовое имя схемы, уникальное для одного сервера Luxms BI. В имени схемы используются только латиница, символ подчёркивания и цифры.

3. GUID, сохраняющийся при переносе датасета с одного сервера Luxms BI на другой.

GUID датасета можно получить из URL при навигации в клиентской части Luxms BI. Например, из ссылки http://demo.luxmsbi.com/ui/#dashboard/4ebc64b0-39ea-4b62-a28d-722724daaa0a?metrics=1 можно получить идентификатор датасета 4ebc64b0-39ea-4b62-a28d-722724daaa0a.

1.9.1.2 Идентификатор метрики

Это целочисленный идентификатор, который можно получить из ссылки. Например, из ссылки http://demo.luxmsbi.com/ui/#dashboard/4ebc64b0-39ea-4b62-a28d-722724daaa0a?metrics=11 можно получить идентификатор метрики 11. Идентификаторы метрик кодируются с помощью параметра metrics.

1.9.1.3 Идентификатор локации

Это целочисленный идентификатор, который можно получить из ссылки. Например, из ссылки http://demo.luxmsbi.com/ui/#dashboard/4ebc64b0-39ea-4b62-a28d-722724daaa0a?locations=23,45,11&metrics=11 можно получить идентификаторы локаций 23, 45 и 11. Идентификаторы локаций кодируются с помощью параметра locations.

1.9.1.4 Идентификатор периода

Это целочисленный идентификатор, который можно получить из ссылки. Например, из ссылки http↩
://demo.luxmsbi.com/ui/#dashboard/4ebc64b0-39ea-4b62-a28d-722724daaa0a?period.↩
end=337727177922248702&metrics=11&preset=1 можно получить идентификатор периода 337727177922248702. Идентификаторы периодов кодируются с помощью параметра period.start для начального периода и period.end для конечного периода.

Внимание! Идентификатор периода представляет собой 64 битное целое число, которое не может быть корректно представлено средствами JavaScript.

1.9.2 Получение информации с помощью запроса GET

Данные Luxms BI можно получить с помощью HTTP запроса GET. Для этого нужно указать датасет и имя таблицы, из которой требуется получить данные:

```
/api/db/${dataset}.${table}
```

Например, чтобы получить справочник единиц измерения в датасете с именем схемы ds_tests, нужно использовать URL:

```
/api/db/ds_tests.units
```

В качестве идентификатора датасета можно использовать любой идентификатор, например целочисленный:

```
/api/db/123.units
```

Также можно использовать и GUID датасета:

```
/api/db/5b3a1c32-5a91-4bdc-9036-cdb28768ccf4.units
```

Все эти способы указания датасета равнозначны и возвращают одинаковый результат.

Для авторизации запроса, необходимо указать `cookie`, полученный на стадии аутентификации. Например, это можно сделать, отправив заголовок `LuxmsBI-User-Session`:

```
curl -k -v \
--header "LuxmsBI-User-Session: ${luxmsbi_cookie}" \
-X GET "${luxmsbi_url}/api/db/ds_tests.units"
```

Пример ответа:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[
   {
      "id":2,
      "value_prefix":null,
      "value_suffix":"тыс.руб.",
      "title":"Тыс. рублей",
      "tiny_title":"тыс.руб.",
      "axis_title":"тыс.руб.",
      "updated":"2015-02-25T16:34:06.604562+03:00",
      "created":"2015-04-03T12:36:22.923085+03:00"
   },
   {
      "id":12,
      "value_prefix":null,
      "value_suffix":"тыс.мин.",
      "title":"Тыс. минут",
      "tiny_title":"тыс.мин.",
      "axis_title":"тыс.мин.",
      "updated":"2015-02-25T16:34:06.604562+03:00",
      "created":"2015-04-03T12:36:22.923085+03:00"
   }]

```

В теле ответа можно увидеть список единиц измерения с указанием их идентификаторов.

Используя `GET` запросы можно узнать идентификаторы метрик, локаций и периодов, чтобы в дальнейшем использовать их для вставки значений в Luxms BI.

1.10 Вставка данных в Luxms BI

Для вставки данных в Luxms BI следует использовать HTTP запросы `POST`.

1.10.1 Вставка данных с помощью запроса POST

В URL запроса нужно указать датасет и имя таблицы, в которую будет происходить вставка данных:

```
/api/db/${dataset}.${table}
```

В теле одного HTTP запроса можно указать один и более объектов для добавления в датасет. Для ускорения процесса вставки данных рекомендуется в теле одного HTTP запросе передавать сразу несколько объектов. При этом либо все переданные в одном HTTP запросе объекты будут добавлены, либо ни один из них не будет добавлен из-за ошибки.

Для авторизации запроса, необходимо указать `cookie`, полученный на стадии аутентификации. Например, это можно сделать, отправив заголовок `LuxmsBI-User-Session`:

Для корректной обработки запроса на стороне сервера необходимо указывать тип и кодировку передаваемых данных:

```
Content-type: application/json; charset=utf-8
```

При вставке одного объекта в теле HTTP запроса необходимо использовать JSON нотацию объекта `{}`, при вставке нескольких объектов в теле HTTP запроса используется JSON нотация списка `[]`.

При попытке вставить объект, который нарушает ограничение уникальности ключа (например: совпадение `id`), сервер вернёт HTTP статус 409 и в теле ответа вернёт запись, которая стала причиной конфликта. Такое поведение реализовано только для вставки одиночных объектов в нотации JSON объект `{}`.

1.10.1.1 Добавление одной ТВК

URL: `/api/db/${dataset}.data`

В URL необходимо указать идентификатор датасета. В качестве идентификатора можно использовать GUID, имя схемы или id схемы.

Данные для вставки кодируются в формате JSON, как указано в таблице:

Имя поля	Значение	Тип поля	Пример
metric_id	Идентификатор метрики	строка	`{"metric_id" : 1}`
loc_id	Идентификатор локации	целое число	`{"loc_id":123 }`
period_id	Идентификатор периода	строка (64бит целое)	`{"period_id" : "337727177922248702"}`
val	Значение показателя	число с плавающей точкой	`{"value":123.23}`

Пример запроса:

```
curl -k -v \
--data '{"metric_id": 131, "loc_id":1, "period_id":"337687646120443902",
"val":123.23}' \
--header "LuxmsBI-User-Session: ${luxmsbi_cookie}" \
--header 'Content-type: application/json; charset=utf-8' \
-X POST "${luxmsbi_url}/api/db/ds_tests.data/"
```

Пример тела ответа при успешной вставке:

```
{"id":675803, "loc_id":1, "period_id":337687646120443902, "value":123.23,
"val":123.23, "metric_id":131}
```

Внимание! При добавлении ТВК не рекомендуется указывать `id` записей. Указание конкретных `id` может привести к ошибкам при последующей вставке других ТВК.

1.10.1.2 Добавление нескольких ТВК

Запрос аналогичен вставке одной ТВК, но в теле запроса передаётся массив (список) объектов для вставки.

Пример запроса:

```
curl -k -v \
--data '[
{"metric_id": 131,"loc_id":10, "period_id":"337687646120443902", "val":123.23},
{"metric_id": 131,"loc_id":11, "period_id":"337687646120443902", "val":0.87}]
' \
--header "LuxmsBI-User-Session: ${luxmsbi_cookie}" \
--header 'Content-type: application/json; charset=utf-8' \
-X POST "${luxmsbi_url}/api/db/ds_tests.data"
```

Пример тела ответа при успешной вставке:

```
[
  {"id":675804, "loc_id":10, "period_id":337687646120443902, "val":123.23,
"metric_id":131},
  {"id":675805, "loc_id":11, "period_id":337687646120443902, "val":0.87,
"metric_id":131}]

```

Внимание! При добавлении ТВК не рекомендуется указывать `id` записей. Указание конкретных `id` может привести к ошибкам при последующей вставке других ТВК.

1.10.1.3 Добавление одной метрики с указанием id

URL: `/api/db/${dataset}.metrics`

В URL необходимо указать идентификатор датасета. В качестве идентификатора можно использовать GUID, имя схемы или id схемы.

Данные для вставки кодируются в формате JSON, как указано в таблице:

Имя поля	Значение	Тип поля	Пример JSON
id	Идентификатор метрики	Целое	`{"id": 11}`
title	Название метрики	Строка	`{"title": "Доходы"}`
tree_level	Уровень метрики в дереве. У корня дерева `tree_level=0`	Целое	`{"tree_level": 1}`
parent_id	Ключ родительской метрики. У корня `parent_id=null`	Строка	`{"parent_id": null}`
is_hidden	Признак скрытия метрики в UI	1 или 0	`{"is_hidden": 0}`
unit_id	Ключ единицы измерения из таблицы units	Целое	`{"unit_id": 2}`
srt	Порядковый номер для сортировки в UI	Целое	`{"srt": 10}`

Внимание! Система назначит идентификатор новой метрике автоматически, если в запросе не будет указано значение для ключа `id`.

Пример запроса:

```
curl -k -v \
--data '{"id": 98765, "title": "Средняя температура", "unit_id": 1}' \
--header "LuxmsBI-User-Session: ${luxmsbi_cookie}" \
--header 'Content-type: application/json; charset=utf-8' \
-X POST "${luxmsbi_url}/api/db/ds_tests.metrics"
```

Пример ответа в случае ошибки:

```
HTTP/1.1 409 Conflict
Content-Type: application/json; charset=utf-8

{
  "id":98765,
  "parent_id":null,
  "alt_id":"98765",
  "title":"Минимальная температура",
  "tree_level":0,
  "unit_id":1,
  "srt":2147483647,
  "is_hidden":0
}
```

В данном случае ошибка произошла из-за конфликта уникальности ключа метрики. Метрика с `id=98765` уже существует.

Пример ответа в случае ошибки:

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json; charset=utf-8

{
  "title": "Ошибка SQL webapi.route",
  "message": "отношение \"ds_tests.metrics\" не существует (символ 13)"
}
```

Статус ответа 500 используется при любых внутренних ошибках сервера.

Пример тела ответа в случае успеха:

```
{
  "id":98765,
  "parent_id":null,
  "alt_id":"98765",
  "title":"Средняя температура",
  "tree_level":0,
  "unit_id":1,
  "srt":2147483647,
  "is_hidden":0
}
```

1.10.1.4 Добавление одной метрики без указания id

URL: `/api/db/${dataset}.metrics`

При вставке метрик без указания `id`, Luxms BI присвоит целочисленный `id` автоматически.

Если часть метрик добавляется с явным указанием `id`, а другая честь с использованием автоназначения, это может привести к проблеме неуникальных ключей в таблице `metrics`. Рекомендуется всегда указывать `id` в явном виде.

Имеется возможность указывать альтернативные текстовые идентификаторы `alt_id` для метрик.

Алгоритм автоматической генерации идентификаторов на стороне сервера для разных комбинаций `id` и `alt_id` представлен в таблице:

id в запросе	alt_id в запросе	Результат id	Результат alt_id	Общий результат
NULL	NULL	Следущее значение системного счётчика (autoincrement)	Текстовое представление поля id, полученное из системного счётчика для поля id	id = alt_id = autoincrement

id в запросе	alt_id в запросе	Результат id	Результат alt_id	Общий результат
Целое число I	NULL	I	I	id = alt_id = I
NULL	Целое число I	План А: попытка использовать целое число I в качестве уникального идентификатора. План Б: При нарушении уникальности ключа id, используется следущее значение системного счётчика	I	id = alt_id = I, если реализация плана А прошла без ошибок. Иначе alt_id = I и id = autoincrement
Целое число I	Текст T	I	T	Используются указанные в запросе значения

Пример запроса без указания id и alt_id:

```
curl -k -v \
--data '{"title": "Средняя температура", "unit_id": 1}' \
--header "LuxmsBI-User-Session: ${luxmsbi_cookie}" \
--header 'Content-type: application/json; charset=utf-8' \
-X POST "${luxmsbi_url}/api/db/ds_tests.metrics"
```

Пример тела ответа в случае успеха:

```
{
  "id":132,
  "parent_id":null,
  "alt_id":"132",
  "title":"Средняя температура",
  "tree_level":0,
  "unit_id":1,
  "srt":2147483647,
  "is_hidden":0
}
```

1.10.1.5 Добавление одной локации с указанием id

URL: /api/db/${dataset}.locations

В URL необходимо указать идентификатор датасета. В качестве идентификатора можно использовать GUID, имя схемы или id схемы.

Данные для вставки кодируются в формате JSON, как указано в таблице:

Имя поля	Значение	Тип поля	Пример JSON
id	Идентификатор локации	Целое	`{"id": 11}`
title	Название локации	Строка	`{"title": "офис"}`
tree_level	Уровень локации в дереве. У корня дерева `tree_level=0`	Целое	`{"tree_level": 1}`
parent_id	Ключ родительской локации. У корня `parent_id=null`	Строка	`{"parent_id": null}`
is_hidden	Признак скрытия локации в UI	1 или 0	`{"is_hidden": 0}`
latitude	Долгота в десятичных градусах	Рациональное	`{"latitude": 37.61556}`
longitude	Широта в десятичных градусах	Рациональное	`{"longitude": 55.75222}`
srt	Порядковый номер для сортировки в UI	Целое	`{"srt": 10}`

Внимание! Система назначит идентификатор новой локации автоматически, если в запросе не будет указано значение для ключа `id`.

Пример запроса:

```
curl -k -v \
--data '{"id": 987650, "title": "Головной офис", "latitude": 37.61556, "longitude": 55.75222}' \
--header "LuxmsBI-User-Session: ${luxmsbi_cookie}" \
--header 'Content-type: application/json; charset=utf-8' \
-X POST "${luxmsbi_url}/api/db/ds_tests.locations"
```

Пример ответа в случае ошибки:

```
HTTP/1.1 409 Conflict
Content-Type: application/json; charset=utf-8

{
  "id":987650,
  "title":"Офис Тайга",
  "latitude":37.61556,
  "longitude":55.75222,
  "parent_id":null,
  "tree_level":0,
  "srt":2147483647,
  "is_hidden":0
}
```

В данном случае ошибка произошла из-за конфликта уникальности ключа локации. Локация с `id=987650` уже существует.

Пример ответа в случае ошибки:

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json; charset=utf-8

{
  "title": "Ошибка SQL webapi.route",
  "message": "отношение \"ds_tests.locations\" не существует (символ 13)"
}
```

Статус ответа 500 используется при любых внутренних ошибках сервера.

Пример тела ответа в случае успеха:

```
{
  "id":987650,
  "title":"Головной офис",
  "latitude":37.61556,
  "longitude":55.75222,
  "parent_id":null,
  "tree_level":0,
  "srt":2147483647,
  "is_hidden":0
}
```

1.10.1.6 Добавление одного периода

URL: `/api/db/${dataset}.periods`

В URL необходимо указать идентификатор датасета. В качестве идентификатора можно использовать GUID, имя схемы или id схемы.

Данные для вставки кодируются в формате JSON, как указано в таблице:

Имя поля	Значение	Тип поля	Пример
id	Идентификатор периода	64 битное целое	`{"id"↩ : "337719944274378750"}`
title	Название периода	Строка	`{"title": "1кв. 2015"}`
start_time	Дата начала периода	Строка/Дата SQL	`{"start_time" : "2015-12-01"}`
period_type	Тип периода, характеризующий длительность	Целое	`{"period_type":6}`

Внимание! Алгоритмы визуализации данных используют особые свойства `id` периодов. Для корректной и эффективной работы клиентской части рекомендуется всегда пользоваться автоматической генерацией `id` периодов и никогда не указывать их при вставке данных.

Внимание! Для id периодов по умолчанию используются 64-bit целые числа, которые не могут быть корректно представлены средствами Javascript!

Пример запроса:

```
curl -k -v \
--data '{"title":"12-2010", "start_time": "2010-12-01", "period_type":6}' \
--header "LuxmsBI-User-Session: ${luxmsbi_cookie}" \
--header 'Content-type: application/json; charset=utf-8' \
-X POST "${luxmsbi_url}/api/db/ds_tests.periods"
```

Пример ответа в случае ошибки:

```
HTTP/1.1 409 Conflict
Content-Type: application/json; charset=utf-8

{
   "id": 337485748297662462,
   "period_type": 6,
   "start_time": "2010-12-01T00:00:00+03:00",
   "qty":1,
   "title": "12-2010",
   "updated": "2016-06-29T15:34:38.740998+03:00",
   "created": "2016-06-29T15:34:38.740998+03:00"
}
```

В данном случае ошибка произошла из-за конфликта уникальности периода. Период с указанной датой и типом уже существует и имеет `id` 337485748297662462.

Пример ответа в случае ошибки:

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json; charset=utf-8

{
  "title": "Ошибка SQL webapi.route",
  "message": "отношение \"ds_tests.periods\" не существует (символ 13)"
}
```

Статус ответа 500 используется при любых внутренних ошибках сервера.

Пример тела ответа в случае успеха:

```
{
   "id": 337485748297662462,
   "period_type": 6,
   "start_time": "2010-12-01T00:00:00+03:00",
   "qty":1,
   "title": "12-2015",
   "updated": "2016-06-29T15:34:38.740998+03:00",
   "created": "2016-06-29T15:34:38.740998+03:00"
}
```

1.11 Изменение данных в Luxms BI

Для изменения данных в Luxms BI следует использовать HTTP запросы PUT.

1.11.1 Изменение данных с помощью запроса PUT

1.11.1.1 Изменение значения ТВК с указанием id записи

URL: `/api/db/${dataset}.data/${record_id}`

Для изменения одного определённого значения ТВК нужно знать `id` соответствующей записи в таблице `data`. Этот `id` возвращается при создании записи с помощью запроса `POST`, как описано в разделе Добавление одной ТВК.

`id` записи указывается в URL запроса в сегменте `${record_id}`, а список изменяемых полей и их значений передаётся в теле запроса в формате JSON.

Например, чтобы изменить значение ТВК в датасете `ds_tests` в записи с `id` = `270430` на `19.74`, нужно выполнить такой HTTP запрос:

```
PUT /api/db/ds_tests.data/270430
Content-type: application/json; charset=utf-8

{"val":19.74}
```

Пример для выполнения в командной строке:

```
curl -k -v \
--data '{"val":19.74}' \
--header "LuxmsBI-User-Session: ${luxmsbi_cookie}" \
--header 'Content-type: application/json; charset=utf-8' \
-X PUT "${luxmsbi_url}/api/db/ds_tests.data/270430"
```

В ответе всегда возвращается список записей в состоянии после внесения изменений. В данном случае возвращается список из одного элемента, так как в запросе был указан уникальный `id` записи.

Пример ответа:

```
[
  {
    "id":270430,
    "loc_id":4,
    "period_id":337540174123368446,
    "updated":"2016-06-16T12:37:24.998473+03:00",
    "created":"2015-04-03T12:36:23.491845+03:00",
    "val":19.74,
    "metric_id":17
  }]

```

1.11.1.2 Изменение значения ТВК по условию

URL: `/api/db/${dataset}.data/${LPE}`

Выборку записей для обновления можно производить не только указывая `id` записи, но и с помощью выражений `Lux Path Expressions`, которые позволяют фильтровать записи по условию. Например, можно указать значения ключей `Что?-Где?-Когда?`, которые уникальным образом идентифицируют запись в таблице `data`, и тем самым, выбрать единственную запись для обновления.

Для фильтрации записей используется функция `.filter`. В качестве аргумента функции нужно указать выражение, накладывающее условия на поля записи.

Символ & означает комбинацию условий с помощью операции логического И.

В выражениях не допускается использование пробелов.

Таким образом, условие SQL:

```
WHERE
    loc_id = 4 AND
    period_id = 337540174123368446 AND
    metric_id = 17
```

можно записать с помощью Lux Path Expression:

```
.filter(loc_id=4&period_id=337540174123368446&metric_id=17)
```

Для выполнения запроса, это выражение `LPE` необходимо указать в сегменте URL на месте `${LPE}`:

```
PUT /api/db/ds_tests.data/.filter(loc_id=4&period_id=337540174123368446&metric_id=17)
Content-type: application/json; charset=utf-8

{"val":19.74}
```

Пример для выполнения в командной строке:

```
curl -k -v --globoff \
--data '{"val":20.02}' \
--header "LuxmsBI-User-Session: ${luxmsbi_cookie}" \
--header 'Content-type: application/json; charset=utf-8' \
-X PUT "${luxmsbi_url}/api/db/ds_tests.data/.filter(loc_id=4&period_id=337540174123368446&metric_id=17)"
```

В ответе всегда возвращается список записей в состоянии после внесения изменений. В данном случае возвращается список из одного элемента, так как в запросе был указан уникальный `id` записи.

Пример ответа:

```
[
  {
    "id":270430,
    "loc_id":4,
    "period_id":337540174123368446,
    "updated":"2016-06-16T12:37:24.998473+03:00",
    "created":"2015-04-03T12:36:23.491845+03:00",
    "val":20.02,
    "metric_id":17
  }]

```

1.11.1.3 Изменение метрики с указанием id

URL: `/api/db/${dataset}.metrics/${id}`

Для изменения одной определённой метрики нужно знать её `id`.

`id` метрики указывается в URL запроса в сегменте `${id}`, а список изменяемых полей и их значений передаётся в теле запроса в формате JSON.

Например, чтобы изменить название метрики с `id` = `10` в датасете `ds_tests` на `ЗАМЕТНОЕ название`, нужно выполнить такой HTTP запрос:

```
PUT /api/db/ds_tests.metrics/10
Content-type: application/json; charset=utf-8

{"title": "ЗАМЕТНОЕ название"}
```

Пример для выполнения в командной строке:

```
curl -k -v \
--data '{"title": "ЗАМЕТНОЕ название"}' \
--header "LuxmsBI-User-Session: ${luxmsbi_cookie}" \
--header 'Content-type: application/json; charset=utf-8' \
-X PUT "${luxmsbi_url}/api/db/ds_tests.metrics/10"
```

В ответе всегда возвращается список метрик в состоянии после внесения изменений. В данном случае возвращается список из одного элемента, так как в запросе был указан уникальный `id` метрики.

Пример ответа:

```
[
  {
      "id":10,
      "parent_id":25,
```

```
      "alt_id":"I11",
      "title":"ЗАМЕТНОЕ название",
      "tree_level":2,
      "unit_id":11,
      "srt":90,
      "is_hidden":0,
      "updated":"2016-08-09T19:55:53.239045+03:00",
      "created":"2015-04-03T12:36:23.346958+03:00"
  }]

```

В одном запросе можно поменять сразу несколько полей метрики. Например, для изменения и названия, и порядка сортировки, нужно в теле сообщения передать такой JSON:

```
{"title": "ЗАМЕТНОЕ название", "srt": 1}
```

1.11.1.4 Изменение локации с указанием id

URL: `/api/db/${dataset}.locations/${id}`

Для изменения одной определённой локации нужно знать её `id`.

`id` локации указывается в URL запроса в сегменте `${id}`, а список изменяемых полей и их значений передаётся в теле запроса в формате JSON.

Например, чтобы изменить название локации с `id` = `10` в датасете `ds_tests` на `Головной офис`, нужно выполнить такой HTTP запрос:

```
PUT /api/db/ds_tests.locations/10
Content-type: application/json; charset=utf-8

{"title": "Головной офис"}
```

Пример для выполнения в командной строке:

```
curl -k -v \
--data '{"title": "Головной офис"}' \
--header "LuxmsBI-User-Session: ${luxmsbi_cookie}" \
--header 'Content-type: application/json; charset=utf-8' \
-X PUT "${luxmsbi_url}/api/db/ds_tests.locations/10"
```

В ответе всегда возвращается список локаций в состоянии после внесения изменений. В данном случае возвращается список из одного элемента, так как в запросе был указан уникальный `id` локации.

Пример ответа:

```
[
    {
        "id":10,
        "title":"Головной офис",
        "latitude":57.8136,
        "longitude":28.3496,
        "parent_id":1,
        "level":1,
        "srt":0,
        "is_hidden":0,
        "tree_level":1,
        "updated":"2016-06-16T12:37:24.998473+03:00",
        "created":"2015-04-03T12:36:23.22108+03:00"
  }]

```

В одном запросе можно поменять сразу несколько полей локации. Например, для изменения названия, широты и долготы, нужно в теле сообщения передать такой JSON:

```
{
    "title": "Головной офис",
    "latitude": 57.8,
    "longitude": 28.3
}
```

1.11.1.5 Изменение периода с указанием id

URL: `/api/db/${dataset}.periods/${id}`

Для изменения одного определённого периода нужно знать его `id`.

`id` периода указывается в URL запроса в сегменте `${id}`, а список изменяемых полей и их значений передаётся в теле запроса в формате JSON.

Например, чтобы изменить название периода с `id` = `336937779190038526` в датасете `ds_tests` на `2000 год`, нужно выполнить такой HTTP запрос:

```
PUT /api/db/ds_tests.periods/336937779190038526
Content-type: application/json; charset=utf-8

{"title": "2000 год"}
```

Пример для выполнения в командной строке:

```
curl -k -v \
--data '{"title": "2000 год"}' \
--header "LuxmsBI-User-Session: ${luxmsbi_cookie}" \
--header 'Content-type: application/json; charset=utf-8' \
-X PUT "${luxmsbi_url}/api/db/ds_tests.periods/336937779190038526"
```

В ответе всегда возвращается список периодов в состоянии после внесения изменений. В данном случае возвращается список из одного элемента, так как в запросе был указан уникальный `id` периода.

Пример ответа:

```
[
    {
      "id":336937779190038526,
      "title":"2000 год",
      "period_type":8,
      "start_time":"2000-01-01T00:00:00+03:00",
      "qty":1,
      "updated":"2016-08-09T19:44:10.147325+03:00",
      "created":"2016-08-09T19:41:10.724793+03:00"
  }]

```

В одном запросе можно поменять сразу несколько полей периода. Например, для изменения названия и времени начала периода, нужно в теле сообщения передать такой JSON:

```
{
    "title": "2000 год",
    "start_time":"2000-01-01"
}
```

1.12 Удаление данных из Luxms BI

Для удаления данных из Luxms BI следует использовать HTTP запросы `DELETE`.

1.12.1 Удаление данных с помощью запроса DELETE

1.12.1.1 Удаление ТВК с указанием id записи

URL: `/api/db/${dataset}.data/${record_id}`

Для удаления одной определённой ТВК нужно знать `id` соответствующей записи в таблице `data`. Этот `id` возвращается при создании записи с помощью запроса `POST`, как описано в разделе Добавление одной ТВК.

`id` записи указывается в URL запроса в сегменте `${record_id}`.

Например, чтобы удалить ТВК в датасете `ds_tests`, у которой `id` = `270430`, нужно выполнить такой HTTP запрос:

```
DELETE /api/db/ds_tests.data/270430
```

Пример для выполнения в командной строке:

```
curl -k -v \
--header "LuxmsBI-User-Session: ${luxmsbi_cookie}" \
-X DELETE "${luxmsbi_url}/api/db/ds_tests.data/270430"
```

В ответе всегда возвращается список идентификаторов удалённых записей. В данном случае возвращается список из одного элемента, так как в запросе был указан уникальный `id` записи.

Пример тела ответа в случае успешного удаления записи:

```
[ 270430 ]
```

Пример ответа, в случае ошибки:

```
HTTP/1.1 404 Not Found
Content-Type: application/json; charset=utf-8

{"key":"OBJECT_NOT_FOUND", "type":"ERR", "message":"Невозможно найти указанный объект."}
```

1.12.1.2 Удаление ТВК по условию

URL: `/api/db/${dataset}.data/${LPE}`

Удаление записей можно производить не только указывая `id` записи, но и с помощью выражений `Lux Path Expressions`, которые позволяют фильтровать записи по условию. Например, можно указать значения ключей `Что?-Где?-Когда?`, которые уникальным образом идентифицируют запись в таблице `data`, и тем самым, выбрать единственную запись для удаления.

Для фильтрации записей используется функция `.filter`. В качестве аргумента функции нужно указать выражение, накладывающее условия на поля записи.

Символ & означает комбинацию условий с помощью операции логического И.

В выражениях не допускается использование пробелов.

Таким образом, условие SQL:

```
WHERE
    loc_id = 4 AND
    period_id = 337540174123368446 AND
    metric_id = 17
```

можно записать с помощью Lux Path Expression:

```
.filter(loc_id=4&period_id=337540174123368446&metric_id=17)
```

Для выполнения запроса, это выражение LPE необходимо указать в сегменте URL на месте ${LPE}:

```
DELETE /api/db/ds_tests.data/.filter(loc_id=4&period_id=337540174123368446&
metric_id=17)
```

Пример для выполнения в командной строке:

```
curl -k -v --globoff \
--header "LuxmsBI-User-Session: ${luxmsbi_cookie}" \
-X DELETE "${luxmsbi_url}/api/db/ds_tests.data/.filter(loc_id=4&
period_id=337540174123368446&metric_id=17)"
```

В ответе всегда возвращается список идентификаторов удалённых записей. В данном случае возвращается список из одного элемента, так как в запросе был указан уникальный id записи.

Пример тела ответа в случае успешного удаления записи:

```
[ 270430 ]
```

Пример ответа, в случае ошибки:

```
HTTP/1.1 404 Not Found
Content-Type: application/json; charset=utf-8

{"key":"OBJECT_NOT_FOUND", "type":"ERR", "message":"Невозможно найти указанны
й объект."}
```

1.12.1.3 Удаление метрики с указанием id

URL: /api/db/${dataset}.metrics/${id}

Внимание! При удалении метрики, автоматически удаляются все значения Точек Визуального Контроля, у которых метрика равна удаляемой метрике!

Для удаления одной определённой метрики нужно знать её id.

id метрики указывается в URL запроса в сегменте ${id}.

Например, чтобы удалить метрику с id = 10 в датасете ds_tests, нужно выполнить такой HTTP запрос:

```
DELETE /api/db/ds_tests.metrics/10
```

Пример для выполнения в командной строке:

```
curl -k -v \
--header "LuxmsBI-User-Session: ${luxmsbi_cookie}" \
-X DELETE "${luxmsbi_url}/api/db/ds_tests.metrics/10"
```

В ответе всегда возвращается список идентификаторов удалённых записей. В данном случае возвращается список из одного элемента, так как в запросе был указан уникальный `id` записи.

Пример тела ответа в случае успешного удаления метрики:

```
[ 10 ]
```

Пример ответа, в случае ошибки:

```
HTTP/1.1 404 Not Found
Content-Type: application/json; charset=utf-8

{"key":"OBJECT_NOT_FOUND", "type":"ERR", "message":"Невозможно найти указанный объект."}
```

1.12.1.4 Удаление локации с уазанием id

URL: `/api/db/${dataset}.locations/${id}`

Внимание! При удалении локации, автоматически удаляются все значения Точек Визуального Контроля, у которых локация равна удаляемой локации!

Для удаления одной определённой локации нужно знать её `id`.

`id` локации указывается в URL запроса в сегменте `${id}`.

Например, чтобы удалить локацию с `id = 123` в датасете `ds_tests`, нужно выполнить такой HTTP запрос:

```
DELETE /api/db/ds_tests.locations/123
```

Пример для выполнения в командной строке:

```
curl -k -v \
--header "LuxmsBI-User-Session: ${luxmsbi_cookie}" \
-X DELETE "${luxmsbi_url}/api/db/ds_tests.locations/123"
```

В ответе всегда возвращается список идентификаторов удалённых записей. В данном случае возвращается список из одного элемента, так как в запросе был указан уникальный `id` записи.

Пример тела ответа в случае успешного удаления локации:

```
[ 123 ]
```

Пример ответа, в случае ошибки:

```
HTTP/1.1 404 Not Found
Content-Type: application/json; charset=utf-8

{"key":"OBJECT_NOT_FOUND", "type":"ERR", "message":"Невозможно найти указанный объект."}
```

1.12.1.5 Удаление периода с уазанием id

URL: `/api/db/${dataset}.periods/${id}`

Внимание! При удалении периода, автоматически удаляются все значения Точек Визуального Контроля, у которых период равен удаляемому периоду!

Для удаления одного определённого периода нужно знать его `id`.

`id` периода указывается в URL запроса в сегменте `${id}`.

Например, чтобы удалить период с `id` = `337485748297662462` в датасете `ds_tests`, нужно выполнить такой HTTP запрос:

```
DELETE /api/db/ds_tests.periods/337485748297662462
```

Пример для выполнения в командной строке:

```
curl -k -v \
--header "LuxmsBI-User-Session: ${luxmsbi_cookie}" \
-X DELETE "${luxmsbi_url}/api/db/ds_tests.periods/337485748297662462"
```

В ответе всегда возвращается список идентификаторов удалённых записей. В данном случае возвращается список из одного элемента, так как в запросе был указан уникальный `id` записи.

Пример тела ответа в случае успешного удаления периода:

```
[ 337485748297662462 ]
```

Пример ответа, в случае ошибки:

```
HTTP/1.1 404 Not Found
Content-Type: application/json; charset=utf-8

{"key":"OBJECT_NOT_FOUND", "type":"ERR", "message":"Невозможно найти указанный объект."}
```

Глава 2

Протокол загрузки данных BIPush

2.1 Введение

Проткол BIPush позволяет добавлять, удалять и модифицировать данные в хранилище данных LuxmsBI. Операции с данными кодируются в формате JSON и передаются в виде пакетов.

LuxmsBI гарантирует атомарность выполнения всех операций в одном пакете. То есть либо все операции будут успешно выполнены, либо все операции будут проигнорированы в силу какой-либо ошибки на стороне сервера.

Один пакет может содержать несколько разных операций CRUD и несколько различных структур данных, к которым будут применены операции.

Перед тем как подавать запросы по протоколу BIPush, необходимо пройти аутентификацию в системе.

2.2 URL /api/data/${dataset}/push

В сегменте `dataset` передаётся имя датасета, в который необходимо добавить значения показателей.

сегмент URL	Значения	Пример
`${dataset}`	GUID, имя схемы или id датасета	**`4ebc64b0-39ea-4b62-a28d-722724daaa0a`** или **`ds_180`** или **`12`**

2.3 Структура пакета данных

Каждый пакет состоит из заголовка и тела.

Пример структуры пакета:

```
1 {
```

```
  "version"  : "",
  "packetId" : "",
  "mode"     : "",
  "operations" : []
}
```

2.3.1 Заголовок Пакета

Заголовок пакета состоит из трех полей:

Имя	Тип	Обязатаельность	Значение по умолчанию	Описание
version	string	*		Версия формата данных
packetId	string	*		Уникальный идентификатор пакета
mode	string		sync	Режим выполнения команд

Протокол BIPush поддерживает три режима обработки пакетов

- *sync* - синхронный режим.

HTTP ответ будет выдан после того, как все операции в пакете будут выполнены. Не рекомендуется использовать для передачи большого количества записей (несколько тысяч и более), так как долгое выполнение операций может привести к таймауту HTTP запроса. В этом случае клиент не имеет возможности проверить успешность выполнения операций.

- *async* - асинхронный режим.

HTTP ответ будет выдан сразу после получения пакета. В ответе будет указана ссылка, по которой можно проверять статус выполнения операций.

В текущей версии Luxms BI режим *async* не реализован.

- *queue* - асинхронный режим с постановкой в очередь.

В текущей версии Luxms BI режим *queue* не реализован.

Пример заголовка:

```
"version" : "1.0",
"packetId" : "P1202.11",
"mode" : "sync"
```

2.3.2 Тело Пакета

Телом пакета является список операций.

Пример тела пакета:

```
"operations" : []
```

2.3.2.1 Операция

Операция состоит из четырех полей:

Имя	Тип	Обязатаельность	Значение по умолчанию	Описание
op	string	*		Операция, которую необходимо выполнить над передаваемыми записями.
tablename	string	*		Имя таблицы (сущности), для которой предназначены записи
format	string		raw	Формат записей
records	array	*		Список записей

Протокол BIPush поддерживает четыре типа операций

- *insert* - вставка записей.
- *update* - замена указанных значений в записях.
- *replace* - Для каждой записи будет произведена замена записи на её новую версию, либо запись будет добавлена как новая, если в хранилище LuxmsBI нет записи с указанным id.
- *delete* - удаление записей с указанными id.

Если LuxmsBI обнаружит нарушение целостности данных в процессе выполнения операции, то операция будет прервана и произойдет откат всех предыдущих операций в пакете.

Возможные причины нарушения целостности данных:

- неверный формат даты
- неуникальный ключ записи (id)
- неверная ссылка на родительскую или связанную сущность
- пустые значения для полей, которые обязаны иметь значение

Пример операции:

```
{
  "op" : "replace",
  "tablename" : "norms",
  "format" : "flat",
  "records" : []
}
```

2.4 Нормативы

Нормативы используются для визуального контроля ключевых показателей и задают ограничения на значения показателей.

В Luxms BI поддерживаются нормативы трех типов:

- план-факт
- диапазон значений (минимум и максимум)
- цветовое кодирование диапазонов значений

Плоский формат `"format" : "flat"` позволяет передавать минимальное и/или максимальное значение для показателя и даты начала и окончания действия норматива в "плоском" формате.

2.4.1 Замена значений нормативов

Для передачи нормативов рекомендуется использовать операцию замены `"op" : "replace"` и плоский формат `"format" : "flat"`.

Пример пакета для передачи нормативов:

```
{
"version" : "1.0",
"packetId" : "<Your packet Id>",
"mode" : "sync"
"operations" : [
  {
    "op" : "replace",
    "tablename" : "norms",
    "format" : "flat",
    "records" : [
      {
        "srcId" : 1,
        "title" : "Средняя температура по больнице",
        "srcParamKey" : "P23",
        "minValue" : 20000,
```

```
16          "startDate" : "2015-01-01T00:00:00+03"
17        },
18        {
19          "srcId" : 2,
20          "title" : "Процент доступности кофе-машин",
21          "srcParamKey" : "P1",
22          "minValue" : 95,
23          "startDate" : "2015-03-01T00:00:00+03"
24        }
25      ]
26    }]
27
28  }
```

Запись описывающая операцию состоит из нескольких полей:

Имя	Тип	Обязатаельность	Значение по умолчанию	Описание
srcId	int	*0		Уникальный код норматива во внешней системе
id	int	*0		Уникальный код норматива в LuxmsBI
title	string	*		Название норматива
srcParentId	int			Код родительского норматива во внешней системе
parentId	int			Код родительского норматива вLuxmsBI
srcParamKey	string	*1		Текстовый ключ показателя во внешней системе, для которого действует норматив
srcParamId	int	*1		Числовой код показателя во внешней системе, для которого действует норматив
paramId	int	*1		Числовой код показателя в LuxmsBI, для которого действует норматив
paramKey	int	*1		Текстовый код показателя в LuxmsBI, для которого действует норматив
srcValueId	int	*2		Уникальный код значения норматива во внешней системе
valueId	int	*2		Уникальный код значения норматива в LuxmsBI
minValue	float	*3		Минимальное значение норматива
maxValue	float	*3		Максимальное значение норматива

Имя	Тип	Обязатаельность	Значение по умолчанию	Описание
startDate	date		-бесконечность	Дата начала действия указанных значений норматива
endDate	date		+бесконечность	Дата окончания действия указанных значений норматива
srcLocIdList	array			список id объектов контроля во внешней системе, для которых действует норматив
locIdList	array			список id объектов контроля в LuxmsBI, для которых действует норматив
periodTypeList	array			список типов периодов, для которых действует норматив

Типы периодов задаются как целочисленные идентификаторы

Глава 3

Протокол загрузки данных Telemetry

3.1 Введение

Проткол Telemetry позволяет загружать значения показателей (телеметрию) в хранилище данных LuxmsBI. Значения показателей кодируются в формате CSV и передаются в виде пакетов.

LuxmsBI гарантирует атомарность обработки одного пакета. То есть, либо все данные в пакете будут успешно сохранены, либо все данные в пакете будут проигнорированы в силу какой-либо ошибки на стороне сервера.

Пакет данных передаётся как тело HTTP POST запроса, а дополнительные параметры запроса передаются в виде `QUERY_STRING`.

При обработке пакета, при необходимости, будут созданы и сохранены в таблице `periods` новые периоды. В случае, если значение показателя уже существует в таблице `data`, значение показателя из пакета будет использовано для перезаписи существующего значения.

Перед тем как подавать запросы по протоколу Luxms BI Sync API, необходимо пройти аутентификацию в системе.

3.2 Параметры запроса /api/data/${dataset}/telemetry

В URL передаётся имя датасета, в который необходимо добавить значения показателей.

сегмент URL	Значения	Пример
`${dataset}`	GUID, имя схемы или id датасета	**`4ebc64b0-39ea-4b62-a28d-722724daaa0a`** или **`ds_180`** или **`12`**

Параметры запроса, передаваемые в `QUERY_STRING`:

Имя поля	Значение	Значение по умолчанию	Пример
period_type	Код типа периода. Строка принимающая одно из значений: `second` `minute` `hour` `day` `week` `month` `quarter` `year`		`period_type=second`
timezone	Аббревиатура задающая часовой пояс, в котором необходимо сгенерировать заголовки для временных меток	`Europe/Moscow`	`timezone=Asia/↩ Irkutsk`
date_format	Строка формата, применяемая при генерации заголовка периода. Поддерживаются форматы, используемые функцией PostgreSQL `to_char`	Для каждого типа периода своё значение.*	`date_format=tmmon`
return_new_periods	Флаг, указывающий на необходимость возврата созданных периодов в ответе**	`false`	`return_new_↩ periods=true`

* Значения поля `date_format` по умолчанию для разных типов периодов:

Значение поля `period_type`	Значение поля `date_format` по умолчанию
`second`	`dd.mm.yyyy hh:mi:ss`
`minute`	`dd.mm.yyyy hh:mi`
`hour`	`dd.mm.yyyy hh`
`day`	`dd.mm.yyyy`
`week`	`dd.mm.yyyy`
`month`	`tmmon yyyy`
`quarter`	`Qкв yyyy`
`year`	`yyyy`

** Для получения в ответе информации о добавленных периодах необходимо использовать для параметра `return_new_periods` значение, равное `true` или равное `1`. Все остальные значения, включая пустую строку, равносильны отсутствию параметра `return_new_periods` в URL и не влияют на ответ сервера.

Пример URL:

```
/api/data/ds_test/telemetry?timezone=Asia/Irkutsk&date_format=yyyy-mi-dd&↩
return_new_periods=true&period_type=day
```

3.3 Формат пакета данных

Каждый пакет состоит из нескольких строк в формате CSV, при этом строка-заголовок с названиями столбцов не используется.

Разделителем полей является символ точка с запятой (`;`). Для квотации значений допускается использование символа двойные кавычки (").

В каждой строке передаются значения в порядке следования:

Поле	Тип значения	Пример
ID еденицы измерения метрики. В настоящий момент не интерпретируется сервером.	INT	0
Текстовый идентификатор метрики	TEXT	`Показатели:Натуральные показатели:Количество зданий`
Текстовый идентификатор локации	TEXT	`Санкт-Петербург:Невский район`
Время события	TEXT	`2018-01-01T01:00:00.000Z`
Значение метрики	FLOAT	33023.0

Пример пакета в формате CSV:

```
0;Показатели:Вибрация:Геометрическая частота вибрации;Станок 1516М ↵
#102052:Сенсор 2;2018-01-01T00:00:00.000Z;10.0
0;Показатели:Вибрация:Геометрическая частота вибрации;Станок 1516М ↵
#102052:Сенсор 2;2018-01-01T01:00:00.000Z;20.0
```

Пример ответа при наличии параметра `return_new_periods`:

```
{
  "periods" : {
    "new" : [
      {
        "id":2639600414137342,
        "parent_id":null,
        "tree_level":0,
        "period_type":2,
        "qty":1,
        "start_time":"2018-07-20T00:31:00+08:00",
        "title":"20.07.2018 12:31",
        "tags":[],
        "src_id":null,
        "alt_id":null,
        "tree_path":null,
        "config":{},
        "updated":"2018-08-29T16:25:47.028933+08:00",
        "created":"2018-08-29T16:25:47.028933+08:00"
```

```
      },{
        "id":2639600427408382,
        "parent_id":null,
        "tree_level":0,
        "period_type":2,
        "qty":1,
        "start_time":"2018-07-20T00:58:00+08:00",
        "title":"20.07.2018 12:58",
        "tags":[],
        "src_id":null,
        "alt_id":null,
        "tree_path":null,
        "config":{},
        "updated":"2018-08-29T16:25:47.028933+08:00",
        "created":"2018-08-29T16:25:47.028933+08:00"
      }]
  }
}
```


3.4 Порядок обработки входных данных

3.4.1 Периоды

Для сопоставления отметок времени из входящего пакета данных с теми, которые уже хранятся в хранилище Luxms BI, производится вычисление ID периода. Для этого время переводится в GMT, затем округляется до заданного типа периода (параметр `period_type`), затем вычисляется ID периода. Если в хранилище не существует периода с таким ID, то происходит вставка новой записи в таблицу `periods`.

При вставке нового периода, заголовок периода генерируется путём перевода отметки времени в указанный часовой пояс (параметр `timezone`), последующего округления до заданного типа периода (параметр `period_type`) и, наконец, применения функции форматирования `to_char` с форматом, указанном в параметре `date_format`.

3.4.2 Метрики

Для метрик во входном пакете указывается путь в дереве метрик от корня до требуемой метрики. Путь состоит из заголовков метрик (поле `title` в таблице `metrics`), разделённых символом двоеточия `:`. Использование кавычек для квотации заголовков не допускается. На основании указанных заголовков метрик сервер производит поиск ID для целевой метрики, и использует найденный ID для вставки строки в таблицу `data`.

3.4.3 Локации

Для локаций во входном пакете указывается путь в дереве локаций от корня до требуемой локации. Путь состоит из заголовков локаций (поле `title` в таблице `locations`), разделённых символом двоеточия `:`. Использование кавычек для квотации заголовков не допускается. На основании указанных заголовков локаций сервер производит поиск ID для целевой локации, и использует найденный ID для вставки строки в таблицу `data`.

3.4.4 Еденицы измерения

Еденицы измерения игнорируются, но должны присутствовать во входном пакете.

Глава 4

Luxms BI Importer

4.1 Введение

Компонент импорта **Luxms BI** (далее - импортер) решает задачи по загрузке данных из внешних источников.

Импортер по сути является имплементацией `ETL` стека.

Для настройки источников данных, правил обработки и способа сохранения результатов в импортере используется конфигурационный файл в формате XML.

4.1.1 Термины и определения

4.1.1.1 Источник данных

Источником данных может быть любое хранилище данных (Data Warehouse), способное отдавать данные в виде плоских таблиц. В текущем документе подразумевается использование SQL database как наиболее гибкого варианта.

4.1.1.2 Датасет

В упрощенном виде датасет **Luxms BI** представляет собой трехмерный куб, на осях которого расположены:

- Метрики в иерархическом представлении.
- Объекты контроля в иерархическом представлении.
- Периоды как вектор.

Метрики имеют размерность `unit`. При импорте данных необходимо учитывать, что имеет смысл агрегировать только метрики с одинаковой размерностью.

4.1.1.3 Фрейм данных (Data Frame)

Фрейм данных представляет собой контейнер, в котором перемещаются данные из источника в датасет **Luxms BI** с применением процедур фильтрации, валидации и других преобразований. Фрейм данных привязан к единственному периоду и может содержать любое количество метрик и объектов контроля. Результат выполнения SQL запроса из источника данных помещается во фрейм в виде плоской таблицы. Таблица должна в основном удовлетворять принципу `Tidy Data`.

4.1.1.4 Батч (Batch)

Конфигурация, достаточно полная для описания процедуры загрузки, трансформации и агрегации данных для одного *типа* периода, заданного в узле конфигурации `.../batch/period/@periodType`. Например, данные за час, день, месяц и т. д.

В процессе загрузки данных будут созданы задачи на основе конфигураций батчей.

4.1.1.5 Период

Период является сущностью датасета **Luxms BI**, которая определяет принадлежность данных к шкале даты-времени. В отличие от времени, природа которого вызывает споры в современной науке, период - дискретная величина. Период имеет ТИП и дату/время начала. Тип периода ДОЛЖЕН быть задан в узле конфигурации `.../batch/period/@periodType` Длительность периода вычисляется импортером неявно в зависимости от типа периода. Подстановка `$date`, для использования в SQL запросах заменяется на дату начала периода. Определение даты окончания периода в SQL запросе зависит от БД поставщика данных и остается на усмотрение пользователя системы.

> Импортер не делает никаких предположений о наличии данных за какой-либо период в источнике данных. Для того, чтобы загрузить данные за какой-то интервал дат, нужно явно указать этот интервал в команде импорта.

4.1.2 Установка

Импортер поставляется в виде консольного приложения для Java SE. Для запуска приложения требуется Java 8. Приложение установлено в директории `/opt/luxmsbi/importer`. В данной директории находятся следующие ресурсы.

- *bin* - директория, содержащая скрипты запуска `importer-default` для систем Linux и `importer-default.bat` для Windows
- *lib* - директория с jar библиотеками приложения.
- *config* - файлы конфигурации импортера.
  - *application.properties* - конфигурация приложения
  - *logback.xml* - настройки логирования.
  - *importer-config.xml* - настройки импорта данных.

4.1.3 Параметры конфигурации application.properties

Настройки, сделанные в этом файле, применяются ко всему приложению в целом и не затрагивают настройки загрузок данных.

- *server.port* - HTTP порт процесса. По умолчанию 8192.
- *logging.level.com.luxms.bi* - уровень логирования приложения.
- *lxbi.jdbcUrl* - JDBC URL для базы данных Luxms BI
- *lxbi.jdbcUsername* - пользователь для базы данных Luxms BI
- *lxbi.jdbcPassword* - пароль для базы данных Luxms BI
- *lxbi.pidFile* - путь к pid-файлу процесса. При запуске импортер будет создавать указанный файл и удалять при остановке процесса. Если параметр не указан, файл создаваться не будет.
- *lxbi.configLocation* - путь к файлу конфигурации XML
- *lxbi.executorConcurrency* - количество потоков для одновременной загрузки батчей. Если батчи сконфигурированы таким образом, что они не зависят друг от друга - рекомендуется выставить этот параметр по количеству ядер процессора.
- *lxbi.checkConfig* - boolean. Если параметр указан, будет проверена XML конфигурация на валидность и процесс будет завершен с кодом `0`, если проверка прошла успешно или `1`, если были ошибки. Ошибки можно увидеть в логе.
- *lxbi.runImport* - период дат в формате `yyyy-MM-dd`. Если параметр указан, будет запущена загрузка данных за указанный период дат, после завершения загрузки процесс будет завершен. Если был указан параметр *lxbi.checkConfig* - данный параметр будет проигнорирован.
- *lxbi.disableCache* - boolean. Отключает внутренний кэш импортера. Не рекомендуется для использования на production системе. Может быть полезен в процессе отладки батчей.

Все параметры, указанные в файле `application.properties` можно также применить как аргументы процесса или переменные окружения. При этом аргументы процесса будут иметь наивысший приоритет, затем переменные окружения и наконец параметры, указанные в файле. Существуют соглашения по именованию параметров конфигурации, аргументов процесса и переменных окружения.

4.1.3.1 Пример

```
# Сделанные установки перекроют параметры файла application.properties, остальные параметры будут применены из файла

# через переменную окружения выставляется уровень concurrency
export LXBI_EXECUTOR_CONCURRENCY=16

# через переменную окружения устанавливается путь к XML файлу конфигурации
export LXBI_CONFIG_LOCATION=file:/opt/luxmsbi/importer/config/test-config.xml

# запускается процесс с агрументом командной строки
# Загрузка данных с 2010 (включительно) по 2017 (исключительно) годы
# После завершения загрузки процесс импортера будет завершен в штатном режиме
```

```
/opt/luxmsbi/importer/bin/importer-default --↩
lxbi.runImport=2010-01-01/2017-01-01
```

4.1.4 Настройки логирования logback.xml

Импортер использует подсистему логирования java logback По умолчанию логирование настроено на запись в файл `/var/log/luxmsbi/importer/importer.log` и включена ротация логов. Для отладочных целей можно просто удалить этот файл, тогда логирование будет производиться в stdout процесса. Уровень логирования можно выставить параметром конфигурации `logging.level.com.luxms.bi`.

> Применяйте уровень `logging.level.com.luxms.bi=DEBUG`, чтобы в полной мере логировать выполнение запросов и результатов к источникам данных. Имейте ввиду, что в таком случае при больших объемах данных снижается скорость работы импортера и утилизируется дисковое пространство.

Ниже приводится список команд, которые могут быть полезны при просмотре лога. Путь к файлу для краткости опущен.

- `cat importer.log | grep AddFrame` - выводит задачи, добавленные в очередь выполнения.
- `cat importer.log | grep ExecuteFrame` - выводит задачи, которые были выполнены.
- `cat importer.log | grep ProcessHealth` - выводит информацию о состоянии процесса и задач.

4.1.5 Конфигурация загрузки importer-config.xml

> После редактирования конфигурации XML необходимо перезапустить процесс импортера, если он запущен как сервис

- Список источников данных `sources`.
- Список получателей данных `datasets`.
- Список запросов и правил преобразований исходных данных в пространство датасета **Luxms BI** `batches`
- Список словарей `dictionaries`

4.1.5.1 /importer

Корневой элемент.

Пример:

```
<importer>
    <datasets>...</datasets>
    <sources>...</sources>
    <batches>...</batches>
    <dictionaries>...</dictionaries>
</importer>
```

4.1.5.2 /importer/datasets/dataset

Описание датасета.

- Аттрибут `id (string)`. Идентификатор датасета. **Должен совпадать с именем схемы** датасета и быть уникальным для всех датасетов.

4.1.5.3 /importer/datasets/dataset/locale

Имя локали, которое будет использовано для датасета.

4.1.5.4 /importer/datasets/dataset/timezone

Название time zone датасета.

4.1.5.5 /importer/units/unit

- Аттрибут `id (integer)`. Должно быть уникальным для всех `units`. Будет сохранен в датасете при запуске приложения.
- Аттрибут `title (string)`. Имя unit. Будет использовано в UI.

Пример:

```
<datasets>
    <dataset id="ds_000">
        <locale>ru</locale>
        <timezone>W-SU</timezone>
        <units>
            <unit id="1" title="Граждане"/>
            <unit id="2" title="Штуки"/>
        </units>
    </dataset>
</datasets>
```

4.1.5.6 /importer/sources/source

Описание источника данных.

- Аттрибут `id (string)`. Идентификатор источника данных. Должен быть уникальным для всех `sources`.
- Аттрибут `url (string)`. URL источника данных.
- Аттрибут `username (string)`. Username для соединения с источником данных. Необязательный элемент.
- Аттрибут `password (string)`. Password для соединения с источником данных. Необязательный элемент.

Пример:

```
<sources>
    <source id="default"
            url="jdbc:postgresql://127.0.0.1:5432/db_source"
            username="dbuser"
            password="dbpass" />
</sources>
```

4.1.5.7 /importer/batches/batch

Описание процедуры преобразования исходных даных в набор данных **Luxms BI**.

- Аттрибут `id (string)`. Идентификатор батча. Должен быть уникальным в пространстве `batches`.
- Аттрибут `sourceId (string)`. Ссылка на соответствующий `id` источника данных, из которого будут загружены данные.
- Аттрибут `datasetId (string)`. Ссылка на соответствующий `id` датсета, в который будут загружены данные.
- Аттрибут `onFrameComplete (string)`. Имя файла для выполнения SQL процедур постобработки на датасете **Luxms BI**, после того, как фрейм данных был помещен в датасет. Файл с этим именем должен находиться в директории `config` импортера.

Пример:

```
<batches>
    <batch id="batch1"
           sourceId="default"
           datasetId="ds_000"
           onFrameComplete="postprocess.sql">
    ...
    </batch>
</batches>
```

4.1.5.8 /importer/batches/batch/request

- Содержит единственный текстовый узел CDATA, в котором описывается запрос к источнику данных. В запросе используется подстановка `$date` условия выборки по дате. Для обеспечения совместимости с разными источниками данных, дата подставляется в виде строки. Формат даты конфигурируется в аттрибуте `sourceFormat` узла `period`

Пример:

```
<request><![CDATA[
    SELECT to_date('' || a || '-' || b, 'yyyy-mm') ab, c, d, e, f, g, h
    FROM data_source WHERE to_date('' || a || '-' || b, 'yyyy-mm')='$date']
]></request>
```

4.1.5.9 /importer/batches/batch/period

- Аттрибут `sourceFormat (string)`. Формат даты для подстановки в запрос к источнику данных. Необязательно. По умолчанию будет использовано `yyyy-MM-dd`.
- Аттрибут `titleFormat (string)`. Формат даты для названия периода. Будет использован в UI. Необязательно. По умолчанию будет использовано `yyyy-MM-dd`. Подробнее о форматировании даты - https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html
- Аттрибут `type (string)`. Обязательно. Тип периода, для которого будут загружены данные. Возможные значения: **second, minute, hour, day, week, month, quarter, year**. Остальные значения будут проигнорированы. В зависимости от указанного типа, значение исходной даты будет округлено до ближайшего значащего.

Пример:

```
<period type="month" sourceFormat="yyyy-MM-dd" titleFormat="MMM yyyy"/>
```

4.1.5.10 /importer/batches/batch/location

- Аттрибут `title (string)`. Обязательно. Имя объекта контроля. Будет использован в UI. В этом аттрибуте возможно использование подстановки вида `$colname`, в таком случае значение будет выбрано из соответствующего столбца.
- Аттрибут `dictionary (string)`. Имя словаря для замены значения `title` в результатах выборки. Имеет смысл только при использовании подстановки `$colname` в аттрибуте `title`. Словарь с соответствующим `id` должен быть описан в текущей конфигурации.

Пример:

```
<location title="Дистрибьюторы">
    <location title="$e" dictionary="DictWithDefaultValue" />
</location>
```

4.1.5.11 /importer/batches/batch/metric

- Аттрибут `title (string)`. Обязательно. Имя метрики. Будет использован в UI. В этом аттрибуте возможно использование подстановки вида `$colname`, в таком случае значение будет выбрано из соответствующего столбца.
- Аттрибут `dictionary (string)`. Имя словаря для замены значения `title` в результатах выборки. Имеет смысл только при использовании подстановки `$colname` в аттрибуте `title`. Словарь с соответствующим `id` должен быть описан в текущей конфигурации.
- Аттрибут `value (string)`. Если аттрибут присутствует - значение указанное в аттрибуте будет использовано как значение `double` для текущей метрики. В этом аттрибуте возможно использование макросов агрегации. Аттрибут наследуется всеми descendant метриками. В этом аттрибуте возможно использование макросов агрегации.
- Аттрибут `unitId (integer)`. ID для `unit`, который будет указан для значения узла. Если аттрибут не указан и не был указан для ancestor узлов - то эта метрика не будет selectable в UI Аттрибут наследуется всеми descendant метриками.
- Аттрибут `srcId (string)`. Необязательно, служит для удобства конфигурирования дэшбордов **Luxms BI**

Пример:

```
<metric srcId="Z" title="Название объединяющее все параметры">
    <metric srcId="F" title="Stock Weight Last" value="sum($f)" unitId="2">
        <metric title="$c"></metric>
    </metric>
</metric>
```

Пример 1

Исходная таблица

location1	location2	metric
Region 1	Dist 1	2
Region 2	Dist 2	4
Region 1	Dist 3	8
Region 2	Dist 3	16
Region 2	Dist 4	32

Конфигурация

```
<location title="Дистрибьюторы">
    <location title="$location1">
        <location title="$location2" dictionary="distributors"/>
    </location>
</location>
```

```
...

<dictionary id="distributors" default="Прочие">
    <entry src="Dist 1" dst="АЗС" />
    <entry src="Dist 2" dst="Шаверма" />
</dictionary>
```

Для приведенной выше конфигурации в датасете будет сформировано дерево и значения столбца `metric` аккумулированы в соответствующие узлы дерева в пространстве датасета.

Результат

- Дистрибьюторы
  - Region 1
    * АЗС
    * Прочие
  - Region 2
    * Шаверма
    * Прочие

4.1.5.12 /importer/dictionaries

Список словарей

4.1.5.13 /importer/dictionaries/dictionary

Словарь. Используется для замены значений при загрузке исходных данных.

- Аттрибут `id (string)`. Обязательно.
- Аттрибут `default (string)`. Необязательно. Если не указан - значения, не найденные в словаре, останутся без изменений.

4.1.5.14 /importer/dictionaries/dictionary/entry

- Аттрибут `src (string)`. Обязательно. Значение в исходной таблице.
- Аттрибут `dst (string)`. Обязательно. Значение для замены.

Пример:

```
<dictionaries>
    <dictionary id="DictWithoutDefaultValue">
        <entry src="Key1" dst="Value1" />
        <entry src="Key2" dst="Value2" />
    </dictionary>
    <dictionary id="DictWithDefaultValue" default="Default">
        <entry src="Key1" dst="Value1" />
        <entry src="Key2" dst="Value2" />
    </dictionary>
</dictionaries>
```

Пример структуры таблицы данных

```
CREATE TABLE data_source (
    a integer,  -- year of date
    b integer,  -- month of date
    c character varying,
    d character varying,
    e character varying,
    f double precision,
    g double precision,
    h double precision
);
```

Пример валидного XML документа, описывающего один датасет и процесс трансформации источника

```
<importer>
    <datasets>
        <dataset id="ds_000">
            <locale>ru</locale>
            <timezone>W-SU</timezone>
            <units>
                <unit id="1" title="₽"/>
            </units>
        </dataset>
    </datasets>
    <sources>
        <source id="default" url="jdbc:sqlite:data.sqlite" />
    </sources>
    <batches>

        <batch id="batch1" sourceId="default" datasetId="ds_000">
            <request><![CDATA[ select * from source_data where yyyy='$date' ]]></request>
            <period type="year" sourceFormat="yyyy" titleFormat="yyyy"/>
            <location title="$l1">
                <location title="$l2">
                    <location title="$l3" srcId="$loc_id" />
                </location>
            </location>
            <metric title="Все метрики">
                <metric title="$m1" value="sum($v)" unitId="1">
                    <metric title="$m2">
                        <metric title="$m3">
                            <metric title="$m4">
                                <metric title="$m5"/>
                            </metric>
                        </metric>
```

```
                    </metric>
                </metric>
            </metric>
        </batch>
    </batches>
</importer>
```

4.1.6 Импорт данных

Импорт подразумевает под собой загрузку и преобразование данных из таблиц источника данных в пространство **Luxms BI** по правилам, которые описаны в XML документе `importer-config.xml`. При использовании JDBC результат выполнения SQL запроса из источника данных ДОЛЖЕН возвращать только данные, относящиеся к одному конкретному периоду, например `месяц Январь 2000 года`. Для этого в SQL запросе используется подстановка `$date`, которая в процессе загрузки данных заменяется на дату начала конкретного периода. Подстановка `$date` имеет строковый тип, формат даты в источнике определяется узлом конфигурации `.../batch/period/@sourceFormat` Результат выполнения SQL запроса в дальнейшем будет обработан как фрейм данных. Записи (rows) в результате SQL запроса не должны иметь логических связей и каких-либо ссылок между собой. Каждая запись будет обрабатываться независимо от остальных записей. Результат выполнения SQL запроса может иметь NULL значения, интерпретация NULL значений импортером зависит от настроек батча.

> Для лучшей производительности импортера всё, что возможно, должно быть посчитано на стороне источника данных.

Результат выполнения SQL запроса МОЖЕТ содержать столбцы, которые будут использоваться:

- для группировки метрик.
- как значения метрик.
- для группировки объектов контроля.

4.1.6.1 Пример. Запрос данных SQL за одну дату

```
SELECT * FROM my_table
WHERE date_col=to_date('$date', 'YYYY-MM-DD');
```

4.1.6.2 Пример. Запрос данных SQL за интервал дат

```
SELECT * FROM my_table
WHERE date_col>=to_date('$date', 'YYYY-MM-DD')
AND date_col<(to_date('$date', 'YYYY-MM-DD')+INTERVAL '1 month');
```

Для загрузки данных из внешних источников нужно выполнить HTTP запрос http://localhost:8192/import/<FROM_DATE>/<TO_DATE>.

- <FROM_DATE> - дата в формате yyyy-mm-dd начиная с которой (включительно) будут загружены данные.
- <TO_DATE> - дата в формате yyyy-mm-dd до которой (исключительно) будут загружены данные.

При получении запроса http://localhost:8192/import/<FROM_DATE>/<TO_DATE> импортер добавит в очередь задачи для загрузки данных для всех сконфигурированных батчей и периодов. Батчи будут добавлены в очередь в том порядке, в котором они описаны в файле конфигурации importer-config.xml. Периоды с разными типами (например дни, месяцы) будут добавлены в очередь от меньшего к большему. В течение жизни процесса импортера можно подавать команды импорта настолько часто, насколько необходимо. Новые задачи будут добавляться в конец очереди. Если периоды будут повторяться - старые данные, которые уже были загружены ранее - будут заменены новыми.

Можно настроить ежедневное обновление акуальных данных поставив HTTP запрос в crontab

4.1.7 Трансформация данных

Трансформация - процедура преобразования данных из таблиц источника данных в пространство **Luxms BI**. Трансформация описывается в узле конфигурации /importer/batches/batch. Для трансформации данных может быть применен один из двух алгоритмов. Тип алгоритма задается в узле конфигурации .../batch/@transform

- DescendantsFill - алгоритм агрегирует значение метрики, начиная от узла метрики, для которого указаны значения @value и @unitId и ниже по дереву для всех descendants узлов.
- GroupingSets - при таком алгоритме результат SQL запроса будет разложен по деревьям метрик и объектов контроля без использования агрегации.

Этот алгоритм должен применяться, если все значения в результате SQL запроса уже были сгруппированы с использованием SQL группировок, например GROUPING SETS или CUBE

4.1.8 Аггрегация данных

Импортер производит агрегирующие операции над фреймом и затем перемещает его в датасет **Luxms BI**. Формулы агрегации задаются в узлах .../metric/@value и представлют собой математические выражения в расширенной нотации. В формуле агрегации возможно применение функций sum, count, avg, round, if, а также общепринятых арифметических операций. Для доступа к значению, хранящемуся в исходной таблице используется подстановка вида $column_name.

Если указан алгоритм трансформации GroupingSets - применение агрегаций не имеет смысла. В этом случае нужно указать агрегирующую функцию sum, которая будет просто "заглушкой"

4.1.8.1 Пример 1

```
<metric title="Сумма по столбцу"
        value="sum($colname)" />
```

По объективным причинам НЕ будет работать `sum($col_1+$col_2)`! Вместо этого используйте `sum($col_1)+sum($col_2)`.

4.1.8.2 Пример 2

```
<metric title="Средняя температура по больнице (отклонение от нормы)"
        value="avg($colname)-36.6" />
```

4.1.8.3 Пример 3

```
<metric title="Отношение двух столбцов"
        value="sum($colname1)/sum($colname2)" />
```

Строго говоря, пример выше не совсем корректен, и в случае `$colname2 = 0` в результате деления на ноль, выражение будет вычислено как `NaN` и результат будет проигнорирован. Возможно это то, что вам нужно в данной ситуации. Если такое поведение не является ожидаемым - ниже приведен более полный пример с проверкой деления на ноль.

```
<metric title="Отношение двух столбцов с проверкой DIV/0"
        value="if(sum($colname2)=0,0,sum($colname1)/sum($colname2))" />
```

4.1.9 HTTP API

В production окружении импортер запускается как сервис, управляемый вызовами по HTTP. По умолчанию импортер слушает TCP порт 8192. Запущенный сервис не выполняет никакой работы, пока не поступит команда по HTTP API. Предоставленный API не является публичным, не использует авторизацию/аутентификацию и должен быть использован только в безопасном окружении.

Ниже перечислены команды HTTP для запуска задач по загрузке данных, отслеживанию состояния процесса и очереди задач.

4.1.9.1 Состояние процесса

/pid

- HTTP Method: GET
- Response type: text/plain
- Description: PID запущенного процесса

```
curl http://localhost:8192/pid
16686
```

/health

- HTTP Method: GET
- Response type: application/json
- Description: состояние процесса

```
curl http://localhost:8192/health
{
  "TRN_TIME" : 0,
  "LOADED_RPS" : 0,
  "LDR_TIME" : 0,
  "LOADED_ROWS" : 0,
  "EXT_TIME" : 0,
  "FRAMES_FAILED" : 0,
  "SOURCE_ROWS" : 0,
  "FRAMES_QUEUE" : 0,
  "TIME" : 325119,
  "PID" : 16686,
  "AGG_TIME" : 0,
  "FRAMES_TOTAL" : 0,
  "SOURCE_RPS" : 0
}
```

/shutdown

- HTTP Method: GET
- Response type: text/plain
- Description: остановить запущенное приложение немедленно. Текущая очередь задач не сохраняется.

```
curl http://localhost:8192/shutdown
OK
```

4.1.9.2 Загрузка данных

/import/batch/{batch}/{date}

- HTTP Method: GET
- Response type: application/json
- Description: загрузка данных для указанного batch за указанную дату.

```
curl http://localhost:8192/import/batch/batch1/2010-01-01
OK
```

/import/{from}/{to}

- HTTP Method: GET
- Response type: application/json
- Description: загрузка данных для всех batch за указанный интервал дат с даты from (включительно) по дату to (исключительно). Даты указываются в формате `ISO8601`

```
curl http://localhost:8192/import/2010-01-01/2017-01-01
OK
```

/queue/start

- HTTP Method: GET
- Response type: text/plain
- Description: запуск очереди задач.

```
curl http://localhost:8192/queue/start
OK
```

/queue/stop

- HTTP Method: GET
- Response type: text/plain
- Description: остановка очереди задач

```
curl http://localhost:8192/queue/stop
OK
```

/frames

- HTTP Method: GET
- Response type: application/json
- Description: список всех задач

```
curl http://localhost:8192/frames
[ {
  "uuid" : "c238c1e0-5242-4187-a9b5-a26a5b617842",
  "period" : "[YEAR:2013-01-01 00:00:00]",
  "route" : "default:batch1:ds_000:data:val",
  "state" : "QUEUE"
}, {
  "uuid" : "f5cc463c-544d-4c2b-ad3a-07defdb84137",
  "period" : "[YEAR:2012-01-01 00:00:00]",
  "route" : "default:batch1:ds_000:data:val",
  "state" : "COMPLETED"
} ]
```

/frames/{state}

- HTTP Method: GET
- Response type: application/json
- Description: список задач с указанным состоянием

```
curl http://localhost:8192/frames/completed
[ {
  "uuid" : "f5cc463c-544d-4c2b-ad3a-07defdb84137",
  "period" : "[YEAR:2012-01-01 00:00:00]",
  "route" : "default:batch1:ds_000:data:val",
  "state" : "COMPLETED"
} ]
```

/frames/count

- HTTP Method: GET
- Response type: application/json
- Description: количество задач по состояниям

```
curl http://localhost:8192/frames/count
{
  "QUEUE" : 1024,
  "PROCESS" : 4,
  "COMPLETED" : 7
}
```

/frames/purge

- HTTP Method: GET
- Response type: application/json
- Description: очистка списка всех завершенных задач, возвращает количество удаленных объектов

```
curl http://localhost:8192/frames/purge
7
```

/ds/{schema}/checksum

- HTTP Method: GET
- Response type: text/plain
- Description: контрольная сумма датасета с указанным именем схемы

```
curl http://localhost:8192/ds/ds_000/checksum
1156777258200
```

4.1.9.3 XML конфигурация

/batches

- HTTP Method: GET
- Response type: application/xml
- Description: XML конфигурация батчей

```
curl http://localhost:8192/batches
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<batches>
    <batch id="batch1" sourceId="default" datasetId="ds_000">
        <request> select * from source_data where yyyy='$date' </request>
        <period sourceFormat="yyyy" titleFormat="yyyy" type="year"/>
        <location title="$l1">
            <location title="$l2">
                <location title="$l3" srcId="$loc_id"/>
            </location>
        </location>
        <metric title="Все метрики">
            <metric value="sum($v)" unitId="3" title="$m1">
                <metric title="$m2">
                    <metric title="$m3">
```

```
                        <metric title="$m4">
                            <metric title="$m5"/>
                        </metric>
                    </metric>
                </metric>
            </metric>
        </metric>
    </batch>
</batches>
```

/datasets

- HTTP Method: GET
- Response type: application/xml
- Description: XML конфигурация датасетов

```
curl http://localhost:8192/datasets
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<datasets>
    <dataset id="ds_000">
        <url>jdbc:postgresql://127.0.0.1:5432/mi</url>
        <username>bi</username>
        <password>bi</password>
        <locale>ru</locale>
        <timezone>W-SU</timezone>
        <units>
            <unit id="1" title="Граждане"/>
            <unit id="2" title="Штуки"/>
            <unit id="3" title="₽"/>
        </units>
    </dataset>
</datasets>
```

/dictionaries

- HTTP Method: GET
- Response type: application/xml
- Description: XML конфигурация словарей

```
curl http://localhost:8192/dictionaries
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dictionaries>
    <dictionary id="DictWithoutDefaultValue">
        <entry src="Key1" dst="Value1"/>
```

```
        <entry src="Key2" dst="Value2"/>
    </dictionary>
    <dictionary id="DictWithDefaultValue" default="Default">
        <entry src="Key1" dst="Value1"/>
        <entry src="Key2" dst="Value2"/>
    </dictionary>
</dictionaries>
```

/sources

- HTTP Method: GET
- Response type: application/xml
- Description: XML конфигурация источников данных

```
curl http://localhost:8192/sources
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<sources>
    <source id="default" url="jdbc:sqlite:data.sqlite"/>
</sources>
```

/loc/spatial/push

- HTTP Method: POST
- Response type: text/plain
- Description: синхронизация геоданных из указанного файла в датасет, указанный с HTTP заголовке `X-DS`

```
curl -X POST -H 'X-DS:ds_000' -H 'Content-type:application/xml' -d ↩
'@spatial.xml' http://localhost:8192/loc/spatial/push
OK
```

Глава 5

Библиотека bixel

Библиотека `bixel` используется для взаимодействия внешних виджетов с клиентской частью LuxmsBI.

Публичный репозиторий: `https://github.com/rcslabs/bixel`

Внешний виджет представляет собой html файл, доступный по http, который загружается в LuxmsBI через iframe.

Библиотека bixel представляет собой интерфейс для обмена сообщениями с клиентской частью.

5.1 Подключение библиотеки в свой проект

Библиотека состоит из одного файла (bixel.js), который может быть подключен разными способами:

5.1.1 Подключение через github

Достаточно просто подключить

```
<script src="https://rawgit.com/rcslabs/bixel/master/bixel.js"></script>
```

в свой html файл.

5.1.2 Подключение локально

Скачать bixel.js в папку с html файлом и указать

```
<script src="bixel.js"></script>
```

5.1.3 Подключение при помощи bower

Запустить команду

```
bower install git://github.com/rcslabs/bixel
```

После чего можно подключить файл

```
<script src="bower_components/bixel/bixel.js"></script>
```

или интегрировать в свою систему сборки (например, grunt, bower-requirejs).

5.1.4 Подключение через require.js

Библиотека bixel поддерживает загрузку через require.js

Например:

index.html:

```
<!DOCTYPE html>
<html>
<head>
  <script data-main="app" ↩
src="https://cdnjs.cloudflare.com/ajax/libs/require.js/2.2.0/require.min.js"></script>
</head>
<body>
  ...
</body>
</html>
```

app.js:

```
requirejs.config({
  paths: {
    bixel: 'bower_components/bixel/bixel'
  }
});

require(['bixel'], function(bixel) {
  bixel.init();
  bixel.on('load', function(data, axis) {
    // ...
  });
});
```

5.2 API библиотеки bixel

5.2.1 Методы библиотеки bixel

5.2.1.1 Метод bixel.init

Метод bixel.init используется для инициализации библиотеки. В качестве единственного необязательного параметра можно передать настройки - объект содержащий необязательные поля:

- `metricsCount` - количество метрик, в которых заинтересован виджет
- `locationsCount` - количество метрик, в которых заинтересован виджет
- `periodsCount` - количество метрик, в которых заинтересован виджет

Если эти опции не указаны, будут передаваться все, которые выбраны на панелях интерфейса LuxmsBI или в настройках дэша, а также данные будут загружаться по всему кубу.

Метод `bixel.init` возвращает объект типа Promise, который сигнализирует о результате.

Пример:

```
bixel.init({
  periodsCount: 1,
  metricsCount: 1
}).then(function() {
  // succeeded
}, function() {
  // error
});
```

5.2.1.2 Метод bixel.on

Метод bixel.on добавляет подписку на события.

```
bixel.on(eventType, callback)
```

- eventType - строка, может принимать значения `loading`, `load`, `no-data`
- callback - функция, вызываемая по событию, в зависимости от типа принимает разное количество аргументов.

5.2.2 Функции обратного вызова

5.2.2.1 Функция обратного вызова loading

Callback вызывается в момент начала загрузки данных. В качестве единственного параметра получает объект типа `bixel.Axis`.

```
bixel.on('loading', function(axis) {
  console.log('Началась загрузка данных по кубу размера' +
    axis.getMetrics().length + 'x' +
    axis.getLocations().length + 'x' +
    axis.getPeriods().length);
});
```

5.2.2.2 Функция обратного вызова load

Callback вызывается, когда закончена загрузка данных и получает два параметра:

- data: `bixel.Data`
- axis: `bixel.Axis`

```
bixel.on('load', function(data, axis) {
  var metric = axis.getMetrics()[0];
  var location = axis.getLocations()[0];
  var period = axis.getPeriods()[0];
  var val = data.getValue(metric, location, period);
  console.log("Данные загружены:", value.toString());
});
```

5.2.2.3 Функция обратного вызова no-data

Callback вызывется, когда данных нет:

- Метрика, локация или период не выбраны на панелях интерфейса
- все данные имеют пустые значения

В качестве единственного параметра получает объект типа `bixel.Axis`.

5.2.3 Объекты

5.2.3.1 Объект Axis

Объект, содержащий описание осей (метрики-локации-периоды)

Реализует методы:

- `getMetrics()` - массив объектов `Metric`
- `getLocations()` - массив объектов `Location`
- `getPeriods()` - массив объектов `Period`
- `getUnits()` - массив объектов `Units` - только те, с которыми связаны метрики

5.2.3.2 Объект Data

Объект, хранящий полученные значения

Реализует методы:

- `getValue(z, y, x)` - получить значение в указанной точке. В качестве параметров принимает объекты `Metric`, `Location`, `Period` в любом порядке. Возвращает объект `DataItem` (Number)

5.2.3.3 Объект DataItem

В общем случае это js объект типа `Number` с переопределенным методом `toString`

Метод `toString` возвращает форматированное согласно соответствующему `Unit` значение (добавляет префикс, суффикс, название, разделители между разрядами числа)

Метод 'valueOf' вернет примитивный number

Если значение отсутствует в базе данных, то:

- `valueOf()` вернет NaN (можно проверить, вызвав `isNaN(val)`)
- `toString()` вернет строку `"-"`

В случае text-data значениями могут быть объекты типа String. В том с другом случае

```
typeof value === 'object'
```

Поэтому при необходимости проверки, ее можно осуществить через `instanceof`

```
if (val instanceof String) { ... }
if (val instanceof Number) { ... }
```

5.2.3.4 Объект Metric

Структура, содержащая описание метрики:

- `id`:string - уникальный идентификатор
- `title`:string - название метрики
- `color`:string - цвет, сопоставленный метрике
- `bgColor`:string - цвет фона (предпочтительный для закрашивания)
- `unit_id`:string - идентификатор `Unit`

5.2.3.5 Объект Location

Структура, содержащая описание локации:

- `id`:string - уникальный идентификатор
- `title`:string - название метрики
- `color`:string - цвет, сопоставленный метрике
- `bgColor`:string - цвет фона (предпочтительный для закрашивания)

5.2.3.6 Объект Period

Структура, содержащая описание периода:

- `id`:string - уникальный идентификатор
- `title`:string - название периода
- `color`:string - цвет, сопоставленный периоду
- `bgColor`:string - цвет фона (предпочтительный для закрашивания)

5.2.3.7 Объект Unit

Структура, содержащая описание еденицы измерения (размерности):

- `id`:string - уникальный идентификатор
- `value_prefix`:string - префикс, пишется перед значением
- `value_suffix`:string - суффикс, пишется после значения

Глава 6

Работа с библиотекой bixel. Пример 1.

6.1 Введение

Библиотека `bixel` используется для взаимодействия внешних виджетов с клиентской частью LuxmsBI

Публичный репозиторий: `https://github.com/rcslabs/bixel`

Внешний виджет представляет собой html файл, доступный по http, который загружается в LuxmsBI через iframe.

Библиотека bixel представляет собой интерфейс для обмена сообщениями с клиентской частью

6.2 Цель

Создать виджет на базе датасета `Налоги США`, который бы показывал штат с максимальным значением метрики `Taxes/Total taxes` по разным годам.

`LuxmsBI - Налоги США`

Дэш использует:

- Локации: все штаты
- Метрики: `T1`, "Total taxes"
- Периоды: выбранные в верхней панели

6.3 Шаг 1. index.html

Создаем файл `index.html`

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
</head>
<body>
</body>
</html>
```

6.4 Шаг 2. scripts

Будем использовать простой шаблон на базе библиотеки `knockout`. Подключаем библиотеки `knockout.js` и `bixel` внутри <head>:

```
<script ↩
src="https://cdnjs.cloudflare.com/ajax/libs/knockout/3.3.0/knockout-min.js"></script>
<script src="https://rawgit.com/rcslabs/bixel/master/bixel.js"></script>
```

6.5 Шаг 3. Разработка модели

Моделью (в терминах knockout) для шаблона будет объект, содержащий поля:

- `loading`: KnockoutObservable<boolean> - статус, идет ли загрузка в данный момент
- `period`: KnockoutObservable<`bixel.Period`> - текущий период, будет иметь тип "год"
- `metric`: KnockoutObservable<`bixel.Period`> - текущая метрика (всегда будет `Total taxes`)
- `location`: KnockoutObservable<`bixel.Location`> - штат, в котором значение метрики максимальное
- `value`: KnockoutObservable<`Number`> - максимальное значение метрики среди всех штатов

6.6 Шаг 4. Шаблон

В <body> создаем шаблон, основанный на модели

```
<div id="result">

  <span>Загрузка ...</span>



  <h2>Лучшее значение по всем штатам:</h2>
  <span data-bind="text:metric().title"></span>: <span ↩
data-bind="text:value"></span>
  <br/>
  <span data-bind="text:location().title"></span>
  <br/>
  <span>На период: </span><span data-bind="text:period().title"></span>

</div>
```

В момент загрузки (`loading() == true`) будем показывать надпись "Загрузка..."

Когда данные загружены покажем:

- название метрики (`metric().title`)
- найденное лучшее значение (`value`) - будет автоматически приведено к строке через переопределенный метод `toString()`, который форматирует значение согласно настройкам `unit` (добавляет знак доллара и разделители у числа)
- название штата с наилучшим значением (`location().title`)
- выбранный год (`period().title`)

6.7 Шаг 5. Создаем модель к шаблону

Добавляем <`script`> и в нем описываем структуру:

```
var model = {
  loading: ko.observable(true),
  period: ko.observable(null),
  metric: ko.observable(null),
  location: ko.observable(null),
  value: ko.observable(null)
};
```

Глобальная переменная `model` является описанием модели

Связываем модель с шаблоном (элемент с id=result)

```
ko.applyBindings(model, document.getElementById('result'));
```

6.8 Шаг 6. Инициализация bixel

Добавляем скрипт

```
bixel.init({
  periodsCount: 1,
  metricsCount: 1
});
```

Виджет заинтересован в одном периоде (выбранный год) и в одной метрике (Total taxes, указывается в настройках дэша)

6.9 Шаг 7. Обработка события loading

Добавляем скрипт

```
bixel.on('loading', function(axis) {
  model.loading(true);
});
```

Будем устанавливать в модели свойство `loading` в `true`

6.10 Шаг 8. Обработка события load

Обработка события load происходит, когда загружены данные

```
bixel.on('load', function(data, axis) {
  var locations = axis.getLocations();
  var metric = axis.getMetrics()[0];
  var period = axis.getPeriods()[0];
  //...
});
```

В соответствии с настройками из `bixel.init` в этом методе будет получена одна метрика и один период (размеры соответствующих массивов будут равны единице)

В качестве локаций придут все штаты, поэтому запоминаем их для дальнейшего использования

6.11 Шаг 9. Поиск лучшей локации

Будем использовать простой цикл:

```
var bestVal = -Infinity;     // Храним лучшее значение
var bestLocation = null;     // и лучшую локацию (штат)

for (var i = 0; i < locations.length; i++) {
  var value = data.getValue(locations[i], metric, period);
  if (value > bestVal) {
    bestLocation = locations[i];
    bestVal = value;
  }
}
```

На каждой итерации получаем значение для очередного штата, сравниваем с `bestVal` (Сравнение работает, поскольку это объекты типа `Number`)

После окончания работы цикла в переменных `bestVal` и `bestLocation` будут соответственно лучше значение и лучший штат

6.12 Шаг 10. Заполняем модель

```
model.location(bestLocation);
model.metric(metric);
model.period(period);
model.value(bestVal);
model.loading(false);
```

6.13 Шаг 11. Deploy

Итак, у нас получился index.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <script ↩
src="https://cdnjs.cloudflare.com/ajax/libs/knockout/3.3.0/knockout-min.js"></script>
  <script src="https://rawgit.com/rcslabs/bixel/master/bixel.js"></script>
</head>
<body>
  <h1>Внешний виджет</h1>
```

```
11  <div id="result">

13    <span>Загрузка ...</span>



17    <h2>Лучшее значение по всем штатам:</h2>
18    <span data-bind="text:metric().title"></span>: <span ↩
data-bind="text:value"></span>
19    <br/>
20    <span data-bind="text:location().title"></span>
21    <br/>
22    <span>На период: </span><span data-bind="text:period().title"></span>

24  </div>


27  <script>
28    // ui
29    var model = {
30      loading: ko.observable(true),
31      period: ko.observable(null),
32      metric: ko.observable(null),
33      location: ko.observable(null),
34      value: ko.observable(null)
35    };
36    ko.applyBindings(model, document.getElementById('result'));



40    bixel.init({
41      locationsCount: Infinity,
42      periodsCount: 1,
43      metricsCount: 1
44    });

46    bixel.on('loading', function(axis) {
47      model.loading(true);
48    });

50    bixel.on('load', function(data, axis) {
51      var locations = axis.getLocations();
52      var metric = axis.getMetrics()[0];
53      var period = axis.getPeriods()[0];

55      var bestVal = -Infinity;
56      var bestLocation = null;
```

```
58      for (var i = 0; i < locations.length; i++) {
59        var value = data.getValue(locations[i], metric, period);
60        if (value > bestVal) {
61          bestLocation = locations[i];
62          bestVal = value;
63        }
64      }

66      model.location(bestLocation);
67      model.metric(metric);
68      model.period(period);
69      model.value(bestVal);
70      model.loading(false);
71    });

73  </script>
74 </body>
75 </html>
```

Выкладываем `index.html` на какой-нибудь доступный web-сервер, например, по адресу http://example.com/best-state/index.html

6.14 Шаг 12. настройка дэша в LuxmsBI

Создаем новый дэш а датасете "Налоги США"

- Тип дэша (view_class): `external`
- Заголовок дэша: "Total taxes: лучший штат"
- config.dataSource.parameters: `["T1"]` (одна метрика Total taxes)
- config.dataSource.locations: "(level~2)" - все локации уровня 2 (штаты)
- config.url: "http://example.com/best-state/index.html"

6.15 Результат

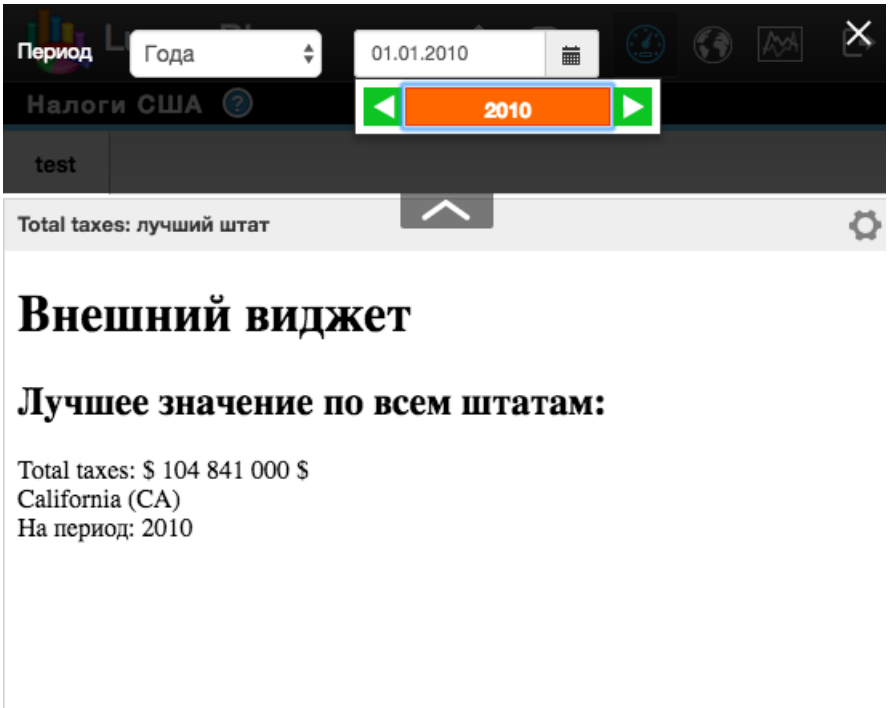

Рис. 6.1 Результат встраивания bixel в дэшборд Luxms BI