ИНФРАСТРУКТУРА ЭЛЕКТРОННОГО ПРАВИТЕЛЬСТВА

ВЫПОЛНЕНИЕ РАБОТ ПО РАЗВИТИЮ КОМПОНЕНТА «ВИТРИНА ДАННЫХ» ЕДИНОЙ СИСТЕМЫ МЕЖВЕДОМСТВЕННОГО ЭЛЕКТРОННОГО ВЗАИМОДЕЙСТВИЯ

Руководство администратора Компонента «Витрина данных»

Версия 1.16.3

Листов 304

Москва, 2024

СОДЕРЖАНИЕ

АННОТАЦИЯ

В данном программном документе приведено Руководство администратора Компонента «Витрина данных», предназначенного для загрузки публикуемых данных в отдельную базу данных на стороне поставщика данных, а также для формирования отдельной базы данных в соответствии с результатами выполнения запросов на предоставление или репликации данных со стороны получателя данных.

В разделе «Общие сведения о программе» указаны назначение и возможности Программы и сведения о технических и программных средствах, обеспечивающих выполнение данной Программы.

В разделе «Настройка программы» приведены описания действий по настройке программы на состав технических и программных средств.

В разделе «Запуск и остановка» приведены описания действий по запуску и остановке модулей Программы.

В разделе «Резервное копирование» приведены рекомендации по выполнению планового резервного копирования, контролю результатов резервного копирования, восстановлению информации из резервных копий.

В разделе «Дополнительные возможности» описание дополнительных функциональных возможностей Программы и способов их выбора.

В разделе «Сообщения администратору» приведены описания сообщений, выдаваемых в ходе выполнения настройки, проверки Программы, а также в ходе выполнения Программы, описание их содержания и действий, которые необходимо предпринять по этим сообщениям.

В разделе «Метрики в модулях Компонента «Витрина данных» приведена таблица метрик модулей Компонента «Витрина данных» данных с перечнем функций по каждому модулю.

В приложении к руководству администратора приведены описания спецификаций модулей Программы, функции LISTAGG, ее синтаксис и поддержка в модулях Программы, пример XML-файла со структурой витрины а также инструкция по эксплуатации CSV-Uploader.

1 ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ

1.1 Назначение программы

Национальная система управления данными (далее – НСУД) представляет собой систему, состоящую из взаимосвязанных элементов информационно-технологического, организационного, методологического, кадрового и нормативно-правового характера и обеспечивающую достижение целей и выполнение задач, обозначенных в Концепции Национальной системы управления данными, утвержденной распоряжением Правительства Российской Федерации от 3 июня 2019 года № 1189-р.

НСУД предназначена для управления информацией, содержащейся в информационных системах органов и организаций государственного сектора, а также в информационных ресурсах, созданных в целях реализации полномочий органов и организаций государственного сектора (далее – государственные данные) и для осуществления информационного обмена между Поставщиками и Получателями данных, присоединившимися к НСУД (далее – Участники НСУД).

Управление процессами информационного обмена между Участниками НСУД осуществляется средствами федеральной государственной информационной системы «Единая информационная платформа Национальной системы управления данными» (далее – ФГИС «ЕИП НСУД»).

Для передачи данных между Участниками НСУД используется среда взаимодействия НСУД, состоящая из Системы межведомственного электронного взаимодействия 3.0 (далее – СМЭВ) и (или) подсистемы обеспечения доступа к данным СМЭВ (далее – ПОДД СМЭВ) (СМЭВ 4.0), обеспечивающих транспорт и процессинг данных, а также агентов ПОДД СМЭВ, устанавливаемых на стороне Участников НСУД.

Для формирования и (или) для получения данных с использованием среды взаимодействия НСУД необходим комплекс программных и технических средств в составе информационно-телекоммуникационной инфраструктуры участника НСУД, описываемое в данном документе.

Компонент «Витрина данных» является частью НСУД и предназначена для загрузки публикуемых данных в отдельную БД на стороне Поставщика данных. Программа представляет собой типовое программное обеспечение, устанавливаемое на стороне поставщиков/потребителей данных.

1.2 Возможности программы

В настоящий момент реализовано две конфигурации Программы:

- Стандарт;
- Лайт.

Возможности конфигурации Стандарт

Программа обеспечивает выполнение следующих задач:

- описание логической модели данных;
- настройка программы и структуры таблиц в ее БД для хранения публикуемых данных;
- загрузка и хранение публикуемых данных в БД программы;
- извлечение данных из внешних систем (внешних ИС по отношению к Витрине данных НСУД);

- выполнение запросов в соответствии с протоколом ПОДД через механизмы ПОДД СМЭВ:
  - поддержка протокола коммуникации Агента СМЭВ4;
  - предоставление публикуемых данных (в т. ч. BLOB-объектов и/или с использованием табличных параметров);
  - генерация формируемых документов на основании публикуемых данных;
  - репликация публикуемых данных (в качестве витрины-поставщика);
  - получение реплицируемых данных (в качестве витрины-получателя).
- обмен в соответствии с протоколом СМЭВ3:
  - подключение к СМЭВ3 как информационной системы участника взаимодействия;
  - обработку запросов на предоставление публикуемых данных (видов сведений), в т.ч. BLOB-объектов;
  - инициативная рассылка оповещений об обновлении публикуемых данных.
- публикация конечных точек API для обработки запросов с использованием спецификации OpenAPI версии 3;
- предоставление публикуемых данных информационным системам с использованием интерфейса REST-запросов;
- восстановление данных в непротиворечивое состояние после сбоев;
- поддержка языка SQL (см. Приложение 2 «Поддержка функций SQL», раздел «1 SQL-синтаксис»);
- журналирование событий функциональных блоков;
- мониторинг информации о работоспособности экземпляра Программы.

Возможности конфигурации Лайт

Программа обеспечивает выполнение следующих задач:

- автоматическая настройка взаимосвязей между компонентами программы;
- автоматический запуск всех необходимых компонентов программы после установки;
- автоматическая настройка витрины и структуры ее таблиц на основании содержимого XML-файла, загружаемого через пользовательский web-интерфейс;
- выгрузка шаблона через графический интерфейс (для упрощения процесса подготовки загружаемых данных);
- загрузка данных в витрину:
  - через графический интерфейс;
  - REST API;
  - файловый обмен.
- настройка параметров работы витрины через графический интерфейс;
- выполнение запросов на предоставление данных в соответствии с протоколом ПОДД через механизмы СМЭВ ПОДД.

1.3 Технические и программные средства

Рекомендации по аппаратному и программному обеспечению, а также необходимая конфигурация сети для оптимального баланса между производительностью и стабильностью работы всех компонентов программы приведены в разделе Рекомендуемые технические и программные средства документа «Техническое описание системы».

2 НАСТРОЙКА ПРОГРАММЫ

2.1 Настройка технических средств

Серверы, на которых устанавливается Компонент «Витрина данных», должны соответствовать техническим характеристикам указанным в документе «Техническое описание системы» раздел Рекомендуемые технические и программные средства, в котором приводятся требования к серверному оборудованию (CPU, RAM, HDD и т.д.), программному обеспечению и каналам связи.

Необходимые настройки для серверов описаны в документе «Руководство по установке», в котором приводятся требования к серверам, доступности портов для каждого сервера, настройка протоколов, наличие библиотек и т.д.

2.2 Настройка программных средств

Все предварительные действия необходимые перед установкой программы, процесс установки и проверка корректной установки программы описан в документе «Руководство по установке» в разделе «Подготовка к установке».

Внимание:

Программные средства настраиваются в зависимости от используемой конфигурации. Состав компонентов приведен в разделе Состав компонентов в дистрибутиве документа «Техническое описание системы».

2.2.1 Настройка ProStore

2.2.1.1 Настройка Сервиса исполнения запросов (query-execution)

Конфигурация инстанса узла **Сервиса исполнения запросов** (query-execution) Prostore представляет собой текстовый YAML-файл, параметры которого организованы в древовидную структуру.

Файл конфигурации содержит логику и порядок работы **Сервиса исполнения запросов**.

Для наглядности конфигурация сервиса исполнения запросов разделена на отдельные секции.

Пример файла конфигурации Prostore приведен в разделе Конфигурация ноды документации Prostore.

2.2.1.2 Настройка коннекторов

Следующие коннекторы требуют настройки конфигурации:

- Kafka-Clickhouse reader connector;
- Kafka-Clickhouse writer connector;
- Kafka-Postgres reader connector;
- Kafka-Postgres writer connector;
- Kafka Jet writer connector.

Конфигурация коннектора представляет собой текстовый YAML-файл, параметры которого организованы в древовидную структуру.

Пример файла конфигурации Prostore приведен в разделе Конфигурация коннекторов документации Prostore.

2.2.2 Настройка СМЭВ QL Сервера

2.2.2.1 Конфигурирование сервера

Конфигурирование СМЭВ QL сервера выполняется путем изменения параметров настроек, определенных в файлах `credentials.yaml` и `application.yaml`.

- **application.yaml** - конфигурирует поведение сервера;
- **credentials.yaml** - конфигурирует представление сервера.

2.2.2.1.1 Конфигурация файла application.yml

```yaml
ktor:
  deployment:
    port: "$PORT:8080"
  application:
    modules:
      - ru.gov.digital.smevql.ApplicationKt.mainModule

sources:
  directory: "$SOURCES_DIR:sources"
models:
  directory: "$MODELS_DIR:models"
states:
  directory: "$STATES_DIR:states"

swagger:
  file: smevql-openapi.yaml # путь к файлу openapi спецификации
  servers:
    - "http://127.0.0.1:8080/smevql/api/v1"

storage: # Блок параметров хранения информации
  adapter: redis # На текущий момент поддерживается только redis
  pool: # Настройка подключений к redis
    - host: 127.0.0.1
      port: 6379
  max-pool-size: 20 # Максимальный размер пула соединений
  user: "" # Пользователь для подключения к redis
  password: "" # Пароль

access: # Блок настроек доступа к выполнению операций чтения данных и операций стейтмашины. Допускается задание черного или белого списка
  black-list: [ ] # Указывает список потребителей, для которых доступ запрещен
  white-list: [ ] # Указывает список потребителей, для которых доступ разрешен

request:
  strategy: delegate # Стратегия исполнения запросов delegate|atomic
  timeout: 20s # Таймаут исполнения запросов
  base-path: smevql/api/v1 # Префикс для всех роутов
  max-nested-level: 5 # Предельная вложенность запрашиваемых ресурсов
  pagination:
    default: 100 # Количество элементов на странице по умолчанию
    max: 1000 # Максимальное количество элементов на странице
  logging:
    long:
      duration: 20s # Продолжительность исполнения запросов к источникам, выше которой необходимо производить логирование
      percentage: 100 # Процент логирования длительных запросов к источникам. Допустимо использовать вещественные числа, например, 0.1 - только каждый тысячный долгий запрос
  limits: # Блок параметров конфигурации лимитов
    enabled: true # флаг влюкчения проверок лимитов
```

```
    total: # Параметры по всем запросам
      value: 500000 # Общее число запросов в период времени
      period: 1D # Период лимитирования
    mnemonic: # Блок настроек лимитирования по потребителям
      value: 5000 # Число запросов в период времени
      period: 1D # Период лимитирования
    purpose: # Блок настроек лимитирования по целям
      value: 5000 # Число запросов в период времени
      period: 1D # Период лимитирования
    user: # Блок настроек лимитирования по пользователям
      value: 1000 # Число запросов в период времени
      period: 1D # Период лимитирования
    records-ttl:
      day: 1M # Период хранения дневной статистики
      week: 1M # Период хранения статистики за неделю
      month: 1Y # Период хранения статистики за месяц
      year: 2Y # Период хранения статистики за год
  async: # Блок настроек асинхронного выполнения запросов
    request-in: 10s # Значение интервала опроса получения данных асинхронного
результата. Выдается в качестве значения атрибута request_in на запрос получения
данных
    read-timeout: 5m # Таймаут вычитывания асинхронных данных из источников.
Используется для источников с типом smevql, если была возвращена информация по
асинхронным результатам

delta:
  commit-interval: 60s # Интервал фиксации дельты источника при изменении данных
  force-commit-interval: 30m # Интервал принудительной фиксации дельт всех источников

state:
  max-nested-event: 5 # Максимально допустимая вложенность связанных переходов стейт
машины
  max-updated-rows: 1 # Максимальное количество обновляемых строк при событии стейт
машины

index-recommendations: # Рекомендации по аналитике
  enabled: true # Флаг включения формирования рекомендаций
  period: 7D  # Период формирования. 7 дней
  concurrency: 10 # Количество параллельных корутин сохранения статистики

# Массив описания standalone таблиц
standalone-tables: [ ]
# Пример описания
# standalone-tables:
#  - readable-table: "misdm05.readable_book"
#    writable-table: "misdm05.writable_book"
#    anchor: "update_at"
#    soft-delete: "delete_at"

# Массив описания proxy таблиц
proxy-tables: [ ]
# Пример описания
# proxy-tables:
#  - table: "misdm05.notebook"
#    anchor: "update_at"
#    soft-delete: "delete_at"

push: # Настройки отправки push уведомлений
  notification-path: "/{target}/push/notify" # Шаблон пути агента, на который
необходимо отправлять нотификации
  state-machine-enabled: false # Признак публикации нотификаций на основе событий
стейт машины
```

```
  status-prostore-enabled: true # Признак публикации нотификаций на основе событий статусов Простора
  prostore:
    status-event-topic:
      topic: "$PS_STATUS_EVENT_TOPIC:status.event"
      property:
        bootstrap.servers: "$PS_KAFKA:localhost:9092"
        group.id: smevql-server-status-event
        auto.offset.reset: earliest
  retry:
    max-attempts: 3 # Количество попыток отправки нотификации
    min-period: 5s # Минимальный период ожидания перед повторной попыткой
    max-period: 10s # Максимальный период ожидания перед повторной попыткой

storage-queue:
  host: "$QUEUE_HOST:localhost"
  port: "$QUEUE_PORT:5432"
  database: "$QUEUE_DATABASE:smevql"
  schema: "$QUEUE_SCHEMA:smevqlqueue"
  user: "$QUEUE_USER:"
  password: "$QUEUE_PASSWORD:"

environment: "$ENVIRONMENT:dev"

agent: # Параметры конфигурирования агента СМЭВ4
  host: 127.0.0.1 # Хост агента
  port: 8171 # Порт приема api-gateway запросов
  mnemonic: "" # Мнемоника агента

signature: # Блок настроек механизма подписания
  enabled: true # Признак включения подписания и проверки подписи
  validate-enabled: false # Признак предоставления дополнительного URL проверки подписи
  algorithm: "GOST3410_2012_256" # Алгоритм формирования подписи
  serial-number: "" # Серийный номер ключа подписи
  alias: "" # Алиас ключа. Заполняется либо серийный номер, либо алиас
  notarius: # Блок настроек сервиса Notaris, используемый для подписания
    host: localhost # Хост, на котором доступен Notaris
    port: 8080 # Порт, на котором доступен Notaris
```

Настройка конфигурации **СМЭВ QL сервера** осуществляется путем редактирования параметров настроек в файле `application.yml`, в котором могут быть настроены следующие секции:

- `sources` - определение директории хранения источников;
- `models` - определение директории хранения моделей;
- `states` - определение директории хранения состояний;
- `swagger` - настройка файла openapi спецификации;
- `storage` - управление подключением к внутреннему хранилищу данных СМЭВ-QL;
- `access` - блок настроек доступа к выполнению операций чтения данных и операций стейт-машины;
- `request` - блок настроек исполнения запросов;
- `delta` - управление принудительными коммитами дельт витрин данных;
- `state` - установка ограничений в машинах-состояний;
- `index_recommendations` - рекомендации по аналитике;
- `push` - настройки отправки push уведомлений; - `environment` - определяет значение

среды разработки;

- `agent` - параметры конфигурирования Агента СМЭВ4;
- `signature` - блок настроек механизма подписания.

2.2.2.1.1.1 Секция storage

В секции `storage` настраиваются параметры хранения информации, например:

```
storage: # Блок параметров хранения информации
  adapter: redis # На текущий момент поддерживается только redis
  pool: # Настройка подключений к redis
    - host: 127.0.0.1
      port: 6379
  max_pool_size: 20 # Максимальный размер пула соединений
  user: "" # Пользователь для подключения к redis
  password: "" # Пароль
```

Параметры конфигурации

- `adapter` - система хранения данных, на текущий момент поддерживается только redis;
- `pool` - указание host и port для подключения к хранилищу данных;
- `host` - адрес хоста;
- `port` - порт хоста;
- `max_pool_size` - максимальный размер пула соединений;
- `user` - имя пользователя для авторизации в системе хранения данных;
- `password` - пароль для авторизации в системе хранения данных.

2.2.2.1.1.2 Секция access

В секции `access` настраивается доступ к выполнению операций чтения данных и операций стейт-машины.

Допускается задание черного или белого списка.

Например:

```
access: # Блок настроек доступа к выполнению операций чтения данных и операций стейтмашины. Допускается задание черного или белого списка
  black_list: [ ] # Указывает список потребителей, для которых доступ запрещен
  white_list: [ ] # Указывает список потребителей, для которых доступ разрешен
```

Параметры конфигурации

- `black_list` - перечень мнемоник ИС Потребителей, которым запрещен доступ к СМЭВ QL. **Не заполняется, если заполнен white_list!**
- `white_list` - перечень мнемоник ИС Потребителей, которым разрешен доступ к СМЭВ QL. **Не заполняется, если заполнен black_list!**

2.2.2.1.1.3 Секция request

В секции `request` хранятся настройки исполнения запросов, например:

```
request:
  timeout: 20s # Таймаут исполнения запросов
  base_path: smevql/api/v1 # Префикс для всех роутов
  max_nested_level: 5 # Предельная вложенность запрашиваемых ресурсов
  pagination:
      default: 100 # Количество элементов на странице по умолчанию
      max: 1000 # Максимальное количество элементов на странице
  logging:
      long:
      duration: 20s # Продолжительность исполнения запросов к источникам, выше
```

```
которой необходимо производить логирование
      percentage: 100 # Процент логирования длительных запросов к источникам.
Допустимо использовать вещественные числа, например, 0.1 - только каждый тысячный
долгий запрос
  limits: # Блок параметров конфигурации лимитов
      total: # Параметры по всем запросам
      value: 500000 # Общее число запросов в период времени
      period: 1D # Период лимитирования
      mnemonic: # Блок настроек лимитирования по потребителям
      value: 5000 # Число запросов в период времени
      period: 1D # Период лимитирования
      purpose: # Блок настроек лимитирования по целям
      value: 5000 # Число запросов в период времени
      period: 1D # Период лимитирования
      user: # Блок настроек лимитирования по пользователям
      value: 1000 # Число запросов в период времени
      period: 1D # Период лимитирования
      records_ttl:
      day: 1M # Период хранения дневной статистики
      week: 1M # Период хранения статистики за неделю
      month: 1Y # Период хранения статистики за месяц
      year: 2Y # Период хранения статистики за год
  async: # Блок настроек асинхронного выполнения запросов
      request_in: 10s # Значение интервала опроса получения данных асинхронного
результата. Выдается в качестве значения атрибута request_in на запрос получения
данных
      read_timeout: 5m # Таймаут вычитывания асинхронных данных из источников.
Используется для источников с типом smevql, если была возвращена информация по
асинхронным результатам
```

Параметры конфигурации

- `timeout` - таймаут исполнения запросов;
- `base_path` - префикс для роута запросов. Например, `smevql/api/v1`;
- `max_nested_level` - предельная вложенность запрашиваемых ресурсов;
- `pagination` - управление количеством элементов при ответе. Содержит следующие атрибуты:
  - `default` - количество элементов на странице по умолчанию;
  - `max` - максимальное количество элементов на странице;
- `logging` - управление логированием «долгих» запросов. Содержит следующие атрибуты:
  - `duration` - продолжительность исполнения запросов к источникам, выше которой необходимо производить логирование. Например: `20s`;
  - `percentage` - процент логирования длительных запросов к источникам. Допустимо использовать вещественные числа, например, `0.1` - только каждый тысячный долгий запрос;
- `limits` - установка ограничений на количество запросов. Содержит следующие атрибуты:
  - `total` - общее допустимое количество запросов к серверу:
    - `value` - целочисленное значение количества запросов, например: `1000`;
    - `period` - на какой период устанавливается ограничение, например: 1D (на сутки)
  - `mnemonic` - допустимое для одного потребителя данных количество запросов к серверу:

        - `value` - целочисленное значение количества запросов, например: `100`;
        - `period` - на какой период устанавливается ограничение, например: `1D` (на сутки);
      - `purpose` - допустимое количество запросов к одному ресурсу (таблице, объекту)
        - `value` - целочисленное значение количества запросов, например: `100`;
        - `period` - на какой период устанавливается ограничение, например: `1D` (на сутки);
      - `user` - допустимое количество запросов для пользователя:
        - `value` - целочисленное значение количества запросов, например: `100`;
        - `period` - на какой период устанавливается ограничение, например: `1D` (на сутки);
      - `records_ttl` - настройка хранения статистики по лимитам:
        - `day: 1M` - период хранения дневной статистики;
        - `week: 1M` - период хранения статистики за неделю;
        - `month: 1Y` - период хранения статистики за месяц;
        - `year: 2Y`- период хранения статистики за год;
  - `async` - блок настроек асинхронного выполнения запросов. Содержит следующие атрибуты:
      - `request_in` - значение интервала опроса получения данных асинхронного результата. Выдается в качестве значения атрибута request_in на запрос получения данных;
      - `read_timeout` - таймаут вычитывания асинхронных данных из источников. Используется для источников с типом smevql, если была возвращена информация по асинхронным результатам

2.2.2.1.1.4 Секция delta

В секции `delta` настраивается управление принудительными коммитами дельт витрин данных.

Например:

```
delta:
  commit_interval: 60s # Интервал фиксации дельты источника при изменении данных
  force_commit_interval: 30m # Интервал принудительной фиксации дельт всех источников
```

Параметры конфигурации

- `commit_interval` - интервал фиксации дельты источника при изменении данных, например 60s;
- `force_commit_interval` - интервал принудительной фиксации дельт всех источников, например 30s.

Если `commit_interval: 0` и `force_commit_interval: 0`, то для добавления/изменения/удаления данных в Prostore не используется механизм открытия и закрытия дельт, а данные меняются прямым запросом.

При этом возникают следующие ограничения:

- в бекап текущей реализации данные, софрмированние вне дельт, не попадают;
- в результаты запросов к материализованным представлениям данные, софрмированние вне дельт, не попадают;
- в рамках подписок данные, софрмированние вне дельт, не передаются.

2.2.2.1.1.5 Секция state

В секции `state` устанавливаются ограничения в стейт машинах.

Например:

```
state:
  max_nested_event: 5 # Максимально допустимая вложенность связанных переходов стейт машины
  max_updated_rows: 1 # Максимальное количество обновляемых строк при событии стейт машины
```

Параметры конфигурации

- `max_nested_event` - максимально допустимая вложенность связанных переходов стейт машины, например 5;
- `max_updated_rows` - максимальное количество обновляемых строк при событии стейт машины, например 1.

2.2.2.1.1.6 Секция index_recommendations

В секции `index_recommendations` настраиваются рекомендации по аналитике, например:

```
index_recommendations: # Рекомендации по аналитике
  period: 7D  # Период формирования. 7 дней
```

Параметры конфигурации

- `period` - устанавливается период формирования аналитики, например 7 дней.

2.2.2.1.1.7 Секция standalone-tables

В секции `standalone-tables` настраиваются данные standalone таблиц.

Например:

```
standalone-tables: [ ]
# Пример описания
# standalone-tables:
#  - readable-table: "misdm05.readable_book"
#    writable-table: "misdm05.writable_book"
#    anchor: "update_at"
#    soft-delete: "delete_at"
```

Параметры конфигурации

- `readable-table` - название readable таблицы;
- `writable-table` - название writable таблицы;
- `anchor` - название атрибута характеризующего дату и время последнего изменения данных в нетемпоральной таблице;
- `soft-delete` - название атрибута характеризующего дату и время удаления данных в нетемпоральной таблице.

2.2.2.1.1.8 Секция proxy-tables

В секции `proxy-tables` настраиваются данные прокси-таблиц.

Например:

```
# Массив описания proxy таблиц
  proxy-tables: [ ]
  # Пример описания
  # proxy-tables:
  #  - table: "misdm05.notebook"
  #    anchor: "update_at"
  #    soft-delete: "delete_at"
```

Параметры конфигурации

- `table` - название прокси таблицы;
- `anchor` - название атрибута характеризующего дату и время последнего изменения данных в нетемпоральной прокси таблице;
- `soft-delete` - название атрибута характеризующего дату и время удаления данных в нетемпоральной прокси таблице.

2.2.2.1.1.9 Секция push

В секции `push` содержатся настройки сервиса формирования push-уведомлений. Например:

```
push: # Настройки отправки push уведомлений
  notification-path: "/{target}/push/notify" # Шаблон пути агента, на который необходимо отправлять нотификации
  state-machine-enabled: false # Признак публикации нотификаций на основе событий стейт машины
  status-prostore-enabled: true # Признак публикации нотификаций на основе событий статусов Простора
  prostore:
    status-event-topic:
      topic: "$PS_STATUS_EVENT_TOPIC:status.event"
      property:
        bootstrap.servers: "$PS_KAFKA:localhost:9092"
        group.id: smevql-server-status-event
        auto.offset.reset: earliest
  retry:
    max-attempts: 3 # Количество попыток отправки нотификации
    min-period: 5s # Минимальный период ожидания перед повторной попыткой
    max-period: 10s # Максимальный период ожидания перед повторной попыткой
```

Параметры конфигурации

- `notification_path` - шаблон пути агента, на который необходимо отправлять нотификации;
- `state_machine_enabled` - признак публикации нотификаций на основе событий стейт машины;
- `retry` - настройки попыток отправок нотификаций.

2.2.2.1.1.10 Секция agent

В секции `agent` указываются параметры подключения к Агенту СМЭВ4 поставщика, например:

```
agent: # Параметры конфигурирования агента СМЭВ4
  host: 127.0.0.1 # Хост агента
  port: 8171 # Порт приема api-gateway запросов
  mnemonic: "" # Мнемоника агента
```

Параметры конфигурации

- `host` - хост агента;
- `port` - порт приема api-gateway запросов;
- `mnemonic` - мнемоника агента СМЭВ4.

2.2.2.1.1.11 Секция signature

В секции `signature` настраивается управление механизмом цифровой подписи. Например:

```
signature: # Блок настроек механизма подписания
  enabled: true # Признак включения подписания и проверки подписи
  validate_enabled: false # Признак предоставления дополнительного URL проверки
подписи
  algorithm: "GOST3410_2012_256" # Алгоритм формирования подписи
  serial_number: "" # Серийный номер ключа подписи
  alias: "" # Алиас ключа. Заполняется либо серийный номер, либо алиас
  notarius: # Блок настроек сервиса Notaris, используемый для подписания
      host: localhost # Хост, на котором доступен Notaris
      port: 8080 # Порт, на котором доступен Notaris
```

Параметры конфигурации

- `enable` - включение (true), отключение (false) подписи ответов и проверки подписей других источников;
- `validate_enabled` - признак предоставления дополнительного URL проверки подписи (доступность вызова REST-метода проверки подписи);
- `algorithm` - алгоритм формирования подписи. На текущий момент доступен только GOST3410_2012_256;
- `serial_number` - серийный номер ключа подписи;
- `alias` - алиас ключа, заполняется либо серийный номер, либо алиас;
- `notarius` - настройки подключения (host и port) к модулю криптографии notarius.

2.2.2.1.2 Конфигурация файла credentials.yml

```
version: 1.0.0
system:
    mnemonic: smev_ql_mnemonic
    instance: smev_ql_instance
```

Параметры конфигурации

- `version` - номер версии СМЭВ QL;
- `mnemonic` - мнемоника СМЭВ QL, по этому параметру осуществляется идентификация СМЭВ QL сервер во внешних системах и СМЭВ4;
- `instance` - наименование инстанса СМЭВ QL.

2.2.2.1.3 Общий сценарий выполнения

1. Администратор системы открывает на редактирование нужный файл (`credentials.yaml` и/или `application.yaml`) настроек СМЭВ QL сервер и меняет требуемые параметры.
2. Перезапускает приложение для применения новых настроек. Для этого открывает консоль утилиты работы со СМЭВ QL и выполняет команду:

```
./smevql restart
```

2.2.3 Настройка СМЭВ3-адаптера

2.2.3.1 Конфигурация СМЭВ3-адаптер (application.yml)

Файл `application.yml` – основной конфигурационный файл **СМЭВ3-адаптера**, в котором задана логика и порядок работы адаптера:

- получение входящих запросов, их обработка;
- настройка подключения к **СМЭВ** и FTP-серверу **СМЭВ3**, к **Prostore** через REST-запросы;
- настройка алгоритма формирования и проверки электронной подписи(ЭП) и т.д.

2.2.3.1.1 Пример файла application.yml

```yaml
vertx:
  #Настройки вертикса
  #тут можно указать все настройки из документации для vertx
  props:
    # метрики
    metricsOptions:
      enabled: true
      # тип метрик, например prometheusOptions | jmxMetricsOptions
      prometheusOptions:
        enabled: true
        startEmbeddedServer: true
        embeddedServerOptions:
          #порт для сервера с метриками
          port: 9033
  web-client:
    max-pool-size: 20

spring:
  liquibase:
    enabled: false
  main:
    allow-bean-definition-overriding: true

smev:
  #url смэва
  endpointUrl: http://localhost:7979/api/v1/soap/
  keystoreType: "DUMMY"
  keystoreFile: x
  keystorePass: x
  privateKeyAlias: x
  privateKeyPass: x
  certificateAlias: x
  signatureURI: "http://www.w3.org/2001/04/xmldsig-more#dummy"
  # алгоритм подписи
  signatureAlgorithm: "DUMMY"
  #метод подписи
  digestMethod: "http://www.w3.org/2001/04/xmldsig-more#dummy"
  #версия схемы смев
  #availiabe 1.2 and 1.3
  version: 1.3
  #верификация входящих сообщений
  incomingVerificationEnabled: false
  #подпись исходящих сообщений
  outgoingSigningEnabled: false
  #таймаут отправки сообщения в смев
  timeout: 30000
  #время между попытками перепосылки в смев
  retry-timeout: 30000
  #максимальный размер очереди, ожидающей отправки сообщений
  webMaxWaitQueueSize: -1
  #пул коннектов
  webMaxPoolSize: 20

receiver:
  # количество вертиклов
  instances: 1
  receiver-property:
    -
      #селектор из смэв
      selector:
```

```
        namespace: a
        root-element-name: b
      #пебл шаблон, который будет обрабатываться для определенного selector
      template: smev3-adapter/templates/smev.xml.peb
      #задержка между запросами, в случае если очередб пуста
      idle-delay: PT1m
      # файл, который будет отправлен в случае ошибки
      fallback-response: smev3-adapter/templates/fallback.xml
    -
      idle-delay: PT1m
      selector:
        namespace: urn://x-artefacts-testperson/1.0
        root-element-name: TestPersonRequest
      template: smev3-adapter/templates/smev.xml.peb

prostore-rest-client:
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9195}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}
  default-schema: demo_view

environment:
  name: ${ENVIRONMENT_NAME:test}

zookeeper:
  connection-string: ${ZOOKEEPER_DS_ADDRESS:t5-adsp-01.ru-central1.internal}
  retryPolicy:
    baseSleepTime: 1000
    maxRetries: 3
    maxSleepTime: 3000
  chroot: ${ZOOKEEPER_DS_CHROOT:/adapter}

migration:
  zk-enabled: ${MIGRATION_ZK_ENABLE:false}

paramstorage:
  base-path: '/smev/paramstorage'

deltastorage:
  base-path: '/smev/deltastorage'

sign:
  #алгоритм подписи файла
  digest-algorithm: 1.2.643.7.1.1.2.2

blob:
  # настройки подключения к BLOB адаптеру
  blob-source:
    host: 'localhost'
    port: 8080
    path:
  ftp-destination:
    #хост фтп смева
    host: localhost
    #порт фтп смева
    port: 21
    #корневой каталог
    #path: aaa/bbb/ccc
    #пользователь
    user: user
    #пароль
```

```
    password: 123

rest:
  #вкл/выкл
  enabled: false
  #порт на котором будет запущена рестовая ручка
  port: 8080
  # путь get запроса
  get: /le
  #путь post запроса
  post: /le
  #обрабатываемый шаблон
  template: smev3-adapter/templates/smev.xml.peb

#рассылка смев
scheduler:
  #вкл/выкл
  enabled: false
  #интервал между запусками
  interval: PT30s
  #обрабатываемый шаблон
  template: smev3-adapter/templates/pfr-delta.peb

pool:
  #корутин пул для обработки смев шаблонов
  reader-executor: 20
  #корутин пул для обработки  шедулера
  schedule-executor: 1
  logExecutor: 20

logging:
  level:
    root: info
    ru:
      rtlabs:
        smev:
          logging: trace
  request-response:
    smev-request: true
    smev-response: true

backup:
  zk-path: ${SMEV3_BACKUP_ZK_PATH:/${environment.name}/smev3-adapter}
  commandTopic: ${BACKUP_COMMAND_TOPIC:adapter.command}
  adapterCommandBroadcast:
${S3A_ADAPTER_COMMAND_BROADCAST_TOPIC:adapter.command.broadcast}
  backupTopic: ${BACKUP_TOPIC:adapter.backup}
  statusTopic: ${STATUS_TOPIC:adapter.status}
  timeout: ${BACKUP_TIMEOUT:PT180s}
  kafka:
    consumer:
      property:
        bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
        group.id: ${SMEV3_BACKUP_GROUP_ID:smev3-adapter_adapter_command}
        auto.offset.reset: latest
    producer:
      property:
        bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}

# Параметры подключения к сервису печатных форм. Указывается при использовании
функции toSpf
spf:
```

```
  host: ${SPF_HOST:localhost}
  port: ${SPF_PORT:8080}
  # Дополнительные параметры. Указываются ключ-значения сертификатов, необходимых для сервиса ПФ
  params: {}
```

2.2.3.2 Параметры конфигурации

Настройка конфигурации **СМЭВ3-адаптера** осуществляется путем редактирования параметров настроек в файле `application.yml`.

Обязательными параметрами для настройки **СМЭВ3-адаптера** являются секции: `smev`, `receiver`, `datasource`. Остальные параметры следует оставить без изменения и настраивать только для решения определенных бизнес-задач.

Pebble-шаблоны для настройки задаются в секциях: `receiver-property`, `rest` и `scheduler`.

Для каждого вида сведений, предоставляемых Витриной, следует создавать отдельное сопоставление `receiver` и устанавливать значения `receiver-property`. Остальные параметры следует оставить без изменения.

В файле конфигурации могут быть настроены следующие секции:

- `vertx` – настройка параметров фреймворка Vert.x (подробнее на сайте разработчиков: https://vertx.io/docs/).
- `spring` – подключение к фреймворку spring boot (используется для разработки);
- `smev` – настройки подключения к **СМЭВ3-адаптеру**;
- `receiver`- настройка взаимодействия СМЭВ-запросов с Peblle-шаблонами (для каждого `receiver` можно настроить количество `instance`);
- `prostore-api-client` - блок параметров конфигурирования взаимодействия с **ProStore**;
- `datasource` – параметры подключения к **ProStore**;
- `environment` - настройки окружения;
- `zookeeper` – параметры подключения к **Zookeeper**;
- `migration` – настройка параметров миграции сервисной базы данных **СМЭВ3-адаптера***\` в базу данных **Zookeeper**;
- `paramstorage` – указывается корневой путь хранилища параметров;
- `deltastorage` – указывается корневой путь хранилища дельт;
- `sign` - настройка формирования и проверки электронной подписи(ЭП) в SOAP-пакетах СМЭВ3;
- `blob` - интеграция с **BLOB-адаптер**;
- `rest` - настройки подключения для возможности выполнения rest-запросов к СМЭВ3-адаптеру и получения ответов на них;
- `scheduler` - настройка планировщика заданий (запуск дельт по расписанию);
- `pool` - размер прерываемого кода;
- `logging` – настраивается логирование работы модуля;
- `backup` - настройки бекапирования;
- `spf` - Параметры подключения к сервису печатных форм (Указывается при использовании функции toSpf).

2.2.3.2.1 Секция vertx

Секция `vertx` предназначена для настройки параметров фреймворка Vert.x (подробнее на сайте разработчиков: https://vertx.io/docs/). Для включения сбора метрик используйте следующий код:

```yaml
vertx:
  #Настройки вертикса
  #тут можно указать все настройки из документации для vertx
  props:
    # метрики
    metricsOptions:
      enabled: true
      # тип метрик, например prometheusOptions | jmxMetricsOptions
      prometheusOptions:
        enabled: true
        startEmbeddedServer: true
        embeddedServerOptions:
          #порт для сервера с метриками
          port: 9033
  web-client:
    max-pool-size: 20
```

2.2.3.2.2 Секция spring

Секция `spring` подключение к фреймворку spring boot (используется для разработки).

Например:

```yaml
spring:
  liquibase:
    enabled: false
  main:
    allow-bean-definition-overriding: true
```

2.2.3.2.3 Секция smev

Секция `smev` отвечает за настройки подключения к СМЭВ3.

Например:

```yaml
smev:
  endpointUrl: http://127.0.0.1:7979/api/v1/soap/
  keystoreType: "DUMMY"
  keystoreFile: x
  keystorePass: x
  privateKeyAlias: x
  privateKeyPass: x
  certificateAlias: x
  signatureURI: "http://www.w3.org/2001/04/xmldsig-more#dummy"
  signatureAlgorithm: "DUMMY"
  digestMethod: "http://www.w3.org/2001/04/xmldsig-more#dummy"
  incomingVerificationEnabled: false
  outgoingSigningEnabled: true
  #таймаут отправки сообщения в смев
  timeout: 30000
  #время между попытками перепосылки в смев
  retry-timeout: 30000
  #максимальный размер очереди, ожидающей отправки сообщений
  webMaxWaitQueueSize: -1
  #пул коннектов
  webMaxPoolSize: 20
```

В случае, когда СМЭВ3 не отвечает на запрос, в новой версии СМЭВ3-адаптера (секция `smev`), добавлена возможность, которая позволяет задать время ожидания ответа (`timeout`) перед повторной отправкой запроса к СМЭВ3:

- `timeout` - таймаут отправки сообщения в СМЭВ3;
- `retry-timeout` - время между повторной попыткой отправки запроса в СМЭВ3;
- `webMaxWaitQueueSize` - максимальный размер очереди, ожидающей отправки

сообщений;

- `webMaxPoolSize` - пул коннектов.

> Примечание:
>
> Для удобного отслеживания в лог-файлах всех запросов/ответов к СМЭВ3 в рамках одной бизнес-операции, в файл `logback-json.xml` добавлен параметр `ReqMessageID`. При обработке ошибки от СМЭВ3, в лог-файл добавляется описание ошибки и код ошибки СМЭВ3 (в том случае, если СМЭВ3 вернул данный код в блоке `description`).

2.2.3.2.4 Секция receiver

Секция `receiver` предназначена для настройки параметров взаимодействия СМЭВ-запросов с pebble-шаблонами.

Например:

```
receiver:
  receiver-property:
    -
      selector:
        namespace: a
        root-element-name: b
      template: templates/smev.xml.peb
      idle-delay: PT1m
      fallback-response: templates/fallback.xml
    -
      selector:
        namespace: urn://x-artefacts-testperson/1.0
        root-element-name: TestPersonRequest
      template: templates/smev.xml.peb
      idle-delay: PT1s
```

Параметры настроек

- `namespace` - пространство имен в XML.
- `root-element-name` - имя корневого элемента запроса обрабатываемого ВС (как указано в заявке на регистрацию ВС);
- `template` имя файла, содержащего pebble-шаблон обработки запросов для данного ВС;
- `idle-delay` - периодичность опроса очереди СМЭВ3 для получения новых запросов (в формате ISO 8601).
- `mtom-xop-postfix` - включение бинарных данных посредством ссылки `xop`, например:

```
<xop:Include xmlns:xop="http://www.w3.org/2004/08/xop/include" ref="cid:320038b2-0485-4658-89cc-980b7c8b5193@smev_client"/>
```

Пример файла smev.xml.peb

```
<TestPersonResponse xmlns="urn://x-artefacts-testperson/1.0">
{% set my_blob = fromblob ("/Picture_13.jpg", "my_fname", "my_mime") %}
<photo>{{ toftp ("some test 1", "file.txt", "text/plain") }}</photo>
<photo>{{ toftp ("some test 2", "file.txt", "text/plain") }}</photo>
<photo>{{ toftp ("some test 3", "file2.txt", "text/plain") }}</photo>
<photo>{{ tomtom (my_blob, my_mime) }}</photo>
</TestPersonResponse>
```

Пример файла fallback.xml (Ответ при ошибке обработки запроса)

```
<TestPersonResponse xmlns="urn://x-artefacts-testperson/1.0">
    <text>Произошла ошибка при обработке запроса: %error_message%</text>
</TestPersonResponse>
```

2.2.3.2.5 Секция prostore-rest-client

В секции `prostore-rest-client` реализован блок параметров конфигурирования взаимодействия с ProStore.

Например:

```
prostore-rest-client:
  host: ${PS_HOST:t5-prostore-01.ru-central1.internal}
  port: ${PS_PORT:9195}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}
```

Параметры настроек

- `host` - адрес Prostore, например `PS_HOST:t5-prostore-01.ru-central1.internal`;
- `port` - порт Prostore, например `PS_PORT:9195`;
- `max-pool-size` - максимальное число подключений к Prostore, например `PS_MAX_POOL_SIZE:8`.

2.2.3.2.6 Секция environment

В секции `environment` указывается среда разработки (dev, test, stable, prod)

Например:

```
environment:
  name: ${ENVIRONMENT_NAME:test}
```

Параметры настроек

- `name` - Название окружения, например `ENVIRONMENT_NAME:test`.

2.2.3.2.7 Секция zookeeper

Секция `zookeeper` предназначена для настройки параметров подключения к **Zookeeper**.

Например:

```
zookeeper:
  connection-string: ${ZOOKEEPER_DS_ADDRESS:t5-adsp-01.ru-central1.internal}
  retryPolicy:
    baseSleepTime: 1000
    maxRetries: 3
    maxSleepTime: 3000
  chroot: ${ZOOKEEPER_DS_CHROOT:/adapter}
```

Параметры настроек

- `connect-string` - адреса серверов для подключения к **Zookeeper** (разделитель - `,`);
- `baseSleepTime` - начальное значение таймаута ожидания при повторных запросах;
- `maxRetries` - максимальное количество повторных запросов;
- `maxSleepTime` - максимальное значение таймаута ожидания при повторных запросах.

2.2.3.2.8 Секция migration

Секция `migration` реализована настройка миграции зукипера для задачи бекапирования.

Например:

```
migration:
  zk-enabled: ${MIGRATION_ZK_ENABLE:false}
```

Параметры настроек

– `enabled` - подключение миграции, например `{MIGRATION_ENABLE:false}`.

2.2.3.2.9 Секция paramstorage

В секции `paramstorage` указывается корневой путь до сервера **Zookeeper** для механизма параметров (ключ-значение).

Например:

```
paramstorage:
  base-path: '/smev/paramstorage'
```

2.2.3.2.10 Секция deltastorage

В секции `deltastorage` указывается корневой путь до сервера **Zookeeper** для механизма дельт.

Например:

```
deltastorage:
  base-path: '/smev/deltastorage'
```

2.2.3.2.11 Секция sign

Секция `sign` предназначена для формирования и проверки электронной подписи (ЭП) в SOAP-пакетах СМЭВ3.

Например:

```
sign:
  digest-algorithm: 1.2.643.7.1.1.2.2
```

Параметры настроек

– `digest-algorithm` - алгоритм ключа проверки электронной подписи.

2.2.3.2.12 Секция blob

Секция `blob` предназначена для настройки взаимодействия модуля **СМЭВ3-адаптер** с:

– **BLOB-адаптером** для считывания BLOB-полей (см. Взаимодействие через СМЭВ3-адаптер);

– FTP-сервером СМЭВ3, на который модуль *СМЭВ3-адаптер* выгружает содержимое BLOB-полей и/или большие табличные данные.

Например:

```
blob:
  # настройки подключения к BLOB адаптеру
  blob-source:
    host: 'localhost'
    port: 8080
    path:
  ftp-destination:
    #хост фтп смева
    host: localhost
    #порт фтп смева
    port: 21
    #корневой каталог
    #path: aaa/bbb/ccc
    #пользователь
    user: user
    #пароль
    password: 123
```

Параметры настроек

- `blob-source` - настройка подключения к **BLOB-адаптеру**;
- `ftp-destination` - настройка подключения к FTP-серверу СМЭВ3.

2.2.3.2.13 Секция rest

Секция `rest` предназначена для настройки возможности выполнения REST-запросов к СМЭВ3-адаптеру и получения ответов на них.

Например:

```
rest:
  #вкл/выкл
  enabled: false
  #порт на котором будет запущена рестовая ручка
  port: 8080
  # путь get запроса
  get: /le
  #путь post запроса
  post: /le
  #обрабатываемый шаблон
  template: smev3-adapter/templates/smev.xml.peb
```

2.2.3.2.14 Секция scheduler

Секция `scheduler` предназначена для настройки планировщика заданий в случае, если планируется использовать механизм отправки дельт по расписанию.

Например:

```
scheduler:
  enabled: true
  interval: PT30s
  template: templates/pfr-delta.peb
```

Параметры настроек

- `enabled` - включение планировщика заданий;
- `interval` - интервал между отправкой дельт;
- `template` - путь к Pebble-шаблону;

2.2.3.2.15 Секция pool

В секции `pool` указывается размер прерываемого кода.

Например:

```
pool:
  reader-executor: 20
  schedule-executor: 1
  restExecutor: 1
  logExecutor: 20
```

2.2.3.2.16 Секция logging

В секции `logging` настраивается логирование работы модуля.

Например:

```
logging:
  request-response:
    smev-request: false
    smev-response: false
```

Параметры настроек

- `smev-request` - логирование запросов;
- `smev-response` - логирование ответов.

2.2.3.2.17 Секция backup

Секция `backup` предназначена для настроек бекапирования модуля.

Например:

```
backup:
  zk-path: ${SMEV3_BACKUP_ZK_PATH:/${environment.name}/smev3-adapter}
  commandTopic: ${BACKUP_COMMAND_TOPIC:adapter.command}
  adapterCommandBroadcast:
${S3A_ADAPTER_COMMAND_BROADCAST_TOPIC:adapter.command.broadcast}
  backupTopic: ${BACKUP_TOPIC:adapter.backup}
  statusTopic: ${STATUS_TOPIC:adapter.status}
  timeout: ${BACKUP_TIMEOUT:PT180s}
  kafka:
    consumer:
      property:
        bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
        group.id: ${SMEV3_BACKUP_GROUP_ID:smev3-adapter_adapter_command}
        auto.offset.reset: latest
    producer:
      property:
        bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
```

Параметры настроек

- `zk-path` - путь к корневой ноде zookeeper для бэкапирования, например `{COUNTER_BACKUP_ZK_PATH:/${environment.name}/counter-provider/counters}`;
- `commandTopic` - топик команд бэкапирования, например: `{BACKUP_COMMAND_TOPIC:adapter.command}`;
- `backupTopic` - топик для отправки забэкапированных данных, например: `{BACKUP_TOPIC:adapter.backup}`;
- `statusTopic` - топик для отправки статусов бэкапирования, например: `{STATUS_TOPIC:adapter.status}`.

2.2.3.2.18 Секция spf

В секции `spf` указываются параметры подключения к сервису печатных форм. Указывается при использовании функции toSpf.

Например:

```
spf:
  host: ${SPF_HOST:localhost}
  port: ${SPF_PORT:8080}
  # Дополнительные параметры. Указываются ключ-значения сертификатов, необходимых для сервиса ПФ
  params: {}
```

Параметры настроек

- `host` - адрес подключения, например `{SPF_HOST:localhost}`;
- `port` - порт подключения, например: `{SPF_PORT:8080}`;
- `params` - Дополнительные параметры. Указываются ключ-значения сертификатов, необходимых для сервиса ПФ.

2.2.4 Настройка CSV-Uploader

2.2.4.1 Конфигурация CSV-uploader (application.yml)

Файл `application.yml` – основной конфигурационный файл модуля **CSV-uploader**, в котором задана логика и порядок работы загрузчика, а также другие настройки необходимые

для корректной работы адаптера.

2.2.4.1.1 Пример файла application.yml

```yaml
# Kafka Prostore
  .kafkaUrl: &kafkaUrl ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}

http-server:
  # Порт для старта веб сервера
  port: ${HTTP_PORT:8080}
  # Включить веб-сервер
  enabled: ${HTTP_ENABLED:true}

send:
  # Размер отправляемой порции данных
  chunk-row-count: ${CHUNK_ROW_COUNT:1000}
  # Размер буфера на чтение файла
  file-buffer-size: ${FILE_BUFFER_SIZE:1048576}
  # Количество Job на чтение
  read-job-count: ${READ_JOB_COUNT:4}
  # Размер Channel для сериализации
  serialize-channel-size: ${SERIALIZE_CHANNEL_SIZE:20}
  # Количество Job на сериализацию
  serialize-job-count: ${SERIALIZE_JOB_COUNT:4}
  # Размер Job на отправку
  send-channel-size: ${SEND_CHANNEL_SIZE:20}
  # Количество Job на отправку
  send-job-count: ${SEND_JOB_COUNT:4}

file-size:
  # Ограничение на размер оправляемого файла (мегабайты)
  restriction: ${SEND_FILE_SIZE_RESTRICTION:1024}

logging.level:
  root: info
  ru.itone: debug

environment:
  # Название окружения
  name: ${ENVIRONMENT_NAME:test}
  # Папка для ошибочных файлов
  error-folder: ${ENVIRONMENT_ERROR_FOLDER:error}

zookeeper:
  # Адрес сервера zookeeper
  connection-string: ${ZK_CONNECTION:localhost}
  # Таймаут сессии
  session-timeout-ms: ${ZK_SESSION_TIMEOUT_MS:30000}
  # Таймаут подключения
  connection-timeout-ms: ${ZK_CONNECTION_TIMEOUT_MS:86400000}
  chroot: ${ZOOKEEPER_DS_CHROOT:/adapter}

migration:
  enabled: ${MIGRATION_ENABLE:false}

prostore-rest-client:
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9195}
  http:
      max-pool-size: ${PS_MAX_POOL_SIZE:8}

prostore:
```

```yaml
  zookeeper:
      # Адрес сервера zookeeper для загрузки данных в простор
      connection-string: ${ZK_PROSTORE_CONNECTION:locahost:2181}

validation:
  enable: ${VALIDATION_ENABLE:true}
  rest-uploader-url: ${REST_UPLOADER_URL:http://localhost:8081}
  # обязательность использования ФЛК
  mandator: ${VALIDATION_MANDATOR:false}

upload:
  # требуется токен для аутентификации на rest-uploader
  jwt-auth: ${JWT_AUTH:false}

kafka:
  create-topic:
    # Количество партиций на загрузку через EDML
    num-partitions: ${EDML_UPLOAD_NUM_PARTITIONS:1}
    # Фактор репликации при создании топика
    replication-factor: ${EDML_UPLOAD_REPLICATION_FACTOR:1}
  topic:
    # Топик для журналирования
    journal-log: journal.log
    flk-log: flk.log
  consumer:
    # Количество партиций на выгрузку через EDML
    num-partitions: ${EDML_DOWNLOAD_NUM_PARTITIONS:1}
    property:
    bootstrap.servers: *kafkaUrl
    group.id: csv-uploader
    auto.offset.reset: earliest
    enable.auto.commit: true
  producer:
    property:
      bootstrap.servers: *kafkaUrl

# Настройки парсера csv файлов
csv-parser:
  # Символ разделителя значений
  separator: ${CSV_PARSER_SEPARATOR:;}
  # Символ кавычки
  quote-char: ${CSV_PARSER_QUOTE_CHAR:"}
  # Символ экранирования значений
  escape-char: ${CSV_PARSER_ESCAPE_CHAR:'}
  # Настройка интерпретации значений как null. Допустимые значения:
  #  - EMPTY_SEPARATORS - пустое значение между двумя разделителями, например ;;
  #  - EMPTY_QUOTES - пустые кавычки, например ;"";
  #  - BOTH - оба варианта
  #  - NEITHER - никогда. Пустая строка всегда определяется как пустая строка
  field-as-null: ${CSV_PARSER_FIELD_AS_NULL:EMPTY_SEPARATORS}

metrics:
  port: ${METRICS_PORT:9837}

backup:
  zk-path: ${CSV_UPLOADER_BACKUP_ZK_PATH:/${environment.name}/csv-uploader/config}
  commandTopic: ${BACKUP_COMMAND_TOPIC:adapter.command}
  backupTopic: ${BACKUP_TOPIC:adapter.backup}
  statusTopic: ${STATUS_TOPIC:adapter.status}
  kafka:
    consumer:
      property:
```

```
      bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost}
      group.id: ${CSV_UPLOADER_BACKUP_GROUP_ID:csv_uploader_adapter_command}
      auto.offset.reset: latest
    producer:
      property:
        bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost}

jet-connector:
  use: ${JET_CONNECTOR_USE:true}
```

2.2.4.2 Параметры конфигурации

Настройка конфигурации **CSV-uploader** осуществляется путем редактирования параметров настроек в файле `application.yml`.

Некоторые настройки доступны для редактирования через пользовательский интерфейс модуля, например, Настройка отображения количества записей и Запуск по расписанию в *Журнале операций* .

В файле конфигурации **CSV-uploader** могут быть настроены следующие секции:

- `kafkaUrl` - URL для доступа к **Kafka**;
- `http-server` - настройки порта подключения;
- `send` - настройка отправки файлов;
- `file-size` - ограничение на размер отправляемого файла (мегабайты);
- `logging.level` - настройка сохранения лог-файла;
- `environment` - определяет значение среды разработки;
- `zookeeper` - настройка подключения **Zookeeper**;
- `migration` - настройка миграции зукипера для задачи бекапирования;
- `prostore-rest-client` - блок параметров конфигурирования взаимодействия с **ProStore**;
- `prostore` - адрес сервера zookeeper для загрузки данных в **ProStore**;
- `validation` - включение/выключение механизма валидации загрузки с помощью модуля **REST-Uploader**;
- `upload` - требование токена для аутентификации на **REST-Uploader**;
- `kafka` - настройка подключения к шине данных **Apache Kafka**;
- `csv-parser` - настройка парсинга CSV;
- `metrics` - настройка получения метрик;
- `backup` - настройка бэкапирования модуля;
- `jet-connector` - подготовлен для оптимизации задержек записи.

2.2.4.2.1 Секция kafkaUrl

В секция `kafkaUrl` указывается URL-адрес для доступа к Apache Kafka (ProStore).

Например:

```
.kafkaUrl: &kafkaUrl ${KAFKA_BOOTSTRAP_SERVERS:dev-dtm-one05.ru-central1.internal:9092}
```

Параметры конфигурации

`KAFKA_BOOTSTRAP_SERVERS` - URL-адрес для доступа к Apache Kafka (ProStore).

2.2.4.2.2 Секция http-server

Секция `http-server` предназначена для настройки порта и протокола передачи данных (одно из значений `http` или `https`).

Например:

```
http:
        port: ${HTTP_PORT:8080}
        enabled: ${HTTP_ENABLED:true}
```

Параметры конфигурации

- `port` - порт для старта веб-сервера, например `HTTP_PORT:8080`;
- `enabled` - статус включения/отключения веб-сервера, например `HTTP_ENABLED:true`.

2.2.4.2.3 Секция send

В секции `send` настраивается отправка файлов.

Например:

```
send:
        chunk-row-count: ${CHUNK_ROW_COUNT:1000}
        file-buffer-size: ${FILE_BUFFER_SIZE:1048576}
        read-job-count: ${READ_JOB_COUNT:4}
        serialize-channel-size: ${SERIALIZE_CHANNEL_SIZE:20}
        serialize-job-count: ${SERIALIZE_JOB_COUNT:4}
        send-channel-size: ${SEND_CHANNEL_SIZE:20}
        send-job-count: ${SEND_JOB_COUNT:4}
```

Параметры конфигурации:

- `chunk-row-count` - размер отправляемой порции данных, например `CHUNK_ROW_COUNT:100`;
- `file-buffer-size` - размер буфера на чтение файла, например `FILE_BUFFER_SIZE:1048576`;
- `read-job-count` - количество Job на чтение, например `READ_JOB_COUNT:4`;
- `serialize-channel-size` - размер `Channel` для сериализации, например `SERIALIZE_CHANNEL_SIZE:20`;
- `serialize-job-count` - количество задач на сериализацию, например `SERIALIZE_JOB_COUNT:4`;
- `send-channel-size` - размер задач на отправку, например `SEND_CHANNEL_SIZE:20`;
- `send-job-count` - количество задач на отправку, например `SEND_JOB_COUNT:4`;

2.2.4.2.4 Секция file-size

Секция `file-size` отвечает за ограничение на размер отправляемого файла (указывется в мегабайтах).

```
file-size:
        #
        restriction: ${SEND_FILE_SIZE_RESTRICTION:1024}
```

Параметры конфигурации

- `restriction` - ограничение на размер отправляемого файла, например `SEND_FILE_SIZE_RESTRICTION:1024`.

2.2.4.2.5 Секция logging.level

В секции `logging.level` настраиваются записи логирования.

Например:

```
logging.level:
        root: info
        ru.itone: debug
```

2.2.4.2.6 Секция environment

В секции `environment` выбирается среда разработки (например, значение `test`, `prod` и т.д).

Например:

```
environment:
        name: ${ENVIRONMENT_NAME:test}
        error-folder: ${ENVIRONMENT_ERROR_FOLDER:error}
```

Параметры конфигурации

- `name` - название окружения, например `ENVIRONMENT_NAME:test`;
- `error-folder` - папка для ошибочных файлов, например `ENVIRONMENT_ERROR_FOLDER:error`.

2.2.4.2.7 Секция zookeeper

В секции `zookeeper` настраиваются параметры подключения к серверу **Zookeeper**.

Например:

```
zookeeper:
        # Адрес сервера zookeeper
        connection-string: ${ZK_CONNECTION:localhost}
        # Таймаут сессии
        session-timeout-ms: ${ZK_SESSION_TIMEOUT_MS:30000}
        # Таймаут подключения
        connection-timeout-ms: ${ZK_CONNECTION_TIMEOUT_MS:86400000}
        chroot: ${ZOOKEEPER_DS_CHROOT:/adapter}
```

Параметры конфигурации

- `connection-string` - адрес сервера **Zookeeper**, например `ZK_CONNECTION:localhost`;
- `session-timeout-ms` - таймаут сессии, например `ZK_SESSION_TIMEOUT_MS:30000`;
- `connection-timeout-ms` - таймаут подключения, например `ZK_CONNECTION_TIMEOUT_MS:86400000`;
- `chroot` - Zookeeper DS chroot path, например `ZOOKEEPER_DS_CHROOT:/adapter`.

2.2.4.2.8 Секция migration

В секции `migration` реализована настройка миграции зукипера для задачи бекапирования.

Например:

```
migration:
        enabled: ${MIGRATION_ENABLE:false}
```

Параметры настроек

- `enabled` - включение миграции (по умолчанию выключена), например `{MIGRATION_ENABLE:false}`.

2.2.4.2.9 Секция prostore-rest-client

В секции `prostore-rest-client` реализован блок параметров конфигурирования взаимодействия с ProStore.

Например:

```
prostore-rest-client:
        host: ${PS_HOST:localhost}
        port: ${PS_PORT:9195}
        http:
                max-pool-size: ${PS_MAX_POOL_SIZE:8}
```

Параметры настроек

- `host` - адрес Prostore, например `PS_HOST:localhost`;
- `port` - порт Prostore, например `PS_PORT:9195`;
- `max-pool-size` - максимальное число подключений к Prostore, например `PS_MAX_POOL_SIZE:8`.

2.2.4.2.10 Секция prostore

В секции `prostore` указывается адрес сервера zookeeper для загрузки данных в Prostore. Например:

```
prostore:
       zookeeper:
              connection-string: ${ZK_PROSTORE_CONNECTION:locahost:2181}
```

Параметры настроек

- `connection-string` - адрес сервера **Zookeeper** для загрузки данных в Prostore, например `ZK_PROSTORE_CONNECTION:locahost:2181`.

2.2.4.2.11 Секция validation

В секции `validation` реализован механизм настройки валидации ФЛК.
Например:

```
enable: ${VALIDATION_ENABLE:true}
rest-uploader-url: ${REST_UPLOADER_URL:http://localhost:8081}
mandator: ${VALIDATION_MANDATOR:false}
```

Параметры конфигурации

- `enable` - валидация включена (по умолчанию), например `{VALIDATION_ENABLE:true}`;
- `rest-uploader-url` - URL к сервису rest-uploader для выполнения валидации, например `{REST_UPLOADER_URL:http://localhost:8081}`;
- `mandator` - флаг использования ФЛК, например `{VALIDATION_MANDATOR:false}`.

2.2.4.2.12 Секция upload

В секции `upload` реализована настройка требования токена для аутентификации на REST-Uploader (если `true`, то при переключении на вкладку **Загрузка** появляется модальное окно для задания токена в текстовом виде и кнопка **Сохранить**).

Например:

```
upload:
       jwt-auth: ${JWT_AUTH:false}
```

Параметры конфигурации

- `jwt-auth` - требование токена для аутентификации на REST-Uploader, например `{JWT_AUTH:false}`.

2.2.4.2.13 Секция kafka

В секции `kafka` настраиваются параметры подключения к шине данных **Apache Kafka**.
Например:

```
kafka:
        create-topic:
                num-partitions: ${EDML_UPLOAD_NUM_PARTITIONS:1}
                replication-factor: ${EDML_UPLOAD_REPLICATION_FACTOR:1}
        topic:
                journal-log: journal.log
        consumer:
                num-partitions: ${EDML_DOWNLOAD_NUM_PARTITIONS:1}
                property:
                        bootstrap.servers: *kafkaUrl
                        group.id: csv-uploader
                        auto.offset.reset: earliest
                        enable.auto.commit: true
        producer:
                property:
                        bootstrap.servers: *kafkaUrl
```

Параметры конфигурации

- `num-partitions` - количество партиций на загрузку через EDML, например `EDML_UPLOAD_NUM_PARTITIONS:1`;
- `replication-factor` - фактор репликации при создании топика, например `EDML_UPLOAD_REPLICATION_FACTOR:1`.

2.2.4.2.14 Секция csv-parser

> **Внимание:**
>
> При загрузке файлов с форматно-логическим контролем, важно, чтобы настройки секции csv-parser были одинаковыми в модулях CSV-Uploader(если используется его UI),REST-Uploader и DATA-Uploader.

Секция `csv-parser` - настройка парсинга CSV.

Например:

```
csv-parser:
        separator: ${CSV_PARSER_SEPARATOR:;}
        quote-char: ${CSV_PARSER_QUOTE_CHAR:"}
        escape-char: ${CSV_PARSER_ESCAPE_CHAR:'}
        field-as-null: ${CSV_PARSER_FIELD_AS_NULL:EMPTY_SEPARATORS}
```

Параметры конфигурации

- `separator` - Символ разделителя значений, например `CSV_PARSER_SEPARATOR:;`;
- `quote-char` - символ кавычки, например `CSV_PARSER_QUOTE_CHAR:"`;
- `escape-char` - Символ экранирования значений, например `CSV_PARSER_ESCAPE_CHAR:'`;

  Настройка интерпретации значений как null. Допустимые значения:
  - EMPTY_SEPARATORS - пустое значение между двумя разделителями, например ;;
  - EMPTY_QUOTES - пустые кавычки, например ;»»;
  - BOTH - оба варианта
  - NEITHER - никогда. Пустая строка всегда определяется как пустая строка
- `field-as-null` - способ определения null поля, например `CSV_PARSER_FIELD_AS_NULL:EMPTY_SEPARATORS`.

Дополнительное описание параметров

1. Параметр `CSV_PARSER_ESCAPE_CHAR` работает следующим образом: если символ экранирования и символ кавычки равны `"`, то будет использован **RFC4180Parser**,

который считывает все символы между двумя двойными кавычками, при этом двойная кавычка в тексте поля должна быть экранирована двойной кавычкой (Например `"поле, ""содержащее двойную кавычку"""` будет считано как `поле, "содержащее двойную кавычку"`). В противном случае будет использован **CSVParser**, использующий символ экранирования для обозначения «непечатаемых символов».

2. Параметр `CSV_PARSER_FIELD_AS_NULL` может принимать следующие значения:
   - `EMPTY_SEPARATORS` - два разделителя полей (см. csv-parser/separator) подряд считаются null. Например: строка [aaa,,ccc] содержит значения [«aaa», null, «bbb»], а строка [aaa,»»,ccc] содержит значения [«aaa», «», «bbb»].
   - `EMPTY_QUOTES` - два «ограничителя строки» (см. csv-parser/escape-char) подряд считаются null. Например: строка [aaa,»»,ccc] содержит значения [«aaa», null, «bbb»], а строка [aaa,,ccc] содержит значения [«aaa», «», «bbb»].
   - `BOTH` - оба варианта (см. `EMPTY_SEPARATORS` и `EMPTY_QUOTES`) считаются null. Например: обе строки [aaa,»»,ccc] и [aaa,,bbb] содержат одинаковое значение [«aaa», null, «bbb»].
   - `NEITHER` - ни один из вариантов (см. `EMPTY_SEPARATORS` и `EMPTY_QUOTES`) не считается null. Например: обе строки [aaa,»»,ccc] и [aaa,,bbb] содержат одинаковое значение [«aaa», «», «bbb»].

2.2.4.2.15 Секция metrics

Секция `metrics` предназначена для настройки параметров метрик.
Например:

```
metrics:
  port: ${METRICS_PORT:9837}
```

Параметры конфигурации

- `port` - Порт для метрик, например `METRICS_PORT:9837`.

2.2.4.2.16 Секция backup

Секция `backup` предназначена для настроек бекапирования модуля.
Например:

```
backup:
  zk-path: ${COUNTER_BACKUP_ZK_PATH:/${environment.name}/counter-provider/counters}
  commandTopic: ${BACKUP_COMMAND_TOPIC:adapter.command}
  backupTopic: ${BACKUP_TOPIC:adapter.backup}
  statusTopic: ${STATUS_TOPIC:adapter.status}
  kafka:
    consumer:
      property:
        bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
        group.id: ${COUNTER_BACKUP_GROUP_ID:counter_provider_adapter_command}
        auto.offset.reset: latest
    producer:
      property:
        bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
```

Параметры настроек

- `zk-path` - путь к корневой ноде zookeeper для бэкапирования, например `{COUNTER_BACKUP_ZK_PATH:/${environment.name}/counter-provider/counters}`;
- `commandTopic` - топик команд бэкапирования, например: `{BACKUP_COMMAND_TOPIC:adapter.command}`;
- `backupTopic` - топик для отправки забэкапированных данных, например:

{BACKUP_TOPIC:adapter.backup};

– statusTopic - топик для отправки статусов бэкапирования, например: {STATUS_TOPIC:adapter.status}.

2.2.4.2.17 Секция jet-connector

Секция jet-connector предназначена для оптимизации задержек записи данных. Например:

```
jet-connector:
use: ${JET_CONNECTOR_USE:false}
```

Параметры настроек

– use - флаг активации jet-connector, например {JET_CONNECTOR_USE:false};

Таблица 2.13 Область применения

Режим/СУБД	ADB	ADP	ADQM	ADG
Чтение	Нет	Нет	Нет	Нет
Запись	В перспективе	Да	Нет	Нет

Чтобы воспользоваться **Jet-коннектор** требуется вместо upload external table создавать readable external table, указывающую на топик, как отражено в документации Prostore

В модуль MPPW не передается информация о том, в какую БД физически будут загружаться данные, синтаксис Простора един для всех поддерживаемых СУБД. Конкретная СУБД указывается в настройках Простора.

Даже если загрузка данных выполняется в более чем одну базу, на работе адаптеров это не сказывается.

При формировании запросов с табличными параметрами, в том числе при регистрации РЗ, необходимо явно перечислять поля таблиц, по которым будет выполняться фильтрация записей или объединение таблиц.

Переключение с **jet-connector** на **kafka postgres writer** и наоборот допускается при завершенных операциях загрузки данных.

Предупреждение:

Jet-коннектор в настоящее время применим для ADP, что делает его применимым, только для исталляций одной единственной СУБД ADP. Это ограничение остается на уровне документации при использовании **Jet-коннектор** с другими базами должна появиться ошибка.

2.2.5 Настройка СМЭВ4-адаптера - Модуль исполнения запросов

2.2.5.1 Конфигурация СМЭВ4-адаптера - Модуль исполнения запросов (application.yml)

Файл application.yml – основной конфигурационный файл **СМЭВ4-адаптера - Модуль исполнения запросов**, в котором задана логика и порядок работы адаптера:

– получение и обработка входящих запросов;

– подключение к Сервису формирования документов (секция: printable-forms-service);

– настройки логирования (секция: logging), а также другие настройки необходимые для корректной работы адаптера.

Хинт пагинации FORCE_LLR определен в переменных среды.

2.2.5.1.1 Пример файла application.yml

В конфигурационном файле задаются настройки, которые необходимы для решения текущих бизнес-задач.

```
http-server:
  port: ${HTTP_PORT:8090}

environment:
  name: ${ENVIRONMENT_NAME:test}

executor:
  reader-pool-size: ${EXECUTOR_READER_POOL_SIZE:20}
  max-execute-time: ${EXECUTOR_MAX_EXECUTE_TIME:600}
  log-pool-size: ${EXECUTOR_LOG_POOL_SIZE:20}

send:
  channel-size: ${SEND_CHANNEL_SIZE:1}
  compress: ${SEND_COMPRESS:none}
  max-message-size: ${SEND_MAX_MESSAGE_SIZE:800000}

query:
  data-source-type:
    for-listagg: ${DATA_SOURCE_TYPE_LISTAGG:ADP}
    statistics-request: ${DATA_SOURCE_TYPE_STATISTIC:ADP}
  force-llr-for-order: ${FORCE_LLR_FOR_ORDER:true}
  force-llr-for-all: ${FORCE_LLR_FOR_ALL:false}
  llr-rows-limit: ${LLR_ROWS_LIMIT:200}
  fetch-size: ${FETCH_SIZE:1000}

zookeeper:
  connection-string: ${ZOOKEEPER_DS_ADDRESS:localhost}
  connection-timeout-ms: ${ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000}
  session-timeout-ms: ${ZOOKEEPER_DS_SESSION_TIMEOUT_MS:86400000}
  chroot: ${ZOOKEEPER_DS_CHROOT:/adapter}

prostore-rest-client:
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9195}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}

printable-forms-service:
  host: ${PFS_HOST:localhost}
  port: ${PFS_PORT:8080}
  pool-size: ${PFS_POOL_SIZE:10}
  timeout: ${PFS_TIMEOUT:30}

kafka:
  agent.topic.prefix: ${AGENT_TOPIC_PREFIX:}
  max-concurrent-handle: ${KAFKA_MAX_CONCURRENT_HANDLE:1000}
  commit-interval: ${KAFKA_COMMIT_INTERVAL:5s}
  external:
    bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
    topic.prefix: ${EXTERNAL_TOPIC_PREFIX:${agent.topic.prefix}}
  internal:
    bootstrap.servers: ${PS_KAFKA:localhost:9092}
    topic.prefix: ${INTERNAL_TOPIC_PREFIX:${agent.topic.prefix}}
  consumer:
    query-request:
      topic: ${kafka.external.topic.prefix}query.rq
      max-concurrent-handle: ${kafka.max-concurrent-handle}
```

```
    commit-interval: ${kafka.commit-interval}
    property:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
      group.id: ${kafka.external.topic.prefix}query.consumer
      auto.offset.reset: earliest
      enable.auto.commit: false
  query-cancel-request:
    topic: ${kafka.external.topic.prefix}cancel.rq
    commit-interval: ${kafka.commit-interval}
    property:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
      group.id: ${kafka.external.topic.prefix}cancel.query.consumer
      auto.offset.reset: earliest
      enable.auto.commit: false
  metadata-request:
    topic: ${kafka.external.topic.prefix}metadata.rq
    commit-interval: ${kafka.commit-interval}
    property:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
      group.id: ${kafka.external.topic.prefix}metadata.consumer
      auto.offset.reset: earliest
      enable.auto.commit: false
  metadata-new-data-request:
    topic: ${kafka.external.topic.prefix}metadata.newdata.rq
    commit-interval: ${kafka.commit-interval}
    property:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
      group.id: ${kafka.external.topic.prefix}metadata.newdata.consumer
      auto.offset.reset: earliest
      enable.auto.commit: false
  statistics-request:
    topic: ${kafka.external.topic.prefix}statistics.rq
    commit-interval: ${kafka.commit-interval}
    property:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
      group.id: ${kafka.external.topic.prefix}statistics.rq.consumer
      auto.offset.reset: earliest
      enable.auto.commit: false
  report-request:
    topic: ${kafka.external.topic.prefix}procedure.query.rq
    max-concurrent-handle: ${kafka.max-concurrent-handle}
    commit-interval: ${kafka.commit-interval}
    property:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
      group.id: ${kafka.external.topic.prefix}report.rq.consumer
      auto.offset.reset: earliest
      enable.auto.commit: false
producer:
  query-result: ${kafka.external.topic.prefix}query.rs
  query-error: ${kafka.external.topic.prefix}query.err
  query-estimation-result: ${kafka.external.topic.prefix}query.estimation.rs
  query-cancel-result: ${kafka.external.topic.prefix}cancel.rs
  query-cancel-error: ${kafka.external.topic.prefix}cancel.err
  metadata-result: ${kafka.external.topic.prefix}metadata.rs
  metadata-error: ${kafka.external.topic.prefix}metadata.err
  metadata-newdata-result: ${kafka.external.topic.prefix}metadata.newdata.rs
  metadata-newdata-error: ${kafka.external.topic.prefix}metadata.newdata.err
  statistics-result: ${kafka.external.topic.prefix}statistics.rs
  statistics-error: ${kafka.external.topic.prefix}statistics.err
  report-result: ${kafka.external.topic.prefix}query.rs
  report-error: ${kafka.external.topic.prefix}query.err
  property:
```

```
      bootstrap.servers: ${kafka.external.bootstrap.servers}
    internal:
      mppr-query-request: ${kafka.internal.topic.prefix}mppr.delegate.rq
      tp-delete-tmp: ${kafka.internal.topic.prefix}tp.delete.tmp
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}

statistics:
  enabled: ${STATISTICS_ENABLED:false}
  timeout-min: ${STATISTICS_TIMEOUT_MIN:60}
  datamarts:
    - name: demo_dev
      tables:
        - name: all_types
          columns:
            - varchar_c
            - char_c
            - bigint_c

logging:
  request-response:
    query-request: ${QUERY_REQUEST_LOG_ENABLED:false}
    query-response: ${QUERY_RESPONSE_LOG_ENABLED:false}
    pf-request: ${PF_REQUEST_LOG_ENABLED:false}
    pf-response: ${PF_RESPONSE_LOG_ENABLED:false}

metrics:
  port: ${METRICS_PORT:9837}
```

2.2.5.2 Параметры конфигурации

Настройка конфигурации **СМЭВ4-адаптера - Модуль исполнения запросов** осуществляется путем редактирования параметров настроек в файле `application.yml`, где настраиваются секции:

- `http-server` - указывается порт для подключения;
- `environment` - указывается название окружения (`test`, `prod` и т.д.);
- `executor` - настраивается размер пула для запросов;
- `send` - настраиваются ограничения на размер загружаемого файла;
- `query` - настройка выполнения запросов;
- `zookeeper` - подключения в Zookeeper;
- `prostore-rest-client` - блок параметров конфигурирования взаимодействия с ProStore;
- `prostore`- указываются настройки подключения к ProStore;
- `printable-forms-service` - настройки подключения к Сервис формирования документов;
- `kafka` - настройки параметров подключения к шине данных Apache Kafka;
- `statistics` - управление статистикой;
- `logging` - настройка сохранения лог-файла;
- `metrics` - настройка получения метрик.

2.2.5.2.1 Секция http-server

В секции `http-server` указывается порт веб-сервера.

Например:

```
http:
  port: ${HTTP_PORT:8090}
```

Параметры настроек

- `port` - порт веб-сервера, например: `HTTP_PORT:8090`.

2.2.5.2.2 Секция environment

Секция `environment` предназначена для настройки параметров окружения.

Например:

```
environment:
  name: ${ENVIRONMENT_NAME:test}
```

Параметры настроек

- `name` - название окружения (test, prod и т.д.), например: `ENVIRONMENT_NAME:test`.

2.2.5.2.3 Секция executor

Секция `executor` предназначена для указания размера пула для чтения **Kafka** и времени выполнения задач.

Например:

```
executor:
  reader-pool-size: ${EXECUTOR_READER_POOL_SIZE:20}
  max-execute-time: ${EXECUTOR_MAX_EXECUTE_TIME:600}
  log-pool-size: ${EXECUTOR_LOG_POOL_SIZE:20}
```

Параметры настроек

- `reader-pool-size` - размер пула для чтения **Kafka**, например `EXECUTOR_READER_POOL_SIZE:20`;
- `max-execute-time` - максимальное время выполнения задачи (сек), например `EXECUTOR_MAX_EXECUTE_TIME:600`;
- `log-pool-size` - размер используемого пула для журналирования запросов и ответов, например `EXECUTOR_LOG_POOL_SIZE:20`.

2.2.5.2.4 Секция send

В секции `send` настраиваются ограничения на размер загружаемого файла.

Например:

```
send:
  channel-size: ${SEND_CHANNEL_SIZE:1}
  compress: ${SEND_COMPRESS:none}
  max-message-size: ${SEND_MAX_MESSAGE_SIZE:800000}
```

Параметры настроек

- `channel-size` - размер канала на отправку сообщения, например `SEND_CHANNEL_SIZE:10`;
- `compress` - сжатие выгружаемых сообщений (none или zstd), например `SEND_COMPRESS:none`;
- `max-message-size` - максимальный размер отправляемого сообщения, например `SEND_MAX_MESSAGE_SIZE:800000`.

2.2.5.2.5 Секция query

В секции `query` выполняется настройка выполнения запросов.

Например:

```
query:
  data-source-type:
    for-listagg: ${DATA_SOURCE_TYPE_LISTAGG:ADP}
    statistics-request: ${DATA_SOURCE_TYPE_STATISTIC:ADP}
  force-llr-for-order: ${FORCE_LLR_FOR_ORDER:true}
  force-llr-for-all: ${FORCE_LLR_FOR_ALL:false}
  llr-rows-limit: ${LLR_ROWS_LIMIT:200}
  fetch-size: ${FETCH_SIZE:1000}
```

Параметры настроек

- `data-source-type` - выполнение запроса с LISTAGG на (ADB/ADP), например `DATA_SOURCE_TYPE:ADB`;
- `force-llr-for-order` - выполнение ORDER BY запроса с использованием пагинации, например `FORCE_LLR_FOR_ORDER:true`;
- `force-llr-for-all` - выполнение всех запросов через LLR, например `FORCE_LLR_FOR_ALL:false`;
- `llr-rows-limit` - ограничение выгрузки через ЛЛР, при использовании в запросе лимита со значением меньшим, чем указанное значение, например `LLR_ROWS_LIMIT:200`, будет использован режим LLR;
- `fetch-size` - размер выгрузки через JDBC, например `FETCH_SIZE:1000`.

Внимание:

Для опции `force-llr-for-order` параметр `false` можно устанавливать только при развертывании витрины на единственной БД ADP.

2.2.5.2.6 Секция zookeeper

Секция `zookeeper` определяет настройки подключения к Zookeeper DS.

Например:

```
zookeeper:
  connection-string: ${ZOOKEEPER_DS_ADDRESS:t5-adsp-01.ru-central1.internal}
  connection-timeout-ms: ${ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000}
  session-timeout-ms: ${ZOOKEEPER_DS_SESSION_TIMEOUT_MS:86400000}
  chroot: ${ZOOKEEPER_DS_CHROOT:/adapter}
```

Параметры настроек

- `connection-string` - подключение в Zookeeper DS, например `ZOOKEEPER_DS_ADDRESS:t5-adsp-01.ru-central1.internal`;
- `connection-timeout-ms` - Zookeeper DS таймаут подключения, например `ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000`;
- `session-timeout-ms` - Zookeeper DS таймаут сессии, например `ZOOKEEPER_DS_SESSION_TIMEOUT_MS:86400000`;
- `chroot` - Zookeeper DS chroot path, например `ZOOKEEPER_DS_CHROOT:/adapter`.

2.2.5.2.7 Секция prostore-rest-client

В секции `prostore-rest-client` реализован блок параметров конфигурирования взаимодействия с **ProStore**.

Например:

```
prostore-rest-client:
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9195}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}
```

Параметры настроек

- `host` - адрес Prostore, например `PS_HOST:localhost`;
- `port` - порт Prostore, например `PS_PORT:9195`;
- `max-pool-size` - максимальное число подключений к Prostore, например `PS_MAX_POOL_SIZE:8`.

2.2.5.2.8 Секция printable-forms-service

Секция `printable-forms-service` определяет настройки подключения к Сервис формирования документов.

Например:

```
printable-forms-service:
  host: ${PFS_HOST:localhost}
  port: ${PFS_PORT:8080}
  pool-size: ${PFS_POOL_SIZE:10}
  timeout: ${PFS_TIMEOUT:30}
```

Параметры настроек

- `host` - адрес сервера формирования документов, например `PFS_HOST:localhost`;
- `port` - порт сервера формирования документов, например `PFS_PORT:8080`;
- `pool-size` - размер пула соединений для ПФ, например `PFS_POOL_SIZE:10`;
- `timeout` - таймаут переподключения к сервису формирования документов (секунды), например `PFS_TIMEOUT:30`.

2.2.5.2.9 Секция kafka

В секции `kafka` настраиваются параметры подключения к шине данных **Apache Kafka**.

Например:

```
kafka:
  agent.topic.prefix: ${AGENT_TOPIC_PREFIX:}
  max-concurrent-handle: ${KAFKA_MAX_CONCURRENT_HANDLE:1000}
  commit-interval: ${KAFKA_COMMIT_INTERVAL:5s}
  external:
    bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
    topic.prefix: ${EXTERNAL_TOPIC_PREFIX:${agent.topic.prefix}}
  internal:
    bootstrap.servers: ${PS_KAFKA:localhost:9092}
    topic.prefix: ${INTERNAL_TOPIC_PREFIX:${agent.topic.prefix}}
  consumer:
    query-request:
      topic: ${kafka.external.topic.prefix}query.rq
      max-concurrent-handle: ${kafka.max-concurrent-handle}
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.external.bootstrap.servers}
        group.id: ${kafka.external.topic.prefix}query.consumer
        auto.offset.reset: earliest
        enable.auto.commit: false
    query-cancel-request:
      topic: ${kafka.external.topic.prefix}cancel.rq
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.external.bootstrap.servers}
        group.id: ${kafka.external.topic.prefix}cancel.query.consumer
        auto.offset.reset: earliest
        enable.auto.commit: false
    metadata-request:
      topic: ${kafka.external.topic.prefix}metadata.rq
```

```
    commit-interval: ${kafka.commit-interval}
    property:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
      group.id: ${kafka.external.topic.prefix}metadata.consumer
      auto.offset.reset: earliest
      enable.auto.commit: false
  metadata-new-data-request:
    topic: ${kafka.external.topic.prefix}metadata.newdata.rq
    commit-interval: ${kafka.commit-interval}
    property:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
      group.id: ${kafka.external.topic.prefix}metadata.newdata.consumer
      auto.offset.reset: earliest
      enable.auto.commit: false
  statistics-request:
    topic: ${kafka.external.topic.prefix}statistics.rq
    commit-interval: ${kafka.commit-interval}
    property:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
      group.id: ${kafka.external.topic.prefix}statistics.rq.consumer
      auto.offset.reset: earliest
      enable.auto.commit: false
  report-request:
    topic: ${kafka.external.topic.prefix}procedure.query.rq
    max-concurrent-handle: ${kafka.max-concurrent-handle}
    commit-interval: ${kafka.commit-interval}
    property:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
      group.id: ${kafka.external.topic.prefix}report.rq.consumer
      auto.offset.reset: earliest
      enable.auto.commit: false
producer:
  query-result: ${kafka.external.topic.prefix}query.rs
  query-error: ${kafka.external.topic.prefix}query.err
  query-estimation-result: ${kafka.external.topic.prefix}}query.estimation.rs
  query-cancel-result: ${kafka.external.topic.prefix}cancel.rs
  query-cancel-error: ${kafka.external.topic.prefix}cancel.err
  metadata-result: ${kafka.external.topic.prefix}metadata.rs
  metadata-error: ${kafka.external.topic.prefix}metadata.err
  metadata-newdata-result: ${kafka.external.topic.prefix}metadata.newdata.rs
  metadata-newdata-error: ${kafka.external.topic.prefix}metadata.newdata.err
  statistics-result: ${kafka.external.topic.prefix}statistics.rs
  statistics-error: ${kafka.external.topic.prefix}statistics.err
  report-result: ${kafka.external.topic.prefix}query.rs
  report-error: ${kafka.external.topic.prefix}query.err
  property:
    bootstrap.servers: ${kafka.external.bootstrap.servers}
  internal:
    mppr-query-request: ${kafka.internal.topic.prefix}mppr.delegate.rq
    tp-delete-tmp: ${kafka.internal.topic.prefix}tp.delete.tmp
    property:
      bootstrap.servers: ${kafka.internal.bootstrap.servers}
```

Параметры конфигурации

– `topic` - префикс для топиков Агента СМЭВ4, например `AGENT_TOPIC_PREFIX`.

2.2.5.2.10 Секция statistics

Секция `statistics` предназначена для управления статистикой.

Например:

```
statistics:
  enabled: ${STATISTICS_ENABLED:false}
  timeout-min: ${STATISTICS_TIMEOUT_MIN:60}
  datamarts:
    - name: demo_dev
      tables:
        - name: all_types
          columns:
            - varchar_c
            - char_c
            - bigint_c
```

Параметры конфигурации

- `enabled` - включение (true)/ выключение (false) расчета статистики, например `STATISTICS_ENABLED:false`;
- `timeout-min` - время обновления статистики (минуты), например `STATISTICS_TIMEOUT_MIN:60`.

2.2.5.2.11 Секция logging

Секция `logging` предназначена для настройки параметров логирования.

Например:

```
logging:
  request-response:
    query-request: ${QUERY_REQUEST_LOG_ENABLED:false}
    query-response: ${QUERY_RESPONSE_LOG_ENABLED:false}
    pf-request: ${PF_REQUEST_LOG_ENABLED:false}
    pf-response: ${PF_RESPONSE_LOG_ENABLED:false}
```

Параметры конфигурации

- `query-request` - журналирование query запросов, например `QUERY_REQUEST_LOG_ENABLED:false`;
- `query-response` - журналирование query ответов, например `QUERY_RESPONSE_LOG_ENABLED:false`;
- `pf-request` - журналирование запросов на сервис формирования документов, например `PF_REQUEST_LOG_ENABLED:false`;
- `pf-response` - журналирование ответов от сервиса формирования документов, например `PF_RESPONSE_LOG_ENABLED:false`.

2.2.5.2.12 Секция metrics

Секция `metrics` предназначена для настройки параметров метрик.

Например:

```
metrics:
  port: ${METRICS_PORT:9837}
```

Параметры конфигурации

- `port` - Порт для метрик, например `METRICS_PORT:9837`.

2.2.6 Настройка СМЭВ4-адаптера – Модуль MPPR

2.2.6.1 Конфигурация СМЭВ4-адаптера - Модуль MPPR (application.yml)

Файл `application.yml` – основной конфигурационный файл модуля, в котором задана его логика и порядок работы модуля: получение входящих запросов, их обработка, а также

настройка подключения к ядру витрины (секция: `prostore`), настройка метрик (секция: `metrics`), а также другие настройки необходимые для корректной работы адаптера.

2.2.6.1.1 Пример файла application.yml

```yaml
http-server:
  port: ${HTTP_PORT:8085}

environment:
  name: ${ENVIRONMENT_NAME:test}

executor:
  reader-pool-size: ${EXECUTOR_READER_POOL_SIZE:20}
  max-execute-time: ${EXECUTOR_MAX_EXECUTE_TIME:600}
  log-pool-size: ${EXECUTOR_LOG_POOL_SIZE:20}

send:
  channel-size: ${SEND_CHANNEL_SIZE:1}
  timeout: ${SEND_TIMEOUT:30}
  delete-topic: ${SEND_DELETE_TOPIC:true}
  compress: ${SEND_COMPRESS:none}

prostore-rest-client:
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9195}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}

prostore:
  kafka:
    message-limit: ${PS_MESSAGE_LIMIT:1000}
    zk-url: ${PS_ZK_KAFKA_URL:localhost:2181}
    statusEventTopic:
      topic: ${PS_STATUS_EVENT_TOPIC:status.event}
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
        group.id: ${kafka.internal.topic.prefix}podd-adapter-mppr-status-event
        auto.offset.reset: earliest
        enable.auto.commit: false

kafka:
  agent.topic.prefix: ${AGENT_TOPIC_PREFIX:}
  max-concurrent-handle: ${KAFKA_MAX_CONCURRENT_HANDLE:10}
  commit-interval: ${KAFKA_COMMIT_INTERVAL:5s}
  external:
    bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
    topic.prefix: ${EXTERNAL_TOPIC_PREFIX:${kafka.agent.topic.prefix}}
  internal:
    bootstrap.servers: ${PS_KAFKA:localhost:9092}
    topic.prefix: ${INTERNAL_TOPIC_PREFIX:${kafka.agent.topic.prefix}}
  consumer:
    query-request:
      topic: ${kafka.internal.topic.prefix}mppr.delegate.rq
      max-concurrent-handle: ${kafka.max-concurrent-handle}
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
        group.id: ${kafka.internal.topic.prefix}mppr.query.consumer
        auto.offset.reset: earliest
        enable.auto.commit: false
        max.poll.records: 1
```

```
      max.poll.interval.ms: 600000
    delta-request:
      topic: ${kafka.internal.topic.prefix}mppr.delta.rq
      max-concurrent-handle: ${kafka.max-concurrent-handle}
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
        group.id: ${kafka.internal.topic.prefix}mppr.delta.consumer
        auto.offset.reset: earliest
        enable.auto.commit: false
        max.poll.records: 1
        max.poll.interval.ms: 600000
    download-data:
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
        group.id: ${kafka.internal.topic.prefix}mppr.x.query.consumer
        auto.offset.reset: earliest
        enable.auto.commit: false
        max.poll.records: 1
  producer:
    query-result: ${kafka.external.topic.prefix}query.rs
    query-error: ${kafka.external.topic.prefix}query.err
    delta-result: ${kafka.external.topic.prefix}delta.rs
    delta-error: ${kafka.external.topic.prefix}delta.err
    property:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
    internal:
      tp-delete-tmp: ${kafka.internal.topic.prefix}tp.delete.tmp
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}

metrics:
  port: ${METRICS_PORT:9843}

logging:
  scl.delta:
    enabled: ${SCL_DELTA_ENABLED:false}
  request-response:
    delta-request: ${DELTA_REQUEST_LOG_ENABLED:false}
    delta-response: ${DELTA_RESPONSE_LOG_ENABLED:false}
    query-request: ${QUERY_REQUEST_LOG_ENABLED:false}
    query-response: ${QUERY_RESPONSE_LOG_ENABLED:false}
```

2.2.6.2 Параметры конфигурации

Настройка конфигурации **СМЭВ4-адаптера - Модуль MPPR** осуществляется путем редактирования параметров настроек в файле `application.yml`, где настраиваются секции:

- `http-server` - указывается порт веб-сервера;
- `environment` - указывается название окружения (`test`, `prod` и т.д.);
- `executor` - предназначена для указания размера пула для запросов;
- `send` - настраиваются ограничения на размер загружаемого файла;
- `prostore-rest-client` - блок параметров конфигурирования взаимодействия с **ProStore**;
- `prostore` - настройка подключения к серверу и базе данных **ProStore**;
- `kafka` - настройки параметров подключения к шине данных Apache Kafka;
- `metrics` - настройка получения метрик;
- `logging` - настройки журналирования запросов и ответов;

2.2.6.2.1 Секция http-server

В секции `http-server` указывается порт веб-сервера.

Например:

```
http-server:
  port: ${HTTP_PORT:8085}
```

Параметры настроек

- `port` - порт веб-сервера, например: `HTTP_PORT:8085`.

2.2.6.2.2 Секция environment

В секции `environment` указывается среда разработки (dev, test, stable, prod)

Например:

```
environment:
  name: ${ENVIRONMENT_NAME:test}
```

Параметры настроек

- `name` - Название окружения, например `ENVIRONMENT_NAME:test`.

2.2.6.2.3 Секция executor

Секция `executor` предназначена для указания размера пула для чтения Kafka и времени выполнения задач.

Например:

```
executor:
  reader-pool-size: ${EXECUTOR_READER_POOL_SIZE:20}
  max-execute-time: ${EXECUTOR_MAX_EXECUTE_TIME:600}
  log-pool-size: ${EXECUTOR_LOG_POOL_SIZE:20}
```

Параметры настроек

- `reader-pool-size` - размер пула для чтения Kafka, например `EXECUTOR_READER_POOL_SIZE:20`;
- `max-execute-time` - максимальное время выполнения задачи (сек), например `EXECUTOR_MAX_EXECUTE_TIME:600`;
- `log-pool-size` - размер пула используемого для журналирования запросов и ответов, например `EXECUTOR_LOG_POOL_SIZE:20`.

2.2.6.2.4 Секция send

В секции `send` настраиваются ограничения на размер загружаемого файла.

Например:

```
send:
  channel-size: ${SEND_CHANNEL_SIZE:1}
  timeout: ${SEND_TIMEOUT:30}
  delete-topic: ${SEND_DELETE_TOPIC:true}
  compress: ${SEND_COMPRESS:none}
```

Параметры настроек

- `channel-size` - размер канала на отправку сообщения, например `SEND_CHANNEL_SIZE:10`;
- `timeout` - таймаут вычитывания данных из топика (сек), например `SEND_TIMEOUT:30`
- `delete-topic` - удаление внешнего топика после выгрузки, например `SEND_DELETE_TOPIC:true`;
- `compress` - сжатие выгружаемых сообщений (none или zstd), например

SEND_COMPRESS:none.

2.2.6.2.5 Секция prostore-rest-client

В секции prostore-rest-client реализован блок параметров конфигурирования взаимодействия с **ProStore**.

Например:

```
prostore-rest-client:
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9195}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}
```

Параметры настроек

- host - адрес **ProStore**, например PS_HOST:localhost;
- port - порт **ProStore**, например PS_PORT:9195;
- max-pool-size - максимальное число подключений к **ProStore**, например PS_MAX_POOL_SIZE:8.

2.2.6.2.6 Секция prostore

В секции prostore осуществляется настройка подключения к серверу и базе данных **ProStore**.

Например:

```
prostore:
  kafka:
    message-limit: ${PS_MESSAGE_LIMIT:1000}
    zk-url: ${PS_ZK_KAFKA_URL:localhost:2181}
    statusEventTopic:
      topic: ${PS_STATUS_EVENT_TOPIC:status.event}
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
        group.id: ${kafka.internal.topic.prefix}podd-adapter-mppr-status-event
        auto.offset.reset: earliest
        enable.auto.commit: false
```

Параметры настроек

- message-limit - лимит сообщений, например PS_MESSAGE_LIMIT:1000;
- zk-url - адрес сервера **Zookeeper** для загрузки данных в **ProStore**, например PS_ZK_KAFKA_URL:localhost:2181.

2.2.6.2.7 Секция kafka

Секция kafka определяет настройки взаимодействия через **СМЭВ4-адаптер** между Поставщиком данных (producer) и Получателем данных (consumer).

В секции kafka собраны настройки параметров подключения к шине данных **Apache Kafka**.

Например:

```
kafka:
  agent.topic.prefix: ${AGENT_TOPIC_PREFIX:}
  max-concurrent-handle: ${KAFKA_MAX_CONCURRENT_HANDLE:10}
  commit-interval: ${KAFKA_COMMIT_INTERVAL:5s}
  external:
    bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
    topic.prefix: ${EXTERNAL_TOPIC_PREFIX:${agent.topic.prefix}}
  internal:
```

```
    bootstrap.servers: ${PS_KAFKA:localhost:9092}
    topic.prefix: ${INTERNAL_TOPIC_PREFIX:${agent.topic.prefix}}
  consumer:
    query-request:
      topic: ${kafka.internal.topic.prefix}mppr.delegate.rq
      max-concurrent-handle: ${kafka.max-concurrent-handle}
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
        group.id: ${kafka.internal.topic.prefix}mppr.query.consumer
        auto.offset.reset: earliest
        enable.auto.commit: false
        max.poll.records: 1
        max.poll.interval.ms: 600000
    delta-request:
      topic: ${kafka.internal.topic.prefix}mppr.delta.rq
      max-concurrent-handle: ${kafka.max-concurrent-handle}
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
        group.id: ${kafka.internal.topic.prefix}mppr.delta.consumer
        auto.offset.reset: earliest
        enable.auto.commit: false
        max.poll.records: 1
        max.poll.interval.ms: 600000
    download-data:
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
        group.id: ${kafka.internal.topic.prefix}mppr.x.query.consumer
        auto.offset.reset: earliest
        enable.auto.commit: false
        max.poll.records: 1
  producer:
    query-result: ${kafka.external.topic.prefix}query.rs
    query-error: ${kafka.external.topic.prefix}query.err
    delta-result: ${kafka.external.topic.prefix}delta.rs
    delta-error: ${kafka.external.topic.prefix}delta.err
    property:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
    internal:
      tp-delete-tmp: ${kafka.internal.topic.prefix}tp.delete.tmp
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
```

Параметры конфигурации

– `topic` - префикс для топиков Агента СМЭВ4, например `AGENT_TOPIC_PREFIX`.

2.2.6.2.8 Секция metrics

Секция `metrics` предназначена для настройки параметров метрик.

Например:

```
metrics:
  port: ${METRICS_PORT:9843}
```

Параметры конфигурации

– `port` - порт для метрик, например `METRICS_PORT:9843`.

2.2.6.2.9 Секция logging

Секция `logging` предназначена для настройки журналирования запросов и ответов.

Например:

```
logging:
  scl.delta:
    enabled: ${SCL_DELTA_ENABLED:false}
  request-response:
    delta-request: ${DELTA_REQUEST_LOG_ENABLED:false}
    delta-response: ${DELTA_RESPONSE_LOG_ENABLED:false}
    query-request: ${QUERY_REQUEST_LOG_ENABLED:false}
    query-response: ${QUERY_RESPONSE_LOG_ENABLED:false}
```

LOG_FORMAT - Логирование в формате (JSON/TEXT) - указывается в `logback.xml`.

2.2.7 Настройка СМЭВ4-адаптера - Модуль MPPW

2.2.7.1 Конфигурация модуля СМЭВ4-адаптер - Модуль MPPW (application.yml)

Файл `application.yml` – основной конфигурационный файл модуля, в котором задана его логика и порядок работы модуля: подключение к Агенту СМЭВ4, Брокеру сообщений **Kafka**, **Zookeeper**, а также настройка подключения к **Prostore** (секция: `prostore`), настройка метрик (секция: `metrics`) и другие настройки необходимые для корректной работы адаптера.

2.2.7.1.1 Пример файла application.yml

В конфигурационном файле следует задавать только те настройки, которые необходимы для решения текущих бизнес-задач.

```
tp:
  data-topic-prefix: ${DATA_TOPIC_PREFIX:tp.data}
  upload-topic-prefix: ${UPLOAD_TOPIC_PREFIX:tmp.w}

upload:
  data-topic-prefix: ${AGENT_TOPIC_PREFIX:}
  retry:
    attempts: 5
    delay: 1m


http-server:
  port: ${HTTP_PORT:8090}

environment:
  name: ${ENVIRONMENT_NAME:test}

executor:
  reader-pool-size: ${EXECUTOR_READER_POOL_SIZE:20}
  max-execute-time: ${EXECUTOR_MAX_EXECUTE_TIME:600}

scheduler:
  delta-apply-request-timeout: ${DELTA_APPLY_REQUEST_TIMEOUT:60}

spring:
  liquibase:
    enabled: ${LIQUIBASE_ENABLED:true} # Для replication.storage.type=kafka необходимо выключить
    change-log: classpath:/liquibase-changes/changelog.xml
    driver-class-name: org.postgresql.Driver
    url:
jdbc:postgresql://${replication.storage.postgres.connection.host}:${replication.storage.postgres.connection.port}/${replication.storage.postgres.connection.database}?currentSchema=${replication.storage.postgres.connection.schema}
    user: ${replication.storage.postgres.connection.user}
```

```
    password: ${replication.storage.postgres.connection.password}

replication:
  should-check-source-topic: false
  should-delete-prostore-topic: false
  prostore-topic-template: "mppw.{datamart}.{table}"
  storage: # блок настроек хранения чанков данных репликации. При изменении
параметров необходимо синхронизировать аналогичные параметры в сервисе mppw
    type: ${STORAGE_TYPE:postgres} # тип, postgres|kafka
    postgres: # параметры подключения к базе. Используется только при type=postgres
      connection:
        database: ${STORAGE_DATABASE:replication}
        schema: ${STORAGE_SCHEMA:public}
        host: ${STORAGE_HOST:localhost}
        port: ${STORAGE_PORT:5432}
        user: ${STORAGE_USER:postgres}
        password: ${STORAGE_PASSWORD:postgres}
      pool:
        max-size: ${STORAGE_POOL_SIZE:30}

delta:
  # параметр ожидания перед повторной попыткой открытия дельты в случае ошибки
  open-delay: ${DELTA_OPEN_DELAY:${scheduler.delta-apply-request-timeout}}s
  # количество попыток открытия дельты в случае ошибки
  open-attempts: ${DELTA_OPEN_ATTEMPTS:5}
  # период проверки открытых дельт
  open-check: ${DELTA_OPEN_CHECK:60}s
  # количество попыток фиксации дельты в случае ошибки
  commit-attempts: ${DELTA_COMMIT_ATTEMPTS:5}
  # период ожидания перед повторной попыткой фиксации дельты
  commit-error-delay: ${DELTA_COMMIT_DELAY:1}s
  # количество попыток отката дельты в случае ошибки
  rollback-attempts: ${DELTA_COMMIT_ATTEMPTS:3}
  # период ожидания перед повторной попыткой отката дельты
  rollback-error-delay: ${DELTA_COMMIT_DELAY:5}s
  #enable/disable/period
  creating-delta-on-upload-request: ${DELTA_CREATING_MODE:enable}
  period: ${CREATING_DELTA_ON_UPLOAD_REQUEST_PERIOD:300}s



send:
  channel-size: ${SEND_CHANNEL_SIZE:10}
  timeout: ${SEND_TIMEOUT:60}

zookeeper:
  connection-string: ${ZOOKEEPER_DS_ADDRESS:localhost}
  connection-timeout-ms: ${ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000}
  session-timeout-ms: ${ZOOKEEPER_DS_SESSION_TIMEOUT_MS:86400000}
  chroot: ${ZOOKEEPER_DS_CHROOT:/adapter}

prostore-rest-client:
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9195}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}

prostore:
  key:
    primary: ${PRIMARY_KEY_NAME:tmp_id}
    type: ${PRIMARY_KEY_TYPE:BIGINT}
  kafka:
```

```
    message-limit: ${PS_MESSAGE_LIMIT:1000}
    zk-url: ${PS_ZK_KAFKA_URL:localhost:2181}
    properties:
      bootstrap.servers: ${PS_KAFKA:localhost:9092}

kafka:
  agent.topic.prefix: ${AGENT_TOPIC_PREFIX:}
  max-concurrent-handle: ${KAFKA_MAX_CONCURRENT_HANDLE:10}
  commit-interval: ${KAFKA_COMMIT_INTERVAL:5s}
  internal:
    bootstrap.servers: ${PS_KAFKA:localhost:9092}
    topic.prefix: ${INTERNAL_TOPIC_PREFIX:${kafka.agent.topic.prefix}}
  consumer:
    tp-request:
      topic: ${kafka.internal.topic.prefix}mppw.tp
      max-concurrent-handle: ${kafka.max-concurrent-handle}
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
        group.id: ${kafka.internal.topic.prefix}podd-adapter-mppw-tp
        auto.offset.reset: earliest
        enable.auto.commit: false
    upload-request:
      topic: ${kafka.internal.topic.prefix}mppw.upload.rq
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
        group.id: ${kafka.internal.topic.prefix}podd-adapter-mppw-upload
        auto.offset.reset: earliest
        enable.auto.commit: false
    delta-apply-request:
      topic: ${kafka.internal.topic.prefix}mppw.delta.in.rq
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
        group.id: ${kafka.internal.topic.prefix}podd-adapter-mppw-delta
        auto.offset.reset: earliest
        enable.auto.commit: false
    upload-data:
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
        group.id: ${kafka.internal.topic.prefix}podd-adapter-mppw-data
        auto.offset.reset: earliest
        enable.auto.commit: false
  producer:
    tp-result: ${kafka.internal.topic.prefix}mppw.rs
    upload-result: ${kafka.internal.topic.prefix}mppw.upload.rs
    delta-apply-result: ${kafka.internal.topic.prefix}mppw.delta.in.rs
    property:
      bootstrap.servers: ${kafka.internal.bootstrap.servers}

metrics:
  port: ${METRICS_PORT:9843}

jet-connector:
  use: ${JET_CONNECTOR_USE:true}
```

2.2.7.2 Параметры конфигурации

Настройка конфигурации **СМЭВ4-адаптера - Модуль MPPW** осуществляется путем редактирования параметров настроек в файле `application.yml`, где настраиваются секции:

- `tp` - указываются настройки топиков для табличных параметров;
- `upload` - настройки загрузки через DATA-Uploader;
- `http-server` - указывается порт для подключения;
- `environment` - указывается название окружения (`test`, `prod` и т.д.);
- `executor` - настраивается размер пула для запросов;
- `scheduler` - настройки планировщика отложенных заданий;
- `replication` - блок настроек хранения чанков данных репликации;
- `send` - настраиваются ограничения на размер загружаемого файла;
- `zookeeper` - указываются настройки подключения к Zookeeper;
- `prostore-rest-client` - блок параметров конфигурирования взаимодействия с **ProStore**;
- `prostore` - указываются настройки генерации на создание и удаление **writable table**;
- `kafka` - настройки параметров подключения к шине данных Apache Kafka;
- `metrics` - настройка получения метрик;
- `jet-connector` - подготовлен для оптимизации задержек записи.

2.2.7.2.1 Секция tp

Секция `tp` предназначена для настройки префиксов топиков, откуда будут вычитываться и куда будут загружаться данные.

Например:

```
tp:
  data-topic-prefix: ${DATA_TOPIC_PREFIX:tp.data}
  upload-topic-prefix: ${UPLOAD_TOPIC_PREFIX:tmp.w}
```

Параметры конфигурации

- `data-topic-prefix` - префикс топика, откуда будут вычитываться данные, например `DATA_TOPIC_PREFIX:tp.data`;
- `upload-topic-prefix` - префикс топика, куда будут загружаться данные для их последующей загрузки в витрину, например `UPLOAD_TOPIC_PREFIX:tmp.w`.

2.2.7.2.2 Секция upload

В секции `upload` - указываются настройки загрузки через **DATA-Uploader**.

Например:

```
upload:
  data-topic-prefix: ${AGENT_TOPIC_PREFIX:}
  retry:
    attempts: 5
    delay: 1m
```

Параметры конфигурации

- `AGENT_TOPIC_PREFIX:` - значение префикса для топиков. Топики взаимодействия с **СМЭВ4-адаптером - Модуль исполнения запросов** (см. Спецификация Модуля исполнения запросов);
- `retry` - повтор попытки загрузки;
- `attempts` - количество попыток;
- `delay` - период задержки (в минутах).

2.2.7.2.3 Секция http-server

В секции `http-server` указывается порт веб-сервера.

Например:

```
http-server:
  port: ${HTTP_PORT:8090}
```

Параметры настроек

– `port` - указывается порт веб-сервера, например: `HTTP_PORT:8090`.

2.2.7.2.4 Секция environment

В секции `environment` указывается среда разработки (dev, test, stable, prod).

Например:

```
environment:
  name: ${ENVIRONMENT_NAME:test}
```

Параметры настроек

– `name` - среда разработки, например `ENVIRONMENT_NAME:test`.

2.2.7.2.5 Секция executor

Секция `executor` предназначена для указания размера пула для чтения **Kafka** и времени выполнения задач.

Например:

```
executor:
  reader-pool-size: ${EXECUTOR_READER_POOL_SIZE:20}
  max-execute-time: ${EXECUTOR_MAX_EXECUTE_TIME:600}
```

Параметры настроек

– `reader-pool-size` - размер пула для чтения **Kafka**, например `EXECUTOR_READER_POOL_SIZE:20`;

– `max-execute-time` - максимальное время выполнения задачи (в секундах), например `EXECUTOR_MAX_EXECUTE_TIME:600`.

2.2.7.2.6 Секция scheduler

Секция `scheduler` предназначена для настроек планировщика отложенных заданий.

Например:

```
scheduler:
  delta-apply-request-timeout: ${DELTA_APPLY_REQUEST_TIMEOUT:60}
```

Параметры настроек

– `delta-apply-request-timeout` - таймаут применения отложенной дельты, например `DELTA_APPLY_REQUEST_TIMEOUT:60`.

2.2.7.2.7 Секция replication

секции `replication` указываются настройки хранения чанков данных репликации.

Например:

```
replication:
  should-check-source-topic: false
  should-delete-prostore-topic: false
  prostore-topic-template: "mppw.{datamart}.{table}"
  storage: # блок настроек хранения чанков данных репликации. При изменении
параметров необходимо синхронизировать аналогичные параметры в сервисе mppw
    type: ${STORAGE_TYPE:postgres} # тип, postgres|kafka
    postgres: # параметры подключения к базе. Используется только при type=postgres
      connection:
        database: ${STORAGE_DATABASE:replication}
        schema: ${STORAGE_SCHEMA:public}
        host: ${STORAGE_HOST:localhost}
        port: ${STORAGE_PORT:5432}
        user: ${STORAGE_USER:postgres}
        password: ${STORAGE_PASSWORD:postgres}
      pool:
        max-size: ${STORAGE_POOL_SIZE:30}
```

Параметры настроек

- `type` - тип, postgres|kafka, например: `STORAGE_TYPE:postgres`;
- `connection` - параметры подключения к базе.

2.2.7.2.8 Секция delta

В секции `delta` настраивается параметр ожидания перед повторной попыткой открытия дельты при ошибке.

Например:

```
delta:
  # параметр ожидания перед повторной попыткой открытия дельты в случае ошибки
  open-delay: ${DELTA_OPEN_DELAY:${scheduler.delta-apply-request-timeout}}s
  # количество попыток открытия дельты в случае ошибки
  open-attempts: ${DELTA_OPEN_ATTEMPTS:5}
  # период проверки открытых дельт
  open-check: ${DELTA_OPEN_CHECK:60}s
  # количество попыток фиксации дельты в случае ошибки
  commit-attempts: ${DELTA_COMMIT_ATTEMPTS:5}
  # период ожидания перед повторной попыткой фиксации дельты
  commit-error-delay: ${DELTA_COMMIT_DELAY:1}s
  # количество попыток отката дельты в случае ошибки
  rollback-attempts: ${DELTA_COMMIT_ATTEMPTS:3}
  # период ожидания перед повторной попыткой отката дельты
  rollback-error-delay: ${DELTA_COMMIT_DELAY:5}s
  #enable/disable/period
  creating-delta-on-upload-request: ${DELTA_CREATING_MODE:enable}
  period: ${CREATING_DELTA_ON_UPLOAD_REQUEST_PERIOD:300}s
```

Параметры настроек

- `open-delay` - параметр ожидания (в секундах) перед повторной попыткой открытия дельты в случае ошибки, например `DELTA_OPEN_DELAY:${scheduler.delta-apply-request-timeout}`;
- `open-attempts` - количество попыток открытия дельты в случае ошибки, например `DELTA_OPEN_ATTEMPTS:5`;
- `open-check` - период проверки открытых дельт (в секундах), например `DELTA_OPEN_CHECK:60`;
- `commit-attempts` - количество попыток фиксации дельты в случае ошибки, например `DELTA_COMMIT_ATTEMPTS:5`;
- `commit-error-delay` - период ожидания перед повторной попыткой фиксации

дельты (в секундах), например DELTA_COMMIT_DELAY:1;

- rollback-attempts - количество попыток отката дельты в случае ошибки, например DELTA_COMMIT_ATTEMPTS:3;
- rollback-error-delay - период ожидания перед повторной попыткой отката дельты (в секундах), например DELTA_COMMIT_DELAY:5;
- creating-delta-on-upload-request - создание дельты при запросе загрузки, например DELTA_CREATING_MODE:enable;
- period - период создания дельты при запросе загрузки (в секундах), например CREATING_DELTA_ON_UPLOAD_REQUEST_PERIOD:300, используется только при mode=llw и creating-delta-on-upload-request=period;

Для параметра creating-delta-on-upload-request доступны значения:

- enable - формируются дельты для загружаемых порций данных, данное значение используется для работы подписок;
- disable - формирование дельт отключено;
- period - исключается формирование дельт в процессе исполнения команд загрузки данных, вместо этого дельты периодически открываются и закрываются с периодом, указанным в данном параметре. При этом приложение запоминает датамарты по которым завершены инсерты за истекший период. Дельты открываются и закрываются только для датамартов, в которые происходила вставка данных.

Внимание:

При потере связи коннектора с Prostore отмены операции не происходит автоматически и модуль MPPW получает ошибку. После восстановления связи коннектора с Prostore и попытке загрузки данных в эту же таблицу возвращается ошибка о блокировке таблицы предыдущей операции вставки. Для исправления ошибки Администратор должен вручную выполнить команду resume или erase в Prostore.

2.2.7.2.9 Секция send

В секции send настраиваются ограничения на размер загружаемого файла.

Например:

```
send:
  channel-size: ${SEND_CHANNEL_SIZE:10}
  timeout: ${SEND_TIMEOUT:60}
```

Параметры настроек

- channel-size - размер канала на отправку сообщения, например SEND_CHANNEL_SIZE:10;
- timeout - таймаут вычитывания данных из топика (сек), например SEND_TIMEOUT:60.

2.2.7.2.10 Секция zookeeper

Секция zookeeper предназначена для настройки параметров подключения к серверу **Zookeeper**.

Например:

```
zookeeper:
  connection-string: ${ZOOKEEPER_DS_ADDRESS:t5-adsp-01.ru-central1.internal}
  connection-timeout-ms: ${ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000}
  session-timeout-ms: ${ZOOKEEPER_DS_SESSION_TIMEOUT_MS:86400000}
  chroot: ${ZOOKEEPER_DS_CHROOT:/adapter}
```

Параметры конфигурации

- `connection-string` - подключение к Zookeeper DS, например `ZOOKEEPER_DS_ADDRESS:t5-adsp-01.ru-central1.internal`;
- `connection-timeout-ms` - Zookeeper DS таймаут подключения, например `ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000`;
- `session-timeout-ms` - Zookeeper DS таймаут сессии, например `ZOOKEEPER_DS_SESSION_TIMEOUT_MS:86400000`;
- `chroot` - Zookeeper DS chroot path, например `ZOOKEEPER_DS_CHROOT:/adapter`.

2.2.7.2.11 Секция prostore-rest-client

В секции `prostore-rest-client` реализован блок параметров конфигурирования взаимодействия с **Prostore**.

Например:

```
prostore-rest-client:
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9195}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}
```

Параметры настроек

- `host` - адрес **Prostore**, например `PS_HOST:localhost`;
- `port` - порт **Prostore**, например `PS_PORT:9195`;
- `max-pool-size` - максимальное число подключений к **Prostore**, например `PS_MAX_POOL_SIZE:8`.

2.2.7.2.12 Секция prostore

Секция `prostore`

Например:

```
prostore:
  key:
    primary: ${PRIMARY_KEY_NAME:tmp_id}
    type: ${PRIMARY_KEY_TYPE:BIGINT}
  kafka:
    message-limit: ${PS_MESSAGE_LIMIT:1000}
    zk-url: ${PS_ZK_KAFKA_URL:localhost:2181}
    properties:
      bootstrap.servers: ${PS_KAFKA:localhost:9092}
```

Параметры настроек

- `primary` - название первичного ключа, например `tmp_id`;
- `type` - тип первичного ключа, например `BIGINT`;
- `message-limit` - лимит сообщений, например `PS_MESSAGE_LIMIT:1000`;
- `zk-url` - адрес сервера **Zookeeper** для загрузки данных в **Prostore**, например `PS_ZK_KAFKA_URL:localhost:2181`.

2.2.7.2.13 Секция kafka

В секции `kafka` собраны настройки параметров подключения к шине данных **Apache Kafka**.

Например:

```
kafka:
  agent.topic.prefix: ${AGENT_TOPIC_PREFIX:}
  max-concurrent-handle: ${KAFKA_MAX_CONCURRENT_HANDLE:10}
```

```
  commit-interval: ${KAFKA_COMMIT_INTERVAL:5s}
  internal:
    bootstrap.servers: ${PS_KAFKA:localhost:9092}
    topic.prefix: ${INTERNAL_TOPIC_PREFIX:${agent.topic.prefix}}
  consumer:
    tp-request:
      topic: ${kafka.internal.topic.prefix}mppw.tp
      max-concurrent-handle: ${kafka.max-concurrent-handle}
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
        group.id: ${kafka.internal.topic.prefix}podd-adapter-mppw-tp
        auto.offset.reset: earliest
        enable.auto.commit: false
    upload-request:
      topic: ${kafka.internal.topic.prefix}mppw.upload.rq
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
        group.id: ${kafka.internal.topic.prefix}podd-adapter-mppw-upload
        auto.offset.reset: earliest
        enable.auto.commit: false
    delta-apply-request:
      topic: ${kafka.internal.topic.prefix}mppw.delta.in.rq
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
        group.id: ${kafka.internal.topic.prefix}podd-adapter-mppw-delta
        auto.offset.reset: earliest
        enable.auto.commit: false
    upload-data:
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
        group.id: ${kafka.internal.topic.prefix}podd-adapter-mppw-data
        auto.offset.reset: earliest
        enable.auto.commit: false
  producer:
    tp-result: ${kafka.internal.topic.prefix}mppw.rs
    upload-result: ${kafka.internal.topic.prefix}mppw.upload.rs
    delta-apply-result: ${kafka.internal.topic.prefix}mppw.delta.in.rs
    property:
      bootstrap.servers: ${kafka.internal.bootstrap.servers}
```

Параметры конфигурации

– `AGENT_TOPIC_PREFIX` - префикс для топиков Агента СМЭВ4, например.

2.2.7.2.14 Секция metrics

Секция `metrics` предназначена для настройки параметров метрик.

Например:

```
metrics:
  port: ${METRICS_PORT:9843}
```

Параметры конфигурации

– `port` - порт для метрик, например `METRICS_PORT:9843`.

2.2.7.2.15 Секция jet-connector

Секция `jet-connector` предназначена для оптимизации задержек записи данных.

Например:

```
jet-connector:
use: ${JET_CONNECTOR_USE:false}
```

Параметры настроек

– `use` - флаг активации **jet-connector**, например `{JET_CONNECTOR_USE:false}`;

Таблица 2.14 Область применения

Режим/СУБД	ADB	ADP	ADQM	ADG
Чтение	Нет	Нет	Нет	Нет
Запись	В перспективе	Да	Нет	Нет

Чтобы воспользоваться **jet-connector** требуется вместо `upload external table` создавать `readable external table`, указывающую на топик, как отражено в документации Prostore

В модуль MPPW не передается информация о том, в какую БД физически будут загружаться данные, синтаксис **Prostore** един для всех поддерживаемых СУБД. Конкретная СУБД указывается в настройках **Prostore**.

Даже если загрузка данных выполняется в более чем одну базу, на работе адаптеров это не сказывается.

При формировании запросов в табличными параметрами, в том числе при регистрации РЗ, необходимо явно перечислять поля таблиц, по которым будет выполняться фильтрация записей или объединение таблиц.

Переключение с **jet-connector** на **kafka postgres writer** и наоборот допускается при завершенных операциях загрузки данных.

Предупреждение:

jet-connector в настоящее время применим для ADP, что делает его применимым, только для исталляций одной единственной СУБД ADP. Это ограничение остается на уровне документации при использовании jet-connector с другими базами должна появиться ошибка.

2.2.8 Настройка СМЭВ4-адаптера – Модуль импорта данных табличных параметров

2.2.8.1 Конфигурация модуля СМЭВ4-адаптер - Модуль импорта данных табличных параметров (application.yml)

Файл `application.yml` – основной конфигурационный файл модуля, в котором задана его логика и порядок работы модуля.

2.2.8.1.1 Пример файла application.yml

В конфигурационном файле следует задавать только те настройки, которые необходимы для решения текущих бизнес-задач.

```
http-server:
  port: ${HTTP_PORT:8091}

environment:
  name: ${ENVIRONMENT_NAME:test}

zookeeper:
  # Подключение к зукипер
  connection-string: ${ZOOKEEPER_DS_ADDRESS:localhost:2181}
  connection-timeout-ms: ${ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000}
  session-timeout-ms: ${ZOOKEEPER_DS_SESSION_TIMEOUT_MS:86400000}
  chroot: ${ZOOKEEPER_DS_CHROOT:/adapter}
```

```yaml
prostore-rest-client:
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9195}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}

prostore:
  key:
    primary: tmp_id
    type: BIGINT
  standalone:
    source: ${PS_STANDALONE_SOURCE:ADP}


# общие настройки подключения к Kafka
kafka:
  agent.topic.prefix: ${AGENT_TOPIC_PREFIX:}
  # максимальное количество обработчиков входящих запросов
  max-concurrent-handle: ${KAFKA_MAX_CONCURRENT_HANDLE:1000}
  # периодичность фиксации оффсета обработанных сообщений
  commit-interval: ${KAFKA_COMMIT_INTERVAL:5s}
  external:
    bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
    topic.prefix: ${EXTERNAL_TOPIC_PREFIX:${kafka.agent.topic.prefix}}
  internal:
    bootstrap.servers: ${PS_KAFKA:localhost:9092}
    topic.prefix: ${INTERNAL_TOPIC_PREFIX:${kafka.agent.topic.prefix}}
  # параметры консьюмера сообщений
  consumer:
    bootstrap.servers: ${kafka.internal.bootstrap.servers}
    # Смещение при первом запуске приложения
    auto.offset.reset: earliest
  # параметры продюсера сообщений
  producer:
    bootstrap.servers: ${kafka.internal.bootstrap.servers}
  # параметры клиента администратора для удаления временных топиков
  admin:
    bootstrap.servers: ${kafka.internal.bootstrap.servers}

# блок настроек импорта данных табличных параметров
query:
  data-topic-prefix: ${DATA_TOPIC_PREFIX:tp.data}
  # топик запросов
  request-topic: ${kafka.internal.topic.prefix}tp.upload.query
  # топик запросов на исполнение после загрузки данных
  response-topic: ${kafka.external.topic.prefix}query.rq
  # топик с ошибками
  error-topic: ${kafka.external.topic.prefix}query.err
  # блок настроек отмены запросов
  cancel:
    # топик запросов на отмену исполнения
    request-topic: ${kafka.external.topic.prefix}cancel.rq
    # настройки кеша отмененных запросов
    cache:
      # начальный размер кеша
      initial-capacity: 10000
      # время вытеснения из кеша
      expire-after-access: 60m
  # максимальное количество обработчиков входящих запросов
  max-concurrent-handle: ${kafka.max-concurrent-handle}
  # периодичность фиксации оффсета обработанных сообщений
```

```
  commit-interval: ${kafka.commit-interval}
  # дополнительные параметры консьюмера запросов
  consumer-properties:
    group.id: ${kafka.internal.topic.prefix}import.tp.query.consumer
  # дополнительные параметры продюсера ответов
  producer-properties:
    bootstrap.servers: ${kafka.external.bootstrap.servers}
  # дополнительные параметры администрирования кафки
  admin-properties: [ ]

# блок настроек сервиса загрузки данных mppw
mppw:
  # топик запросов к сервису mppw
  request-topic: ${kafka.internal.topic.prefix}mppw.tp
  # топик ответов от сервиса mppw
  response-topic: ${kafka.internal.topic.prefix}mppw.rs
  # максимальное количество обработчиков входящих запросов
  max-concurrent-handle: ${kafka.max-concurrent-handle}
  # периодичность фиксации оффсета обработанных сообщений
  commit-interval: ${kafka.commit-interval}
  # дополнительные параметры консьюмера запросов
  consumer-properties:
    group.id: ${kafka.internal.topic.prefix}import.tp.response.consumer
    # Смещение при первом запуске приложения
    auto.offset.reset: earliest
  # дополнительные параметры продюсера ответов
  producer-properties: [ ]

# блок настроек механизма очистки
delete:
  # входящий топик запросов удаления ресурсов
  request-topic: ${kafka.internal.topic.prefix}tp.delete.tmp
  # максимальное количество обработчиков входящих запросов
  max-concurrent-handle: ${kafka.max-concurrent-handle}
  # периодичность фиксации оффсета обработанных сообщений
  commit-interval: ${kafka.commit-interval}
  # дополнительные параметры консьюмера запросов
  consumer-properties:
    group.id: ${kafka.internal.topic.prefix}import.tp.delete.consumer

# блок настроек дельт
delta:
  # префикс топиков с данными
  data-topic-prefix: ${DATA_TOPIC_PREFIX:tp.data}
  # признак необходимости удаления топиков
  delete-topics: ${IMPORT_DELTA_DELETE_TOPICS:true}
  # дополнительные параметры администрирования кафки
  admin-properties: [ ]
  # блок параметров поставщика данных
  provider:
    # топик запросов
    request-topic: ${kafka.internal.topic.prefix}tp.upload.delta
    # топик ответов
    response-topic: ${kafka.internal.topic.prefix}delta.rq
    # топик ошибок
    error-topic: ${kafka.internal.topic.prefix}tp.upload.err
    # максимальное количество обработчиков входящих запросов
    max-concurrent-handle: ${kafka.max-concurrent-handle}
    # периодичность фиксации оффсета обработанных сообщений
    commit-interval: ${kafka.commit-interval}
    # дополнительные параметры консьюмера запросов
    consumer-properties:
```

```
      group.id: ${kafka.internal.topic.prefix}import.tp.delta.consumer
    # дополнительные параметры продюсера ответов
    producer-properties: [ ]
  # блок параметров получателя данных
  recipient:
    # топик запросов
    request-topic: ${kafka.internal.topic.prefix}tp.upload.delta.in
    # топик ответов
    response-topic: ${kafka.internal.topic.prefix}delta.in
    # топик ошибок
    error-topic: ${kafka.internal.topic.prefix}tp.upload.in.err
    # максимальное количество обработчиков входящих запросов
    max-concurrent-handle: ${kafka.max-concurrent-handle}
    # периодичность фиксации оффсета обработанных сообщений
    commit-interval: ${kafka.commit-interval}
    # дополнительные параметры консьюмера запросов
    consumer-properties:
      group.id: ${kafka.internal.topic.prefix}import.tp.delta.in.consumer
    # дополнительные параметры продъюсера ответов
    producer-properties: [ ]

metrics:
  # Порт сервера для получения метрик
  port: ${METRICS_PORT:19843}
  kafkaAdminClientName: kafkaAdminClient
```

2.2.8.2 Параметры конфигурации

Настройка конфигурации **СМЭВ4-адаптера - Модуль импорта данных табличных параметров** осуществляется путем редактирования параметров настроек в файле `application.yml`, где настраиваются секции:

- `http-server` - указывается порт веб-сервера;
- `environment` - название окружения (`test`, `prod` и т.д.);
- `zookeeper` – строка подключения к сервисной БД Zookeeper;
- `prostore-api-client` - блок параметров конфигурирования взаимодействия с **Prostore**;
- `prostore` - указываются настройки генерации на создание и удаление **writable table**;
- `kafka` - общие настройки подключения к **Kafka**;
- `query` - блок настроек импорта данных табличных параметров;
- `mppw` - блок настроек сервиса загрузки данных mppw;
- `delete` - блок настроек механизма очистки;
- `delta` - настройка работы с импортом дельт;
- `metrics` - настройка получения метрик.

2.2.8.2.1 Секция http-server

В секции `http-server` указывается порт веб-сервера.

Например:

```
http:
  port: ${HTTP_PORT:8091}
```

Параметры настроек

- `port` - указывается порт веб-сервера, например: `HTTP_PORT:8091`.

2.2.8.2.2 Секция environment

В секции `environment` указывается среда разработки (dev, test, stable, prod).

Например:

```
environment:
  name: ${ENVIRONMENT_NAME:test}
```

Параметры настроек

- `name` - среда разработки, например `ENVIRONMENT_NAME:test`.

2.2.8.2.3 Секция zookeeper

Секция `zookeeper` предназначена для настройки параметров подключения к серверу **Zookeeper**.

Например:

```
zookeeper:
  # Подключение к зукипер
  connection-string: ${ZOOKEEPER_DS_ADDRESS:localhost:2181}
  connection-timeout-ms: ${ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000}
  session-timeout-ms: ${ZOOKEEPER_DS_SESSION_TIMEOUT_MS:86400000}
  chroot: ${ZOOKEEPER_DS_CHROOT:/adapter}
```

Параметры настроек

- `connection-string` - подключение к Zookeeper DS, например `ZOOKEEPER:t5-adsp-01.ru-central1.internal:2181`;
- `connection-timeout-ms` - Zookeeper DS таймаут подключения, например `ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000`;
- `session-timeout-ms` - Zookeeper DS таймаут сессии, например `ZOOKEEPER_DS_SESSION_TIMEOUT_MS:86400000`;
- `chroot` - Zookeeper DS chroot path, например `ZOOKEEPER_DS_CHROOT:/adapter`.

2.2.8.2.4 Секция prostore-rest-client

В секции `prostore-rest-client` реализован блок параметров конфигурирования взаимодействия с **Prostore**.

Например:

```
prostore-rest-client:
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9195}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}
```

Параметры настроек

- `host` - адрес **Prostore**, например `PS_HOST:localhost`;
- `port` - порт **Prostore**, например `PS_PORT:9195`;
- `max-pool-size` - максимальное число подключений к **Prostore**, например `PS_MAX_POOL_SIZE:8`.

2.2.8.2.5 Секция prostore

В секции `prostore` указываются настройки генерации на создание и удаление **writable table**.

Например:

```
prostore:
  key:
    primary: tmp_id
    type: BIGINT
  standalone:
    source: ${PS_STANDALONE_SOURCE:ADP}
```

Параметры настроек

- `primary` - название первичного ключа, например `tmp_id`;
- `type` - тип первичного ключа, например `BIGINT`;
- `source` - источник для standalone таблицы (ADB/ADP), например `PS_STANDALONE_SOURCE:ADP`.

2.2.8.2.6 Секция kafka

Секция `kafka` хранит общие настройки подключения к **Kafka**.

Например:

```
kafka:
  agent.topic.prefix: ${AGENT_TOPIC_PREFIX:}
  # максимальное количество обработчиков входящих запросов
  max-concurrent-handle: ${KAFKA_MAX_CONCURRENT_HANDLE:1000}
  # периодичность фиксации оффсета обработанных сообщений
  commit-interval: ${KAFKA_COMMIT_INTERVAL:5s}
  external:
    bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
    topic.prefix: ${EXTERNAL_TOPIC_PREFIX:${agent.topic.prefix}}
  internal:
    bootstrap.servers: ${PS_KAFKA:localhost:9092}
    topic.prefix: ${INTERNAL_TOPIC_PREFIX:${agent.topic.prefix}}
  # параметры консьюмера сообщений
  consumer:
    bootstrap.servers: ${kafka.internal.bootstrap.servers}
    # Смещение при первом запуске приложения
    auto.offset.reset: earliest
  # параметры продюсера сообщений
  producer:
    bootstrap.servers: ${kafka.internal.bootstrap.servers}
  # параметры клиента администратора для удаления временных топиков
  admin:
    bootstrap.servers: ${kafka.internal.bootstrap.servers}
```

Параметры настроек

- `max-concurrent-handle` - максимальное количество обработчиков входящих запросов;
- `commit-interval` - периодичность фиксации оффсета обработанных сообщений;
- `group.id` - GroupId для подписчика;
- `auto.offset.reset` - смещение при первом запуске приложения;
- `enable.auto.commit`- Вкл/выкл автоматический коммит;
- `consumer` - параметры консьюмера сообщений;
- `producer` – параметры продюсера сообщений.

2.2.8.2.7 Секция query

Секция `query` хранит блок настроек импорта данных табличных параметров.

Например:

```
query:
  data-topic-prefix: ${DATA_TOPIC_PREFIX:tp.data}
```

```yaml
# топик запросов
request-topic: ${kafka.internal.topic.prefix}tp.upload.query
# топик запросов на исполнение после загрузки данных
response-topic: ${kafka.external.topic.prefix}query.rq
# топик с ошибками
error-topic: ${kafka.external.topic.prefix}query.err
# блок настроек отмены запросов
cancel:
  # топик запросов на отмену исполнения
  request-topic: ${kafka.external.topic.prefix}cancel.rq
  # настройки кеша отмененных запросов
  cache:
    # начальный размер кеша
    initial-capacity: 10000
    # время вытеснения из кеша
    expire-after-access: 60m
# максимальное количество обработчиков входящих запросов
max-concurrent-handle: ${kafka.max-concurrent-handle}
# периодичность фиксации оффсета обработанных сообщений
commit-interval: ${kafka.commit-interval}
# дополнительные параметры консьюмера запросов
consumer-properties:
  group.id: ${kafka.internal.topic.prefix}import.tp.query.consumer
# дополнительные параметры продюсера ответов
producer-properties:
  bootstrap.servers: ${kafka.external.bootstrap.servers}
# дополнительные параметры администрирования кафки
admin-properties: [ ]
```

Параметры настроек

- `request-topic` - топик запросов;
- `response-topic` - топик запросов на исполнение после загрузки данных;
- `error-topic` - топик с ошибками;
- `cancel`- блок настроек отмены запросов;
- `initial-capacity` - начальный размер кеша;
- `expire-after-access` - время вытеснения из кеша;
- `max-concurrent-handle` - максимальное количество обработчиков входящих запросов;
- `commit-interval` - периодичность фиксации оффсета обработанных сообщений;
- `consumer-properties` - дополнительные параметры консьюмера запросов;
- `producer-properties` - дополнительные параметры продюсера ответов;
- `admin-properties` - дополнительные параметры администрирования **Kafka**.

2.2.8.2.8 Секция mppw

Секция `mppw` хранит блок настроек сервиса загрузки данных **MPPW**.

```yaml
mppw:
  # топик запросов к сервису mppw
  request-topic: ${kafka.internal.topic.prefix}mppw.tp
  # топик ответов от сервиса mppw
  response-topic: ${kafka.internal.topic.prefix}mppw.rs
  # максимальное количество обработчиков входящих запросов
  max-concurrent-handle: ${kafka.max-concurrent-handle}
  # периодичность фиксации оффсета обработанных сообщений
  commit-interval: ${kafka.commit-interval}
  # дополнительные параметры консьюмера запросов
  consumer-properties:
    group.id: ${kafka.internal.topic.prefix}import.tp.response.consumer
    # Смещение при первом запуске приложения
    auto.offset.reset: earliest
  # дополнительные параметры продюсера ответов
  producer-properties: [ ]
```

Параметры настроек

- `request-topic` - топик запросов к сервису **MPPW**;
- `response-topic` - топик ответов от сервиса **MPPW**;
- `max-concurrent-handle` - максимальное количество обработчиков входящих запросов;
- `commit-interval`- периодичность фиксации оффсета обработанных сообщений;
- `consumer-properties` - дополнительные параметры консьюмера запросов;
- `producer-properties` - дополнительные параметры продюсера ответов.

2.2.8.2.9 Секция delete

Секция `delete` хранит блок настроек механизма очистки.

Например:

```yaml
delete:
  # входящий топик запросов удаления ресурсов
  request-topic: ${kafka.internal.topic.prefix}tp.delete.tmp
  # максимальное количество обработчиков входящих запросов
  max-concurrent-handle: ${kafka.max-concurrent-handle}
  # периодичность фиксации оффсета обработанных сообщений
  commit-interval: ${kafka.commit-interval}
  # дополнительные параметры консьюмера запросов
  consumer-properties:
    group.id: ${kafka.internal.topic.prefix}import.tp.delete.consumer
```

Параметры настроек

- `request-topic` - топик запросов к сервису **MPPW**;
- `response-topic` - топик ответов от сервиса **MPPW**;
- `max-concurrent-handle` - максимальное количество обработчиков входящих запросов;
- `commit-interval`- периодичность фиксации оффсета обработанных сообщений;
- `consumer-properties` - дополнительные параметры консьюмера запросов.

2.2.8.2.10 Секция delta

Секция `delta` предназначена для настройки дельт.

Например:

```yaml
delta:
  # префикс топиков с данными
  data-topic-prefix: ${DATA_TOPIC_PREFIX:tp.data}
```

```yaml
  # признак необходимости удаления топиков
  delete-topics: ${IMPORT_DELTA_DELETE_TOPICS:true}
  # дополнительные параметры администрирования кафки
  admin-properties: [ ]
  # блок параметров поставщика данных
  provider:
    # топик запросов
    request-topic: ${kafka.internal.topic.prefix}tp.upload.delta
    # топик ответов
    response-topic: ${kafka.internal.topic.prefix}delta.rq
    # топик ошибок
    error-topic: ${kafka.internal.topic.prefix}tp.upload.err
    # максимальное количество обработчиков входящих запросов
    max-concurrent-handle: ${kafka.max-concurrent-handle}
    # периодичность фиксации оффсета обработанных сообщений
    commit-interval: ${kafka.commit-interval}
    # дополнительные параметры консьюмера запросов
    consumer-properties:
      group.id: ${kafka.internal.topic.prefix}import.tp.delta.consumer
    # дополнительные параметры продюсера ответов
    producer-properties: [ ]
  # блок параметров получателя данных
  recipient:
    # топик запросов
    request-topic: ${kafka.internal.topic.prefix}tp.upload.delta.in
    # топик ответов
    response-topic: ${kafka.internal.topic.prefix}delta.in
    # топик ошибок
    error-topic: ${kafka.internal.topic.prefix}tp.upload.in.err
    # максимальное количество обработчиков входящих запросов
    max-concurrent-handle: ${kafka.max-concurrent-handle}
    # периодичность фиксации оффсета обработанных сообщений
    commit-interval: ${kafka.commit-interval}
    # дополнительные параметры консьюмера запросов
    consumer-properties:
      group.id: ${kafka.internal.topic.prefix}import.tp.delta.in.consumer
    # дополнительные параметры продюсера ответов
    producer-properties: [ ]
```

Параметры настроек

- `data-topic-prefix` - префикс топиков с данными;
- `delete-topics` - признак необходимости удаления топиков;
- `admin-properties` - дополнительные параметры администрирования **Kafka**;
- `provider` - блок параметров поставщика данных;
- `recipient` - блок параметров получателя данных;
- `request-topic` - топик запросов;
- `response-topic` - топик запросов на исполнение после загрузки данных;
- `error-topic` - топик с ошибками;
- `max-concurrent-handle` - максимальное количество обработчиков входящих запросов;
- `commit-interval` - периодичность фиксации оффсета обработанных сообщений;
- `consumer-properties` - дополнительные параметры консьюмера запросов;
- `producer-properties` - дополнительные параметры продюсера ответов.

2.2.8.2.11 Секция metrics

Секция `metrics` предназначена для настройки параметров метрик.

Например:

```
metrics:
  # Порт сервера для получения метрик
  port: ${METRICS_PORT:9843}
  kafkaAdminClientName: kafkaAdminClient
```

Параметры настроек

– `port` - порт для метрик, например `METRICS_PORT:9843`.

2.2.9 Настройка СМЭВ4-адаптера – Модуль группировки данных табличных параметров

2.2.9.1 Конфигурация модуля СМЭВ4-адаптер – Модуль группировки данных табличных параметров (application.yml)

Файл `application.yml` – основной конфигурационный файл модуля, в котором задана его логика и порядок работы модуля.

2.2.9.1.1 Пример файла application.yml

В конфигурационном файле следует задавать только те настройки, которые необходимы для решения текущих бизнес-задач.

```
http-server:
  port: ${HTTP_PORT:8092}

kafka:
  agent.topic.prefix: ${AGENT_TOPIC_PREFIX:}
  # максимальное количество обработчиков входящих запросов
  max-concurrent-handle: ${KAFKA_MAX_CONCURRENT_HANDLE:1000}
  # периодичность фиксации оффсета обработанных сообщений
  commit-interval: ${KAFKA_COMMIT_INTERVAL:5s}
  external:
    bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
    topic.prefix: ${EXTERNAL_TOPIC_PREFIX:${kafka.agent.topic.prefix}}
  internal:
    bootstrap.servers: ${PS_KAFKA:localhost:9092}
    topic.prefix: ${INTERNAL_TOPIC_PREFIX:${kafka.agent.topic.prefix}}
  # параметры консьюмера сообщений
  consumer:
    bootstrap.servers: ${kafka.internal.bootstrap.servers}
    # Смещение при первом запуске приложения
    auto.offset.reset: earliest
  # параметры продъюсера сообщений
  producer:
    bootstrap.servers: ${kafka.internal.bootstrap.servers}
  # Параметры администрирования. Используется для health-check
  admin:
    bootstrap.servers: ${kafka.internal.bootstrap.servers}

group:
  data-topic-prefix: ${DATA_TOPIC_PREFIX:tp.data}
  # Время пересоздания kafka producer, в минутах
  refreshPeriod: ${DATA_PRODUCER_RECREATE_PERIOD:3m}

# блок настроек группировки данных табличных параметров
query:
  data-topic-prefix: ${DATA_TOPIC_PREFIX:tp.data}
  # топик с данными табличных параметров
  data-topic: ${kafka.external.topic.prefix:}query.tp
  # топик нотификаций на загрузку табличных параметров
  notification-topic: ${kafka.internal.topic.prefix:}tp.upload.query
```

```yaml
  # блок настроек отмены запросов
  cancel:
    # топик запросов на отмену исполнения
    request-topic: ${kafka.external.topic.prefix:}cancel.rq
    # настройки кеша отмененных запросов
    cache:
      # начальный размер кеша
      initial-capacity: 10000
      # время вытеснения из кеша
      expire-after-access: 60m
  # дополнительные параметры консьюмера запросов
  consumer-properties:
    bootstrap.servers: ${kafka.external.bootstrap.servers}
    group.id: ${kafka.external.topic.prefix}group.tp.consumer
  # дополнительные параметры продъюсера ответов
  producer-properties:
    bootstrap.servers: ${kafka.internal.bootstrap.servers}

delta:
  table-params:
    # топик с данными табличных параметров
    data-topic: ${kafka.external.topic.prefix}delta.tp
    # топик нотификаций на загрузку табличных параметров
    notification-topic: ${kafka.internal.topic.prefix}tp.upload.delta
    # топик с ошибками обработки
    error-topic: ${kafka.internal.topic.prefix}tp.upload.err
    # дополнительные параметры консьюмера запросов
    consumer-properties:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
      group.id: ${kafka.external.topic.prefix}delta.tp.consumer
    # дополнительные параметры продъюсера ответов
    producer-properties:
      bootstrap.servers: ${kafka.internal.bootstrap.servers}
  result:
    # топик с данными
    data-topic: ${kafka.external.topic.prefix}delta.in.tp
    # топик нотификаций на загрузку табличных параметров
    notification-topic: ${kafka.internal.topic.prefix}tp.upload.delta.in
    # топик с ошибками обработки
    error-topic: ${kafka.internal.topic.prefix}tp.upload.in.err
    # дополнительные параметры консьюмера запросов
    consumer-properties:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
      group.id: ${kafka.external.topic.prefix}delta.in.tp.consumer
    # дополнительные параметры продъюсера ответов
    producer-properties:
      bootstrap.servers: ${kafka.internal.bootstrap.servers}

environment:
  name: ${ENVIRONMENT_NAME:test}

zookeeper:
  connection-string: ${ZOOKEEPER_DS_ADDRESS:localhost}
  connection-timeout-ms: ${ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000}
  session-timeout-ms: ${ZOOKEEPER_DS_SESSION_TIMEOUT_MS:86400000}
  chroot: ${ZOOKEEPER_DS_CHROOT:/adapter}

metrics:
  # Порт сервера для получения метрик
  port: ${METRICS_PORT:9843}
  kafkaAdminClientName: kafkaAdminClient
```

2.2.9.2 Параметры конфигурации

Настройка конфигурации **СМЭВ4-адаптера – Модуль группировки данных табличных параметров** осуществляется путем редактирования параметров настроек в файле `application.yml`, где настраиваются секции:

- `kafkaUrl` – адрес сервера Kafka для Агента СМЭВ4;
- `http-server` - указывается порт веб-сервера;
- `kafka` – настройки сервера Kafka;
- `group` - настройка публикации сгруппированных чанков табличных параметров;
- `query` - блок настроек группировки данных табличных параметров;
- `delta` - указываются настройки группировки дельт;
- `environment` - название окружения (`test`, `prod` и т.д.);
- `zookeeper` - указываются настройки подключения к Zookeeper;
- `metrics` – настройка получения метрик.

2.2.9.2.1 Секция http-server

В секции `http-server` указывается порт веб-сервера.

Например:

```
http:
  port: ${HTTP_PORT:8092}
```

Параметры настроек

- `port` - порт веб-сервера, например: `HTTP_PORT:8092`.

2.2.9.2.2 Секция kafka

Секция `kafka` предназначена для настройки сервера Kafka.

Например:

```
kafka:
  # максимальное количество обработчиков входящих запросов
  max-concurrent-handle: ${KAFKA_MAX_CONCURRENT_HANDLE:1000}
  # периодичность фиксации оффсета обработанных сообщений
  commit-interval: ${KAFKA_COMMIT_INTERVAL:5s}
  external:
    bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
    topic.prefix: ${EXTERNAL_TOPIC_PREFIX:${agent.topic.prefix}}
  internal:
    bootstrap.servers: ${PS_KAFKA:localhost:9092}
    topic.prefix: ${INTERNAL_TOPIC_PREFIX:${agent.topic.prefix}}
  # параметры консьюмера сообщений
  consumer:
    bootstrap.servers: ${kafka.internal.bootstrap.servers}
    # Смещение при первом запуске приложения
    auto.offset.reset: earliest
  # параметры продюсера сообщений
  producer:
    bootstrap.servers: ${kafka.internal.bootstrap.servers}
  # Параметры администрирования. Используется для health-check
  admin:
    bootstrap.servers: ${kafka.internal.bootstrap.servers}
```

Параметры настроек

- `max-concurrent-handle` - максимальное количество обработчиков входящих запросов, например `{KAFKA_MAX_CONCURRENT_HANDLE:1000}`;
- `commit-interval` - периодичность фиксации оффсета обработанных сообщений,

например {KAFKA_COMMIT_INTERVAL:5s};
- eventLoopPool - размер пула event loop. например VERTX_POOL_EVENTLOOPPOOL:10;
- internal - настройка количества определяется адрес сервера Kafka и префикс топика;
- consumer - параметры потребителя сообщений;
- producer - параметры поставщика сообщений.

2.2.9.2.3 Секция group

Секция group отвечает за публикацию сгруппированных чанков табличных параметров.

```
group:
  data-topic-prefix: ${DATA_TOPIC_PREFIX:tp.data}
  # Время пересоздания kafka producer, в минутах
  refreshPeriod: ${DATA_PRODUCER_RECREATE_PERIOD:3m}
```

Параметры настроек

- refreshPeriod - Время пересоздания **kafka producer**, в минутах.

2.2.9.2.4 Секция query

Секция query предназначена для настроек группировки данных табличных параметров. Например:

```
query:
  data-topic-prefix: ${DATA_TOPIC_PREFIX:tp.data}
  # топик с данными табличных параметров
  data-topic: ${kafka.external.topic.prefix:}query.tp
  # топик нотификаций на загрузку табличных параметров
  notification-topic: ${kafka.internal.topic.prefix:}tp.upload.query
  # блок настроек отмены запросов
  cancel:
    # топик запросов на отмену исполнения
    request-topic: ${kafka.external.topic.prefix:}cancel.rq
    # настройки кеша отмененных запросов
    cache:
      # начальный размер кеша
      initial-capacity: 10000
      # время вытеснения из кеша
      expire-after-access: 60m
  # дополнительные параметры консьюмера запросов
  consumer-properties:
    bootstrap.servers: ${kafka.external.bootstrap.servers}
    group.id: ${kafka.external.topic.prefix}group.tp.consumer
  # дополнительные параметры продьюсера ответов
  producer-properties:
    bootstrap.servers: ${kafka.internal.bootstrap.servers}
```

Параметры настроек

- data-topic - топик с данными табличных параметров;
- notification-topic - топик нотификаций на загрузку табличных параметров;
- cancel - блок настроек отмены запросов;
- request-topic - Время пересоздания **kafka producer**, в минутах;
- cache - настройки кеша отмененных запросов;
- initial-capacity – начальный размер кеша;
- expire-after-access - время вытеснения из кеша;
- consumer-properties - дополнительные параметры потребителя запросов;
- producer-properties - дополнительные параметры поставщика ответов.

2.2.9.2.5 Секция delta

В секции `delta` указываются настройки группировки дельт.

Например:

```
delta:
  table-params:
    # топик с данными табличных параметров
    data-topic: ${kafka.external.topic.prefix}delta.tp
    # топик нотификаций на загрузку табличных параметров
    notification-topic: ${kafka.internal.topic.prefix}tp.upload.delta
    # топик с ошибками обработки
    error-topic: ${kafka.internal.topic.prefix}tp.upload.err
    # дополнительные параметры консьюмера запросов
    consumer-properties:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
      group.id: ${kafka.external.topic.prefix}delta.tp.consumer
    # дополнительные параметры продьюсера ответов
    producer-properties:
      bootstrap.servers: ${kafka.internal.bootstrap.servers}
  result:
    # топик с данными
    data-topic: ${kafka.external.topic.prefix}delta.in.tp
    # топик нотификаций на загрузку табличных параметров
    notification-topic: ${kafka.internal.topic.prefix}tp.upload.delta.in
    # топик с ошибками обработки
    error-topic: ${kafka.internal.topic.prefix}tp.upload.in.err
    # дополнительные параметры консьюмера запросов
    consumer-properties:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
      group.id: ${kafka.external.topic.prefix}delta.in.tp.consumer
    # дополнительные параметры продьюсера ответов
    producer-properties:
      bootstrap.servers: ${kafka.internal.bootstrap.servers}
```

Параметры настроек

- `data-topic` - топик с данными табличных параметров;
- `notification-topic` - топик нотификаций на загрузку табличных параметров;
- `error-topic` - топик с ошибками обработки;
- `consumer-properties` - дополнительные параметры потребителя запросов;
- `producer-properties` - дополнительные параметры поставщика ответов.

2.2.9.2.6 Секция environment

В секции `environment` указывается среда разработки (dev, test, stable, prod).

Например:

```
environment:
  name: ${ENVIRONMENT_NAME:test}
```

Параметры настроек

- `name` - среда разработки, например `ENVIRONMENT_NAME:test`.

2.2.9.2.7 Секция zookeeper

Секция `zookeeper` предназначена для настройки параметров подключения к серверу **Zookeeper**.

Например:

```
zookeeper:
  connection-string: ${ZOOKEEPER_DS_ADDRESS:t5-adsp-01.ru-central1.internal}
  connection-timeout-ms: ${ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000}
  session-timeout-ms: ${ZOOKEEPER_DS_SESSION_TIMEOUT_MS:86400000}
  chroot: ${ZOOKEEPER_DS_CHROOT:/adapter}
```

Параметры настроек

- `connection-string` - подключение к Zookeeper DS, например `ZOOKEEPER_DS_ADDRESS:t5-adsp-01.ru-central1.internal`;
- `connection-timeout-ms` - Zookeeper DS таймаут подключения, например `ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000`;
- `session-timeout-ms` - Zookeeper DS таймаут сессии, например `ZOOKEEPER_DS_SESSION_TIMEOUT_MS:86400000`;
- `chroot` - Zookeeper DS chroot path, например `ZOOKEEPER_DS_CHROOT:/adapter`.

2.2.9.2.8 Секция metrics

Секция `metrics` предназначена для настройки параметров метрик.

Например:

```
metrics:
  # Порт сервера для получения метрик
  port: ${METRICS_PORT:9843}
  kafkaAdminClientName: kafkaAdminClient
```

Параметры настроек

- `port` - порт для метрик, например `METRICS_PORT:9843`.

2.2.10 Настройка СМЭВ4-адаптера – Модуль дефрагментации чанков табличных параметров

2.2.10.1 Конфигурация модуля СМЭВ-адаптер - Модуль дефрагментации чанков табличных параметров (application.yml)

Файл `application.yml` – основной конфигурационный файл модуля, в котором задана его логика и порядок работы модуля.

2.2.10.1.1 Пример файла application.yml

В конфигурационном файле следует задавать только те настройки, которые необходимы для решения текущих бизнес-задач.

```
application:
  path: ${APPLICATION_PATH:/var/lib/podd-avro-defragmentator}
  destination_chunk_max_record_count:
${APPLICATION_DESTINATION_CHUNK_MAX_RECORD_COUNT:500}
  request_life_time: 24
  garbage_period: 24
  zookeeper:
    connectionString: ${APPLICATION_ZOOKEEPER_CONNECTIONSTRING:localhost}
    sessionTimeoutMs: ${APPLICATION_ZOOKEEPER_SESSIONTIMEOUTMS:30000}
    connectionTimeoutMs: ${APPLICATION_ZOOKEEPER_CONNECTIONTIMEOUTMS:86400000}
    chroot: ${APPLICATION_ZOOKEEPER_CHROOT:/adapter}

environment:
  name: ${ENVIRONMENT_NAME:test}

kafka:
  agent.topic.prefix: ${AGENT_TOPIC_PREFIX:}
```

```
  max-concurrent-handle: ${KAFKA_MAX_CONCURRENT_HANDLE:100}
  commit-interval: ${KAFKA_COMMIT_INTERVAL:5s}
  external:
    bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
    topic.prefix: ${EXTERNAL_TOPIC_PREFIX:${kafka.agent.topic.prefix}}
  consumer:
    bootstrap-servers: ${kafka.external.bootstrap.servers}
    group-id: ${KAFKA_CONSUMER_GROUPID:defragmentator-consumer}
    topic: ${kafka.external.topic.prefix}query.tp.bin
  producer:
    bootstrap-servers: ${kafka.external.bootstrap.servers}
    topic: ${kafka.external.topic.prefix}query.tp

logging:
  level:
    ru.itone.datamart: ${LOGGING_LEVEL_RUITONEDATAMART:debug}

metrics:
  port: ${METRICS_PORT:9837}
```

2.2.10.2 Параметры конфигурации

Настройка конфигурации **СМЭВ-адаптера - Модуль дефрагментации чанков табличных параметров** осуществляется путем редактирования параметров настроек в файле `application.yml`, где настраиваются секции:

- `application` - настройка модуля в части указания топиков взаимодействия и размера обрабатываемых данных;
- `environment` - указывается название окружения (`test`, `prod` и т.д.);
- `kafka` - настройка подключения к серверу Kafka для Поставщика и Получателя данных;
- `logging` - настройка параметров логирования модуля;
- `metrics` - настройка получения метрик.

2.2.10.2.1 Секция application

Секция `application` предназначена для настройки **СМЭВ-адаптера - Модуль дефрагментации чанков табличных параметров** в части указания топиков взаимодействия и размера обрабатываемых данных.

Например:

```
application:
  path: ${APPLICATION_PATH:/var/lib/podd-avro-defragmentator}
  destination_chunk_max_record_count:
${APPLICATION_DESTINATION_CHUNK_MAX_RECORD_COUNT:500}
  request_life_time: 24
  garbage_period: 24
  zookeeper:
    connectionString: ${APPLICATION_ZOOKEEPER_CONNECTIONSTRING:localhost}
    sessionTimeoutMs: ${APPLICATION_ZOOKEEPER_SESSIONTIMEOUTMS:30000}
    connectionTimeoutMs: ${APPLICATION_ZOOKEEPER_CONNECTIONTIMEOUTMS:86400000}
    chroot: ${APPLICATION_ZOOKEEPER_CHROOT:/adapter}
```

Параметры настроек

- `path` – рабочая папка на локальном диске, в которой будут создаваться подпапки для сохранения чанков, например `APPLICATION_PATH:/var/lib/podd-avro-defragmentator`;
- `destination_chunk_max_record_count` - максимальное количество строк в

сообщении для destination_topic, например APPLICATION_DESTINATION_CHUNK_MAX_RECORD_COUNT:500;
- request_life_time - время жизни запроса, пока идет сборка;
- garbage_period - время жизни отдельных, несобранных порций данных;
- connectionString - строка подключения к zookeeper, например APPLICATION_ZOOKEEPER_CONNECTIONSTRING:localhost;
- sessionTimeoutMs - таймаут сессии, например APPLICATION_ZOOKEEPER_SESSIONTIMEOUTMS:30000;
- connectionTimeoutMs - таймаут подключения, например APPLICATION_ZOOKEEPER_CONNECTIONTIMEOUTMS:86400000;
- chroot - путь до каталога сохранения в ZK, например APPLICATION_ZOOKEEPER_CHROOT:/test.

2.2.10.2.2 Секция environment

В секции environment указывается среда разработки (dev, test, stable, prod)
Например:

```
environment:
  name: ${ENVIRONMENT_NAME:test}
```

Параметры настроек

- name - среда разработки, например ENVIRONMENT_NAME:test.

2.2.10.2.3 Секция kafka

Секция kafka предназначена для настройки модуля в части указания адреса сервера Kafka для Поставщика данных (producer) и Получателя данных (consumer).

```
kafka:
  agent.topic.prefix: ${AGENT_TOPIC_PREFIX:}
  max-concurrent-handle: ${KAFKA_MAX_CONCURRENT_HANDLE:100}
  commit-interval: ${KAFKA_COMMIT_INTERVAL:5s}
  external:
    bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
    topic.prefix: ${EXTERNAL_TOPIC_PREFIX:${kafka.agent.topic.prefix}}
  consumer:
    bootstrap-servers: ${kafka.external.bootstrap.servers}
    group-id: ${KAFKA_CONSUMER_GROUPID:defragmentator-consumer}
    topic: ${kafka.external.topic.prefix}query.tp.bin
  producer:
    bootstrap-servers: ${kafka.external.bootstrap.servers}
    topic: ${kafka.external.topic.prefix}query.tp
```

Параметры настроек

- consumer - настройка адреса сервера **Kafka** для Получателя данных;
- bootstrap-servers - адрес кафки для консьюмеров, например KAFKA_CONSUMER_BOOTSTRAPSERVERS:localhost:9092;
- group-id - группа для консьюмеров;
- producer - настройка адреса сервера **Kafka** для Поставщика данных;
- bootstrap-servers - адрес кафки для продюсеров, например KAFKA_PRODUCER_BOOTSTRAPSERVERS:localhost:9092.

2.2.10.2.4 Секция logging

Секция logging предназначена для настройки параметров логирования.
Например:

```
logging:
  level:
    ru.itone.datamart: ${LOGGING_LEVEL_RUITONEDATAMART:debug}
```

2.2.10.2.5 Секция metrics

Секция `metrics` предназначена для настройки параметров метрик.

Например:

```
metrics:
  port: ${METRICS_PORT:9837}
```

Параметры настроек

– `port` - порт для метрик, например `METRICS_PORT:9837`.

2.2.11 Настройка Модуля группировки чанков

2.2.11.1 Конфигурация Модуля группировки чанков репликации (application.yml)

Файл `application.yml` – основной конфигурационный файл модуля, в котором описаны подключение к сервису **Kafka**, порт веб-сервера, и настройки журналирования запросов и ответов.

2.2.11.1.1 Пример файла application.yml

В конфигурационном файле следует задавать только те настройки, которые необходимы для решения текущих бизнес-задач.

```
http-server:
  port: ${HTTP_PORT:8084}

spring:
  liquibase:
    enabled: ${LIQUIBASE_ENABLED:true} # Для storage.type=kafka необходимо выключить
    change-log: classpath:/liquibase-changes/changelog.xml
    driver-class-name: org.postgresql.Driver
    url:
jdbc:postgresql://${storage.postgres.connection.host}:${storage.postgres.connection.por
t}/${storage.postgres.connection.database}?currentSchema=${storage.postgres.connection.
schema}
    user: ${storage.postgres.connection.user}
    password: ${storage.postgres.connection.password}

storage:
  type: ${STORAGE_TYPE:postgres} # тип, postgres|kafka
  postgres: # параметры подключения к базе. Используется только при type=postgres
    connection:
      database: ${STORAGE_DATABASE:replication}
      schema: ${STORAGE_SCHEMA:public}
      host: ${STORAGE_HOST:localhost}
      port: ${STORAGE_PORT:5432}
      user: ${STORAGE_USER:postgres}
      password: ${STORAGE_PASSWORD:postgres}
    pool:
      max-size: ${STORAGE_POOL_SIZE:30}

kafka:
  agent.topic.prefix: ${AGENT_TOPIC_PREFIX:}
  # максимальное количество обработчиков входящих запросов
  max-concurrent-handle: ${KAFKA_MAX_CONCURRENT_HANDLE:100}
```

```
  # периодичность фиксации оффсета обработанных сообщений
  commit-interval: ${KAFKA_COMMIT_INTERVAL:5s}
  external:
    bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
    topic.prefix: ${EXTERNAL_TOPIC_PREFIX:${kafka.agent.topic.prefix}}
  internal:
    bootstrap.servers: ${PS_KAFKA:localhost:9092}
    topic.prefix: ${INTERNAL_TOPIC_PREFIX:${kafka.agent.topic.prefix}}
  consumer:
    delta-apply-request: ${kafka.external.topic.prefix}delta.in.rq
    property:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
      group.id: ${kafka.external.topic.prefix}podd-adapter-group-repl
      auto.offset.reset: earliest
      enable.auto.commit: false
  producer:
    refresh-interval: 60s
    delta-apply-notification: ${kafka.internal.topic.prefix}subscription.in
    property:
      bootstrap.servers: ${kafka.internal.bootstrap.servers}

environment:
  name: ${ENVIRONMENT_NAME:test}

zookeeper:
  connection-string: ${ZOOKEEPER_DS_ADDRESS:localhost}
  connection-timeout-ms: ${ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000}
  session-timeout-ms: ${ZOOKEEPER_DS_SESSION_TIMEOUT_MS:8640000}
  chroot: ${ZOOKEEPER_DS_CHROOT:/adapter}

logging:
  scl.delta:
      enabled: ${SCL_DELTA_ENABLED:false}

metrics:
  port: ${METRICS_PORT:9837}
```

2.2.11.2 Параметры конфигурации

Настройка конфигурации **Модуля группировки чанков репликации** осуществляется путем редактирования параметров настроек в файле `application.yml`, где настраиваются секции:

- `http-server` - указывается порт веб-сервера.
- `storage` - блок настроек хранения чанков данных репликации. При изменении параметров необходимо синхронизировать аналогичные параметры в сервисе MPPW.
- `kafka` - настройки параметров подключения к шине данных Apache Kafka.
- `logging` - настройки журналирования запросов и ответов.
- `metrics` - настройка порта для получения метрик.

2.2.11.2.1 Секция http-server

В секции `http-server` указывается порт веб-сервера.

Например:

```
http-server:
  port: ${HTTP_PORT:8084}
```

Параметры настроек

– `port` - порт веб-сервера, например: `HTTP_PORT:8084`.

2.2.11.2.2 Секция storage

В секции `storage` указываются настройки хранения чанков данных репликации.

ри изменении параметров необходимо синхронизировать аналогичные параметры в сервисе MPPW.

Например:

```
storage:
  type: ${STORAGE_TYPE:postgres}
  postgres:
    connection:
      database: ${STORAGE_DATABASE:replication}
      schema: ${STORAGE_SCHEMA:public}
      host: ${STORAGE_HOST:localhost}
      port: ${STORAGE_PORT:5432}
      user: ${STORAGE_USER:postgres}
      password: ${STORAGE_PASSWORD:postgres}
    pool:
      max-size: ${STORAGE_POOL_SIZE:30}
```

Параметры настроек

– `type` - тип, postgres|kafka, например: `STORAGE_TYPE:postgres`;

– `connection` - параметры подключения к базе.

2.2.11.2.3 Секция kafka

Секция `kafka` определяет настройки взаимодействия через **СМЭВ4-адаптер** между Поставщиком данных (`producer`) и Получателем данных (`consumer`).

Здесь же задаются настройки параметров подключения к шине данных **Apache Kafka**.

Например:

```
kafka:
  agent.topic.prefix: ${AGENT_TOPIC_PREFIX:}
  max-concurrent-handle: ${KAFKA_MAX_CONCURRENT_HANDLE:100}
  commit-interval: ${KAFKA_COMMIT_INTERVAL:5s}
  external:
      bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
      topic.prefix: ${EXTERNAL_TOPIC_PREFIX:${kafka.agent.topic.prefix}}
  internal:
      bootstrap.servers: ${PS_KAFKA:localhost:9092}
      topic.prefix: ${INTERNAL_TOPIC_PREFIX:${kafka.agent.topic.prefix}}
  consumer:
    delta-apply-request: ${kafka.external.topic.prefix}delta.in.rq
    property:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
      group.id: ${kafka.external.topic.prefix}podd-adapter-group-repl
      auto.offset.reset: earliest
      enable.auto.commit: false
  producer:
    refresh-interval: 60s
    delta-apply-notification: ${kafka.internal.topic.prefix}subscription.in
    property:
      bootstrap.servers: ${kafka.internal.bootstrap.servers}
```

Параметры конфигурации

– `topic` - префикс для топиков агента СМЭВ4, например `AGENT_TOPIC_PREFIX`;

– `max-concurrent-handle` - максимальное количество обработчиков входящих запросов, например `KAFKA_MAX_CONCURRENT_HANDLE:100`;

- `commit-interval` - периодичность фиксации оффсета обработанных сообщений, например `KAFKA_COMMIT_INTERVAL:5s`.

2.2.11.2.4 Секция environment

В секции `environment` выбирается среда разработки (например, значение `test`, `prod` и т.д). Например:

```
environment:
       name: ${ENVIRONMENT_NAME:test}
```

Параметры конфигурации

- `name` - название окружения, например `ENVIRONMENT_NAME:test`.

2.2.11.2.5 Секция zookeeper

В секции `zookeeper` настраиваются параметры подключения к **Zookeeper**. Например:

```
zookeeper:
  connection-string: ${ZOOKEEPER_DS_ADDRESS:localhost}
  connection-timeout-ms: ${ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000}
  session-timeout-ms: ${ZOOKEEPER_DS_SESSION_TIMEOUT_MS:8640000}
  chroot: ${ZOOKEEPER_DS_CHROOT:/adapter}
```

Параметры настроек

- `connection-string` - Подключение к Zookeeper DS, например `ZZOOKEEPER_DS_ADDRESS:localhost`;
- `connection-timeout-ms` - Zookeeper DS таймаут подключения, например `ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000`;
- `session-timeout-ms` - Zookeeper DS таймаут сессии, например `ZOOKEEPER_DS_SESSION_TIMEOUT_MS:40000`;
- `chroot` - Zookeeper DS chroot path, например `ZOOKEEPER_DS_CHROOT:/adapter`.

2.2.11.2.6 Секция logging

Секция `logging` предназначена для настройки журналирования запросов и ответов. Например:

```
logging:
scl.delta:
    enabled: ${SCL_DELTA_ENABLED:false}
```

Параметры конфигурации

- `enabled` - Журналировать события SCL delta, например: `SCL_DELTA_ENABLED:false`.

LOG_FORMAT - Логирование в формате (JSON/TEXT) - указывается в `logback.xml`.

2.2.11.2.7 Секция metrics

Секция `metrics` предназначена для настроек порта получения метрик. Например:

```
metrics:
  port: ${METRICS_PORT:9837}
```

Параметры настроек

- `port` - Порт для получения метрик, например `{METRICS_PORT:9837}`.

2.2.12 Настройка DATA-Uploader – Модуль исполнения асинхронных заданий

2.2.12.1 Конфигурация модуля DATA-Uploader (application.yml)

Файл `application.yml` – основной конфигурационный файл модуля, в котором задана его логика и порядок работы модуля: обеспечение обработки очереди файлов, настройка подключения к ядру витрины (секция: `prostore`), а также другие настройки необходимые для корректной работы модуля.

2.2.12.1.1 Пример файла application.yml

В конфигурационном файле следует задавать только те настройки, которые необходимы для решения текущих бизнес-задач.

```
http-server:
  port: ${SERVER_PORT:8082}

persistence-mode: ${PERSISTENCE_MODE:prostore} # prostore, zookeeper

prostore-rest-client:
  persistence-datamart: ${PERSISTENCE_DATAMART:persistence}
  datasource: ${PERSISTENCE_DATASOURCE:} # по умолчанию пусто, тогда берется единственный датасорс из настроек Простора
  table-completed-inserts-datamarts:
${PERSISTENCE_COMPLETED_INSERTS_TABLE:data_uploader_completed_inserts_datamarts}
  table-deltas-open: ${PERSISTENCE_OPEN_DELTAS_TABLE:data_uploader_deltas_open}
  # Таблица активных экземляров сервиса data-uploader (используется при persistence-mode: prostore)
  table-data-uploader-health-check:
${PERSISTENCE_HEALTH_CHECK_TABLE:data_uploader_health_check}
  # Имя таблицы с задачами загрузки данных (используется при persistence-mode: prostore)
  table-validation-complete:
${PERSISTENCE_VALIDATION_COMPLETE_TABLE:validation_complete}
  # Имя таблицы хранения статусов запросов (используется при persistence-mode: prostore)
  table-requests-status: ${PERSISTENCE_STATUS_TABLE:status}
  # Имя таблицы хранения данных файлов (используется при persistence-mode: prostore)
  table-files: ${PERSISTENCE_FILES_TABLE:files}
  # Имя таблицы хранения активных экземпляров сервисов
  table-data-uploader-active: ${PERSISTENCE_ACTIVE_TABLE:data_uploader_active}
  health-check-refresh-period-sec: ${HEALTH_CHECK_REFRESH_PERIOD:120}
  health-check-wait-period-sec: ${HEALTH_CHECK_WAIT_PERIOD:300}
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9090}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}



redis:
  type: ${REDIS_TYPE:STANDALONE}
  connection-string: ${REDIS_CONNECTION_STRING:redis://localhost:6379}
  password: ${REDIS_PASSWORD:eYVX7EwVmmxKPCDmwMtyKVge8oLd2t81}
  max-pool-size: ${REDIS_MAX_POOL_SIZE:6}
  max-pool-waiting: ${REDIS_MAX_POOL_WAITING:24}
  max-waiting-handlers: ${REDIS_MAX_WAITING_HANDLERS:32}
net-client-options:
   tcp-user-timeout: 30
   idle-timeout: 30
```

```yaml
upload:
  concurrency: ${UPLOAD_CONCURRENCY:100} # Общее количество параллельных задач загрузки
  send-concurrency: ${UPLOAD_SEND_CONCURRENCY:10} # Количество параллельных отправок данных в кафку для каждого файла
  poll-interval: ${UPLOAD_POLL_INTERVAL:500ms} # Интервал опроса очереди сообщений на загрузку
  mode: ${UPLOAD_MODE:mppw} # mppw|llw
  llw-batch: ${UPLOAD_LLW_BATCH:1000} # Число записей в одной команде upsert/delete. Используется только при mode=llw
  creating-delta-on-upload-request: ${DELTA_CREATING_MODE:enable} # enable|disable|period Используется только при mode=llw
  period: ${CREATING_DELTA_ON_UPLOAD_REQUEST_PERIOD:300}s # Используется только при mode=llw и creating-delta-on-upload-request=period
  check-uniqueness: true # выполнять ли проверку уникальности первичных ключей
  pool:
    size: ${UPLOAD_POOL_SIZE:16}
  retry:
    attempts: 5
    delay: 1m


data-storage:
  compress: ${DATA_STORAGE_COMPRESS:zstd} # zstd/none  редактируется синхронно для модулей rest-uploader/data-uploader!!!
  type: ${DATA_STORAGE_TYPE:dir} # redis|dir|s3
  # Директория хранения файлов для типа dir
  dir: ${DATA_STORAGE_DIR:/tmp}
  s3:
    endpoint: ${S3_ENDPOINT:http://127.0.0.1:9000/}
    bucket: ${S3_BUCKET:data} # Имя бакета
    region: ${S3_REGION:}
    access-key: ${S3_USER:minioadmin} # Пользователь, под которым происходит взаимодействие с s3
    secret-key: ${S3_PASSWORD:minioadmin} # Пароль пользователя для взаимодействия с s3

environment:
  name: ${ENVIRONMENT_NAME:test}

zookeeper:
  connection-string: ${ZOOKEEPER_DS_ADDRESS:localhost}
  connection-timeout-ms: ${ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000}
  session-timeout-ms: ${ZOOKEEPER_DS_SESSION_TIMEOUT_MS:8640000}
  chroot: ${ZOOKEEPER_DS_CHROOT:/adapter}

csv-parser:
  separator: ${CSV_PARSER_SEPARATOR:;}
  quote-char: ${CSV_PARSER_QUOTE_CHAR:"}
  escape-char: ${CSV_PARSER_ESCAPE_CHAR:'}
  # Настройка интерпретации значений как null. Допустимые значения:
  #  - EMPTY_SEPARATORS - пустое значение между двумя разделителями, например ;;
  #  - EMPTY_QUOTES - пустые кавычки, например ;"";
  #  - BOTH - оба варианта
  #  - NEITHER - никогда. Пустая строка всегда определяется как пустая строка
  field-as-null: ${CSV_PARSER_FIELD_AS_NULL:EMPTY_SEPARATORS}
  # опция позволяет парсить мультистроковые значения

active:
  timeout: ${ACTIVE_TIMEOUT:60}
  time-to-live: ${ACTIVE_TTL:180}

delta:
```

```
  timeout: ${DELTA_TIMEOUT:300}
  row-max-count: ${DELTA_MAX_ROWS:500000}
  chunk-row-max-count: ${DELTA_CHUNK_ROWS:10000}
  chunk-data-max-size: ${DELTA_CHUNK_DATA_SIZE:1048576}
  # параметр ожидания перед повторной попыткой открытия дельты в случае ошибки
  open-delay: ${DELTA_OPEN_DELAY:60}s
  # количество попыток открытия дельты в случае ошибки
  open-attempts: ${DELTA_OPEN_ATTEMPTS:5}
  # период проверки открытых дельт
  open-check: ${DELTA_OPEN_CHECK:60}s
  # количество попыток фиксации дельты в случае ошибки
  commit-attempts: ${DELTA_COMMIT_ATTEMPTS:5}
  # период ожидания перед повторной попыткой фиксации дельты
  commit-error-delay: ${DELTA_COMMIT_DELAY:1}s
  # количество попыток отката дельты в случае ошибки
  rollback-attempts: ${DELTA_COMMIT_ATTEMPTS:3}
  # период ожидания перед повторной попыткой отката дельты
  rollback-error-delay: ${DELTA_COMMIT_DELAY:5}s


response:
  time-to-live: ${RESPONSE_TTL:36000}

kafka:
  agent.topic.prefix: ${AGENT_TOPIC_PREFIX:}
  internal:
    bootstrap.servers: ${PS_KAFKA:localhost:9092}
    topic.prefix: ${INTERNAL_TOPIC_PREFIX:${kafka.agent.topic.prefix}}
  mppw-consumer:
    property:
      bootstrap.servers: ${kafka.internal.bootstrap.servers}
      group.id: data.uploader.group
      auto.offset.reset: earliest
      enable.auto.commit: true
  mppw-producer:
    property:
      bootstrap.servers: ${kafka.internal.bootstrap.servers}
  data-producer:
    property:
      bootstrap.servers: ${kafka.internal.bootstrap.servers}
  data-topic-prefix: ${kafka.internal.topic.prefix}mppw.upload.data
  mppw-upload-rq-topic: ${kafka.internal.topic.prefix}mppw.upload.rq
  mppw-upload-rs-topic: ${kafka.internal.topic.prefix}mppw.upload.rs

metrics:
  port: ${METRICS_PORT:9837}
```

2.2.12.2 Параметры конфигурации

Настройка конфигурации **DATA-Uploader** осуществляется путем редактирования параметров настроек в файле `application.yml`, где настраиваются секции:

- `http-server` - указывается порт сервера;
- `persistence-mode` - настройки хранения данных;
- `prostore-rest-client` - блок параметров конфигурирования взаимодействия с **ProStore**.
- `redis` - настройка подключения к **Redis**;
- `upload` - указываются настройки параллельной загрузки данных;
- `data-storage` - директория хранения файлов для типа `dir`;
- `environment` - определяет значение среды разработки;

- `zookeeper` - определяет настройки подключения к **Zookeeper**;
- `csv-parser` - настройка парсера CSV-файлов;
- `active` - настройки интервалов между попытками перехода в активное состояние и времени жизни ключа флага активности;
- `delta` - настройка обработок загрузок и отправок заданий на загрузку дельт;
- `response` - настройка времени хранения ответов по заданию в **Redis**;
- `kafka` - настройка распределений отправки топиков;
- `metrics` - настройка получения метрик.

2.2.12.2.1 Секция http-server

В секции `http-server` указывается порт веб-сервера.

Например:

```
server:
  port: ${SERVER_PORT:8081}
```

Параметры настроек

- `port` - порт веб-сервера, например: `SERVER_PORT:8081`.

2.2.12.2.2 Секция persistence-mode

В секции `persistence-mode` указывается настройка хранения данных: или в Prostore или в Zookeeper. в случае выбора Prostore автоматически создаются необходимые таблицы.

Например:

```
persistence-mode: ${PERSISTENCE_MODE:prostore} # prostore | zookeeper
```

Параметры настроек

- `persistence-mode` - настройка хранения данных, например `PERSISTENCE_MODE:prostore`.

2.2.12.2.3 Секция prostore-rest-client

В секции `prostore-rest-client` реализован блок параметров конфигурирования взаимодействия с **Prostore**.

Например:

```
prostore-rest-client:
  persistence-datamart: ${PERSISTENCE_DATAMART:persistence}
  datasource: ${PERSISTENCE_DATASOURCE:} # по умолчанию пусто, тогда берется единственный датасорс из настроек Простора
  table-completed-inserts-datamarts:
${PERSISTENCE_COMPLETED_INSERTS_TABLE:data_uploader_completed_inserts_datamarts}
  table-deltas-open: ${PERSISTENCE_OPEN_DELTAS_TABLE:data_uploader_deltas_open}
  # Таблица активных экземляров сервиса data-uploader (используется при persistence-mode: prostore)
  table-data-uploader-health-check:
${PERSISTENCE_HEALTH_CHECK_TABLE:data_uploader_health_check}
  # Имя таблицы с задачами загрузки данных (используется при persistence-mode: prostore)
  table-validation-complete:
${PERSISTENCE_VALIDATION_COMPLETE_TABLE:validation_complete}
  # Имя таблицы хранения статусов запросов (используется при persistence-mode: prostore)
  table-requests-status: ${PERSISTENCE_STATUS_TABLE:status}
  # Имя таблицы хранения данных файлов (используется при persistence-mode: prostore)
  table-files: ${PERSISTENCE_FILES_TABLE:files}
  # Имя таблицы хранения активных экземпляров сервисов
  table-data-uploader-active: ${PERSISTENCE_ACTIVE_TABLE:data_uploader_active}
```

```
  health-check-refresh-period-sec: ${HEALTH_CHECK_REFRESH_PERIOD:120}
  health-check-wait-period-sec: ${HEALTH_CHECK_WAIT_PERIOD:300}
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9090}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}
```

Параметры настроек

- `persistence-datamart` - датамарт, где будут располагаться таблицы хранения данных, используется при `persistence-mode = prostore`
- `datasource` - источник данных, например `PERSISTENCE_DATASOURCE:`, по умолчанию пусто, в этом случае берется единственный датасорс из настроек Простора;
- `table-completed-inserts-datamarts` - таблица с данными по завершенным запросам добавления данных, например `PERSISTENCE_COMPLETED_INSERTS_TABLE:data_uploader_completed_inserts_datamarts` ;
- `table-deltas-open` - таблица с данными по открытым дельтам, например `PERSISTENCE_OPEN_DELTAS_TABLE:data_uploader_deltas_open`;
- `table-data-uploader-health-check` - таблица с heath-check, например `PERSISTENCE_HEALTH_CHECK_TABLE:data_uploader_health_check`;
- `table-validation-complete` - таблица с задачами загрузки данных, напрммер `PERSISTENCE_VALIDATION_COMPLETE_TABLE:validation_complete`;
- `table-requests-status` - таблица хранения статусов запросов. например `PERSISTENCE_STATUS_TABLE:status`;
- `table-files` - таблица хранения данных файлов, например `PERSISTENCE_FILES_TABLE:files`;
- `table-data-uploader-active` - таблица хранения активных экземпляров сервисов, например `PERSISTENCE_ACTIVE_TABLE:data_uploader_active`;
- `health-check-refresh-period-sec: ${HEALTH_CHECK_REFRESH_PERIOD:120}` - период обновления heath-check в секундах, например `HEALTH_CHECK_REFRESH_PERIOD:120`;
- `health-check-wait-period-sec` - период ожидания heath-check в секундах, например `HEALTH_CHECK_WAIT_PERIOD:300`;
- `host` - адрес **Prostore**, например `PS_HOST:t5-prostore-01.ru-central1.internal`;
- `port` - порт **Prostore**, например `PS_PORT:9195`;
- `max-pool-size` - максимальное число подключений к **Prostore**, например `PS_MAX_POOL_SIZE:8`.

2.2.12.2.4 Секция redis

Секция `redis` определяет настройки подключения к **Redis**.

Например:

```
redis:
  type: ${REDIS_TYPE:STANDALONE}
  connection-string: ${REDIS_CONNECTION_STRING:redis://localhost:6379}
  password: ${REDIS_PASSWORD:eYVX7EwVmmxKPCDmwMtyKVge8oLd2t81}
  max-pool-size: ${REDIS_MAX_POOL_SIZE:6}
  max-pool-waiting: ${REDIS_MAX_POOL_WAITING:24}
  max-waiting-handlers: ${REDIS_MAX_WAITING_HANDLERS:32}
  net-client-options:
    tcp-user-timeout: 30
    idle-timeout: 30
```

Параметры настроек

- `type` - тип подключения к **Redis** (STANDALONE/CLUSTER), например `REDIS_TYPE:STANDALONE`;
- `connection-string` - указывается список серверов для подключения (перечисление через запятую), например `REDIS_CONNECTION_STRING:redis://localhost:6379`;
- `password` - пароль для подключения, например `REDIS_PASSWORD:eYVX7EwVmmxKPCDmwMtyKVge8oLd2t81`;
- `max-pool-size` - устанавливается максимальный размер пула, например `REDIS_MAX_POOL_SIZE:6`;
- `max-pool-waiting` - устанавливается максимальный размер пула ожидающих команд, например `REDIS_MAX_POOL_WAITING:24`;
- `max-waiting-handlers` - устанавливается максимальный размер ожидающих обработчиков, например `REDIS_MAX_WAITING_HANDLERS:32`;
- `net-client-options` - параметры **Redis** клиента:
- `tcp-user-timeout` - таймаут на соединение;
- `idle-timeout` - таймаут ожидания ответа от редиса.

2.2.12.2.5 Секция upload

Секция `upload` определяет настройки параллельной загрузки данных.
Например:

```yaml
upload:
  concurrency: ${UPLOAD_CONCURRENCY:100} # Общее количество параллельных задач загрузки
  send-concurrency: ${UPLOAD_SEND_CONCURRENCY:10} # Количество параллельных отправок данных в кафку для каждого файла
  poll-interval: ${UPLOAD_POLL_INTERVAL:500ms} # Интервал опроса очереди сообщений на загрузку
  mode: ${UPLOAD_MODE:mppw} # mppw|llw
  llw-batch: ${UPLOAD_LLW_BATCH:1000} # Число записей в одной команде upsert/delete. Используется только при mode=llw
  creating-delta-on-upload-request: ${DELTA_CREATING_MODE:enable} # enable|disable|period Используется только при mode=llw
  period: ${CREATING_DELTA_ON_UPLOAD_REQUEST_PERIOD:300}s # Используется только при mode=llw и creating-delta-on-upload-request=period
  check-uniqueness: true # выполнять ли проверку уникальности первичных ключей
  pool:
    size: ${UPLOAD_POOL_SIZE:16}
  retry:
    attempts: 5
    delay: 1m
```

Параметры настроек

- `concurrency` - общее количество параллельных задач загрузки, например `100`;
- `send-concurrency` - количество параллельных отправок данных в кафку для каждого файла, например `10`;
- `poll-interval` - интервал опроса очереди сообщений на загрузку, например `500ms`;
- `mode` - режим загрузки, например `UPLOAD_MODE:mppw`;
  возможные значения: `mppw` / `llw`;
- `llw-batch` - число записей в одной команде upsert/delete, например `UPLOAD_LLW_BATCH:1000`, используется только при mode=llw;
- `creating-delta-on-upload-request` - создание дельты при запросе загрузки, например `DELTA_CREATING_MODE:enable`, используется только при mode=llw;

возможные значения: `enable`/ `disable` / `period`;

- `period` - период создания дельты при запросе загрузки (в секундах), например `CREATING_DELTA_ON_UPLOAD_REQUEST_PERIOD:300`, используется только при mode=llw и creating-delta-on-upload-request=period;
- `check-uniqueness` - флаг выполнения проверки уникальности первичных ключей;
- `pool/size` - размер пула загрузки, например `UPLOAD_POOL_SIZE:16`;
- `retry` - повтор попытки загрузки;
- `attempts` - количество попыток;
- `delay` - период задержки (в минутах).

2.2.12.2.6 Секция data-storage

В секции `data-storage` указывается директория хранения файлов для типа `dir`. Например:

```
data-storage:
  compress: ${DATA_STORAGE_COMPRESS:zstd} # zstd/none  редактируется синхронно для модулей rest-uploader/data-uploader!!!
  type: ${DATA_STORAGE_TYPE:dir} # redis|dir|s3
  # Директория хранения файлов для типа dir
  dir: ${DATA_STORAGE_DIR:/tmp}
  s3:
    endpoint: ${S3_ENDPOINT:http://127.0.0.1:9000/}
    bucket: ${S3_BUCKET:data} # Имя бакета
    region: ${S3_REGION:}
    access-key: ${S3_USER:minioadmin} # Пользователь, под которым происходит взаимодействие с s3
    secret-key: ${S3_PASSWORD:minioadmin} # Пароль пользователя для взаимодействия с s3
```

Параметры настроек

- `compress` - настройки сжатия директории хранения файлов, например `DATA_STORAGE_COMPRESS:zstd`.

> Внимание:
>
> блок `compress` редактируется синхронно для модулей rest-uploader/data-uploader

- `type` - тип файлов, например `DATA_STORAGE_TYPE:dir`;
  возможные значения: redis / dir / s3;
- `dir` - директория хранения файлов для типа dir, например `DATA_STORAGE_DIR:/tmp`;
- `s3` - настройки облачного хранилища S3;
- `endpoint` - адрес конечной точки, например `S3_ENDPOINT:http://127.0.0.1:9000/`;
- `bucket` - имя бакета, например `S3_BUCKET:data`
- `region` - регион хранилища;
- `access-key` - пользователь, под которым происходит взаимодействие с s3, например `S3_USER:minioadmin`;
- `secret-key`- пароль пользователя для взаимодействия с s3, например `S3_PASSWORD:minioadmin`.

2.2.12.2.7 Секция environment

В секции `environment` выбирается среда разработки (например, значение `test`, `prod` и т.д). Например:

```
environment:
       name: ${ENVIRONMENT_NAME:test}
```

Параметры конфигурации

- `name` - название окружения, например `ENVIRONMENT_NAME:test`.

2.2.12.2.8 Секция zookeeper

В секции `zookeeper` настраиваются параметры подключения к **Zookeeper**.

Например:

```
zookeeper:
  connection-string: ${ZOOKEEPER_DS_ADDRESS:localhost}
  connection-timeout-ms: ${ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000}
  session-timeout-ms: ${ZOOKEEPER_DS_SESSION_TIMEOUT_MS:8640000}
  chroot: ${ZOOKEEPER_DS_CHROOT:/adapter}
```

Параметры настроек

- `connection-string` - Подключение к Zookeeper DS, например `ZZOOKEEPER_DS_ADDRESS:localhost`;
- `connection-timeout-ms` - Zookeeper DS таймаут подключения, например `ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000`;
- `session-timeout-ms` - Zookeeper DS таймаут сессии, например `ZOOKEEPER_DS_SESSION_TIMEOUT_MS:40000`;
- `chroot` - Zookeeper DS chroot path, например `ZOOKEEPER_DS_CHROOT:/adapter`.

2.2.12.2.9 Секция csv-parser

> Внимание:
>
> При загрузке файлов с форматно-логическим контролем, важно, чтобы настройки секции csv-parser были одинаковы в модулях CSV-Uploader(если используется его UI),REST-Uploader и DATA-Uploader.

Секция `csv-parser` - настройка парсинга CSV.

Например:

```
csv-parser:
  separator: ${CSV_PARSER_SEPARATOR:;}
  quote-char: ${CSV_PARSER_QUOTE_CHAR:"}
  escape-char: ${CSV_PARSER_ESCAPE_CHAR:'}
  field-as-null: ${CSV_PARSER_FIELD_AS_NULL:EMPTY_SEPARATORS}
  multiline-limit: 0
```

Параметры конфигурации

- `separator` - Символ разделителя значений, например `CSV_PARSER_SEPARATOR:;`;
- `quote-char` - символ кавычки, например `CSV_PARSER_QUOTE_CHAR:"`;
- `escape-char` - Символ экранирования значений, например `CSV_PARSER_ESCAPE_CHAR:'`;

  Настройка интерпретации значений как null. Допустимые значения:
  - EMPTY_SEPARATORS - пустое значение между двумя разделителями, например ;;
  - EMPTY_QUOTES - пустые кавычки, например ;»»;
  - BOTH - оба варианта
  - NEITHER - никогда. Пустая строка всегда определяется как пустая строка
- `field-as-null` - способ определения null поля, например `CSV_PARSER_FIELD_AS_NULL:EMPTY_SEPARATORS`.
- `multiline-limit` - параметры парсинга мультистроковых значений.

Внимание:

Парсер может зависать если в данных встречаются вложенные неэкранированные кавычки..

1. Параметр `CSV_PARSER_ESCAPE_CHAR` работает следующим образом: если символ экранирования и символ кавычки равны `"`, то будет использован **RFC4180Parser**, который считывает все символы между двумя двойными кавычками, при этом двойная кавычка в тексте поля должна быть экранирована двойной кавычкой (Например `"поле, ""содержащее двойную кавычку"""` будет считано как `поле, "содержащее двойную кавычку"`). В противном случае будет использован **CSVParser**, использующий символ экранирования для обозначения «непечатаемых символов».
2. Параметр `CSV_PARSER_FIELD_AS_NULL` может принимать следующие значения:
   - `EMPTY_SEPARATORS` - два разделителя полей (см. csv-parser/separator) подряд считаются null. Например: строка [aaa,,ccc] содержит значения [«aaa», null, «bbb»], а строка [aaa,»»,ccc] содержит значения [«aaa», «», «bbb»].
   - `EMPTY_QUOTES` - два «ограничителя строки» (см. csv-parser/escape-char) подряд считаются null. Например: строка [aaa,»»,ccc] содержит значения [«aaa», null, «bbb»], а строка [aaa,,ccc] содержит значения [«aaa», «», «bbb»].
   - `BOTH` - оба варианта (см. `EMPTY_SEPARATORS` и `EMPTY_QUOTES`) считаются null. Например: обе строки [aaa,»»,ccc] и [aaa,,bbb] содержат одинаковое значение [«aaa», null, «bbb»].
   - `NEITHER` - ни один из вариантов (см. `EMPTY_SEPARATORS` и `EMPTY_QUOTES`) не считается null. Например: обе строки [aaa,»»,ccc] и [aaa,,bbb] содержат одинаковое значение [«aaa», «», «bbb»].

2.2.12.2.10 Секция active

В секции `active` настраиваются интервалы между попытками перехода в активное состояние и времени жизни ключа флага активности.

Например:

```
active:
    timeout: ${ACTIVE_TIMEOUT:60}
    time-to-live: ${ACTIVE_TTL:180}
```

Параметры настроек

- `timeout` - интервал между попытками перехода в активное состояние;
- `time-to-live` - время жизни ключа флага активности.

2.2.12.2.11 Секция delta

Секция `delta` предназначена для указания настройки обработок загрузок и отправок заданий на загрузку дельт.

Например:

```
delta:
  timeout: ${DELTA_TIMEOUT:300}
  row-max-count: ${DELTA_MAX_ROWS:500000}
  chunk-row-max-count: ${DELTA_CHUNK_ROWS:10000}
  chunk-data-max-size: ${DELTA_CHUNK_DATA_SIZE:1048576}
  # параметр ожидания перед повторной попыткой открытия дельты в случае ошибки
  open-delay: ${DELTA_OPEN_DELAY:60}s
  # количество попыток открытия дельты в случае ошибки
  open-attempts: ${DELTA_OPEN_ATTEMPTS:5}
  # период проверки открытых дельт
  open-check: ${DELTA_OPEN_CHECK:60}s
  # количество попыток фиксации дельты в случае ошибки
  commit-attempts: ${DELTA_COMMIT_ATTEMPTS:5}
  # период ожидания перед повторной попыткой фиксации дельты
  commit-error-delay: ${DELTA_COMMIT_DELAY:1}s
  # количество попыток отката дельты в случае ошибки
  rollback-attempts: ${DELTA_COMMIT_ATTEMPTS:3}
  # период ожидания перед повторной попыткой отката дельты
  rollback-error-delay: ${DELTA_COMMIT_DELAY:5}s
```

Параметры настроек

- `timeout` - максимальный интервал времени между первой считанной записью цикла загрузки и отправкой заданий на загрузку дельт, например `DELTA_TIMEOUT:300`;
- `row-max-count` - максимальное количество обработанных записей для старта отправки заданий на загрузку дельт, например `DELTA_CHUNK_ROWS:10000`;
- `chunk-row-max-count` - количество сообщений в одном чанке, например `DELTA_CHUNK_ROWS:10000`;
- `chunk-data-max-size` - максимальный размер чанка, например `DELTA_CHUNK_DATA_SIZE:1048576`;
- `open-delay` - параметр ожидания (в секундах) перед повторной попыткой открытия дельты в случае ошибки, например `DELTA_OPEN_DELAY:60`;
- `open-attempts` - количество попыток открытия дельты в случае ошибки, например `DELTA_OPEN_ATTEMPTS:5`;
- `open-check` - период проверки открытых дельт (в секундах), например `DELTA_OPEN_CHECK:60`;
- `commit-attempts` - количество попыток фиксации дельты в случае ошибки, например `DELTA_COMMIT_ATTEMPTS:5`;
- `commit-error-delay` - период ожидания перед повторной попыткой фиксации дельты (в секундах), например `DELTA_COMMIT_DELAY:1`;
- `rollback-attempts` - количество попыток отката дельты в случае ошибки, например `DELTA_COMMIT_ATTEMPTS:3`;
- `rollback-error-delay` - период ожидания перед повторной попыткой отката дельты (в секундах), например `DELTA_COMMIT_DELAY:5`.

2.2.12.2.12 Секция response

Секция `response` определяет настройка времени хранения ответов по заданию в **Redis**. Например:

```
response:
    time-to-live: ${RESPONSE_TTL:36000}
```

Параметры настроек

- `time-to-live` - Сколько времени **Redis** будет хранить ответ по заданию (ключ status.<uuid>).

2.2.12.2.13 Секция kafka

Секция `kafka` позволяет настраивать отправку топиков в заданные модули.

Например:

```
kafka:
  agent.topic.prefix: ${AGENT_TOPIC_PREFIX:}
  internal:
    bootstrap.servers: ${PS_KAFKA:localhost:9092}
    topic.prefix: ${INTERNAL_TOPIC_PREFIX:${kafka.agent.topic.prefix}}
  mppw-consumer:
    property:
      bootstrap.servers: ${kafka.internal.bootstrap.servers}
      group.id: data.uploader.group
      auto.offset.reset: earliest
      enable.auto.commit: true
  mppw-producer:
    property:
      bootstrap.servers: ${kafka.internal.bootstrap.servers}
  data-producer:
    property:
      bootstrap.servers: ${kafka.internal.bootstrap.servers}
  data-topic-prefix: ${kafka.internal.topic.prefix}mppw.upload.data
  mppw-upload-rq-topic: ${kafka.internal.topic.prefix}mppw.upload.rq
  mppw-upload-rs-topic: ${kafka.internal.topic.prefix}mppw.upload.rs
```

Параметры настроек

- `data-topic-prefix` - префикс топика, куда будут отправляться данные;
- `mppw-upload-rq-topic` - топик для отправки заданий на MPPW модуль;
- `mppw-upload-rq-topic` - топик, в который будут приходить результаты исполнения заданий MPPW модуля.

2.2.12.2.14 Секция metrics

Секция `metrics` предназначена для настройки параметров метрик.

Например:

```
metrics:
  port: ${METRICS_PORT:9837}
```

Параметры настроек

- `port` - порт для метрик, например `METRICS_PORT:9837`.

2.2.13 Настройка REST-Uploader – Модуль асинхронной загрузки данных из сторонних источников

2.2.13.1 Конфигурация модуля REST-Uploader (application.yml)

Файл `application.yml` – основной конфигурационный файл модуля, в котором задана его логика и порядок работы модуля: асинхронная загрузка данных из источников, настройка подключения к ядру витрины (секция: `prostore`), а также другие настройки необходимые для корректной работы модуля.

2.2.13.1.1 Пример файла application.yml

В конфигурационном файле следует задавать только те настройки, которые необходимы для решения текущих бизнес-задач.

```yaml
http-server:
  port: ${SERVER_PORT:8081}

executor:
  reader-pool-size: ${EXECUTOR_READER_POOL_SIZE:20}

file-size:
  restriction: ${SEND_FILE_SIZE_RESTRICTION:512}

environment:
  # Название окружения
  name: ${ENVIRONMENT_NAME:test}

data-storage:
  compress: ${DATA_STORAGE_COMPRESS:zstd} # zstd/none  редактируется синхронно для модулей rest-uploader/data-uploader!!!
  compression-ratio: ${DATA_STORAGE_COMPRESSION_RATIO:4} # допустимый диапазон [-7;22] где -7 самый быстрый, а 22 наибольшее сжатие
  type: ${DATA_STORAGE_TYPE:dir} # redis|dir|s3
  # Директория хранения файлов для типа dir
  dir: ${DATA_STORAGE_DIR:/tmp}
  s3:
    endpoint: ${S3_ENDPOINT:http://127.0.0.1:9000/}
    region: ${S3_REGION:}
    bucket: ${S3_BUCKET:data} # Имя бакета
    access-key: ${S3_USER:minioadmin} # Пользователь, под которым происходит взаимодействие с s3
    secret-key: ${S3_PASSWORD:minioadmin} # Пароль пользователя для взаимодействия с s3

conditions:
  # включение ФЛК и поведение при обнаружении ошибок
  mode: warning
  # период хранения журналов ошибок
  save-time: 1d
  # путь хранения журналов ошибок на общем диске:
  save-path: /tmp/rest-uploader/reports
  # путь к хранению правил в Zookeeper
  zookeeper-path: rest-uploader/conditions
  # таймаут обработки запроса. 0 - таймаут отключен
  rest-timeout: 60s
  # период жизни группы
  save_group_time: 1d
  validation:
    # таймаут выполнения асинхронной проверки
    validation-timeout: 1h
    # таймаут получения сообщений из redis
    poll-timeout: 30s
    # количество корутин асинхронной валидации
    max-concurrent-handle: 1
    # размер порции вычитки сообщений из redis
    batch-size: 1
    # признак выполнения проверки кодировки
    charset-check-enabled: true
    # допустимые кодировки для механизма автоопределения
    valid-charsets: [ UTF-8, US-ASCII, TIS-620]
  health-check:
    # период жизни флага активности
    timeout-active: 3m
    # период обновления статуса
    publish-period: 30s
```

```yaml
zookeeper:
  connection-string: ${ZOOKEEPER_DS_ADDRESS:localhost}
  connection-timeout-ms: ${ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000}
  session-timeout-ms: ${ZOOKEEPER_DS_SESSION_TIMEOUT_MS:40000}
  chroot: ${ZOOKEEPER_DS_CHROOT:/adapter}
  retry-count: 3
  retry-base-sleep-time-ms: 1_000


persistence-mode: ${PERSISTENCE_MODE:prostore} # prostore, zookeeper

prostore-rest-client: # требуется синхронно менять в приложениях rest-uploader и data-uploader !!!
  persistence-datamart: ${PERSISTENCE_DATAMART:persistence}
  datasource: ${PERSISTENCE_DATASOURCE:} # по умолчанию пусто, тогда берется единственный датасорс из настроек Простора
  table-conditions: ${PERSISTENCE_TABLE_CONDITIONS:rest_uploader_conditions}
  table-ids: ${PERSISTENCE_TABLE_IDS:rest_uploader_ids}
  # Таблица активных экземляров сервиса rest-uploader (используется при persistence-mode: prostore)
  table-rest-uploader-health-check:
${PERSISTENCE_TABLE_HEALTH_CHECK:rest_uploader_health_check}
  # Таблица групп файлов (используется при persistence-mode: prostore)
  table-group-info: ${PERSISTENCE_TABLE_GROUP_INFO:group_info}
  # Таблица данных о файлах в группе (используется при persistence-mode: prostore)
  table-group-content: ${PERSISTENCE_TABLE_GROUP_CONTENT:group_content}
  # Таблица уникальных значений полей в группе (используется при persistence-mode: prostore)
  table-group-uniq: ${PERSISTENCE_TABLE_GROUP_UNIQ:group_uniq}
  # Таблица очереди файлов на ФЛК (используется при persistence-mode: prostore)
  table-validation-queue: ${PERSISTENCE_TABLE_VALIDATION_QUEUE:validation_queue}
  # Таблица очереди файлов на загрузку (используется при persistence-mode: prostore)
  table-validation-complete:
${PERSISTENCE_TABLE_VALIDATION_COMPLETE:validation_complete}
  # Таблица статусов запросов (используется при persistence-mode: prostore)
  table-requests-status: ${PERSISTENCE_TABLE_STATUS:status}
  # Таблица с файлами (используется при persistence-mode: prostore)
  table-files: ${PERSISTENCE_TABLE_FILES:files}
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9090}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}

response:
  time-to-live: ${RESPONSE_TTL:10h} # Период хранения информации о статусе запроса


control:
  delta:
    # подключение возможности управления дельтами от клиента. Управление дельтами от клиента допускается только при настройках
    # delta-> creating-delta-on-upload-request=disable у модулей data-uploader/podd-adapter-mppw
    enable: ${CONTROL_DELTA_ENABLE:true}


redis:
  type: ${REDIS_TYPE:STANDALONE}
  connection-string: ${REDIS_CONNECTION_STRING:redis://localhost:6379}
  password: ${REDIS_PASSWORD:eYVX7EwVmmxKPCDmwMtyKVge8oLd2t81}
  max-pool-size: ${REDIS_MAX_POOL_SIZE:6}
  max-pool-waiting: ${REDIS_MAX_POOL_WAITING:24}
```

```
  max-waiting-handlers: ${REDIS_MAX_WAITING_HANDLERS:32}
  net-client-options:
    tcp-user-timeout: 30
    idle-timeout: 30


auth:
  secret: ${AUTH_SECRET:gPHaT%ACXGQaQ30%1id%K7@C}
  enabled: ${AUTH_ENABLED:true}
  access-list-path: rest-uploader/ids

metrics:
  port: ${METRICS_PORT:9837}

# Настройки парсера csv файлов
csv-parser:
  # Символ разделителя значений
  separator: ${CSV_PARSER_SEPARATOR:;}
  # Символ кавычки
  quote-char: ${CSV_PARSER_QUOTE_CHAR:"}
  # Символ экранирования значений
  escape-char: ${CSV_PARSER_ESCAPE_CHAR:'}
  # Настройка интерпретации значений как null. Допустимые значения:
  #  - EMPTY_SEPARATORS - пустое значение между двумя разделителями, например ;;
  #  - EMPTY_QUOTES - пустые кавычки, например ;"";
  #  - BOTH - оба варианта
  #  - NEITHER - никогда. Пустая строка всегда определяется как пустая строка
  field-as-null: ${CSV_PARSER_FIELD_AS_NULL:EMPTY_SEPARATORS}

backup:
  mode: ${BACKUP_MODE:rest} # kafka | rest
  rest:
    uri: ${BACKUP_URI:/backup}
  zk-path: ${REST_UPLOADER_BACKUP_ZK_PATH:/${environment.name}/rest-uploader}
  commandTopic: ${BACKUP_COMMAND_TOPIC:adapter.command}
  backupTopic: ${BACKUP_TOPIC:adapter.backup}
  statusTopic: ${STATUS_TOPIC:adapter.status}
  kafka:
    consumer:
      property:
        bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
        group.id: ${REST_UPLOADER_BACKUP_GROUP_ID:rest-uploader_adapter_command}
        auto.offset.reset: latest
    producer:
      property:
        bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}

# Настройки модуля сбора информации о компонентах витрины
component-info:
  enabled: true
  # DataSource Prostore
  datasource: ''
  # Схема Prostore
  datamart: component_info
  # Имя таблицы для записи информации о компоненте
  table-name: component_info
  # Период попыток создания схемы, при неуспехе
  create-table-period: 60s
  # Период публикации health-check
  publish-period: 60s
  # Период после которого компонент считается неактивным при отсутствии health-
check
```

```
  timeout-active: 300s
  # Список элементов конфига маскируемых при отправке,
  #  если указан узловой элемент, то маскируются все вложенные в него элементы
  secrets:
    - redis.password
```

2.2.13.2 Параметры конфигурации

Настройка конфигурации **REST-uploader** осуществляется путем редактирования параметров настроек в файле `application.yml`, где настраиваются секции:

- `http-server` - указывается порт сервера;
- `executor` - настраивается размер пула для запросов;
- `file-size` - настраиваются ограничения на размер загружаемого файла;
- `environment` - настройки окружения;
- `data-storage` - директория хранения файлов для типа `dir`;
- `conditions` - включение форматно-логического контроля и поведение при обнаружении ошибок;
- `zookeeper` - настройки подключения к Zookeeper;
- `persistence-mode` - настройки хранения данных;
- `prostore-api-client` - блок параметров конфигурирования взаимодействия с **Prostore**;
- `response` - настройка периода хранения информации о статусе запроса;
- `control` - подключение возможности управления дельтами от клиента;
- `redis` - настройка подключения к **Redis**;
- `auth`- указывается секрет для валидации токенов;
- `metrics` - настройка получения метрик;
- `csv-parser` - настройка парсинга CSV;
- `backup` - настройки бекапирования;
- `component-info` - настройки модуля сбора информации о компонентах витрины.

2.2.13.2.1 Секция http-server

В секции `http-server` указывается порт веб-сервера.

Например:

```
http-server:
  port: ${SERVER_PORT:8081}
```

Параметры настроек

- `port` - порт веб-сервера, например: `SERVER_PORT:8081`.

2.2.13.2.2 Секция executor

Секция `executor` предназначена для указания размера пула для запросов.

Например:

```
executor:
  reader-pool-size: ${EXECUTOR_READER_POOL_SIZE:20}
```

Параметры настроек

- `reader-pool-size` - Размер пула для запросов, например `EXECUTOR_READER_POOL_SIZE:20`

2.2.13.2.3 Секция file-size

В секции `file-size` можно настраивать ограничения на размер загружаемого файла.

Например:

```
file-size:
  restriction: ${SEND_FILE_SIZE_RESTRICTION:512}
```

Параметры настроек

– `file-size-restriction` - ограничение на размер загружаемого файла, например `SEND_FILE_SIZE_RESTRICTION:512`.

2.2.13.2.4 Секция environment

Секция `environment` предназначена для настройки среды окружения.

Например:

```
environment:
  name: ${ENVIRONMENT_NAME:test}
```

Параметры настроек

– `name` - Название окружения, например `ENVIRONMENT_NAME:test`.

2.2.13.2.5 Секция data-storage

В секции `data-storage` указывается директория хранения файлов для типа `dir`.

Например:

```
data-storage:
  compress: ${DATA_STORAGE_COMPRESS:zstd} # zstd/none
  compression-ratio: ${DATA_STORAGE_COMPRESSION_RATIO:4} # допустимый диапазон [-7;22] где -7 самый быстрый, а 22 наибольшее сжатие
  type: ${DATA_STORAGE_TYPE:dir} # redis|dir|s3
  # Директория хранения файлов для типа dir
  dir: ${DATA_STORAGE_DIR:/tmp}
  s3:
    endpoint: ${S3_ENDPOINT:http://127.0.0.1:9000/}
    bucket: ${S3_BUCKET:data} # Имя бакета
    region: ${S3_REGION:}
    access-key: ${S3_USER:minioadmin} # Пользователь, под которым происходит взаимодействие с s3
    secret-key: ${S3_PASSWORD:minioadmin} # Пароль пользователя для взаимодействия с s3
```

Параметры настроек

– `compress` - настройки сжатия директории хранения файлов, например `DATA_STORAGE_COMPRESS:zstd`

> Внимание:
>
> блок `compress` редактируется синхронно для модулей rest-uploader/data-uploader

– `type` - тип файлов, например `DATA_STORAGE_TYPE:dir`;
возможные значения: redis / dir / s3;

– `dir` - директория хранения файлов для типа dir, например `DATA_STORAGE_DIR:/tmp`;

– `s3` - настройки облачного хранилища S3;

– `endpoint` - адрес конечной точки, например `S3_ENDPOINT:http://127.0.0.1:9000/`;

– `bucket` - имя бакета, например `S3_BUCKET:data`

– `region` - регион хранилища;

– `access-key` - пользователь, под которым происходит взаимодействие с s3, например `S3_USER:minioadmin`;

– `secret-key`- пароль пользователя для взаимодействия с s3, например `S3_PASSWORD:minioadmin`.

2.2.13.2.6 Секция conditions

В секции `conditions` - реализована возможность включения форматно-логического контроля и настройки поведения при обнаружении ошибок.

Например:

```
conditions:
  mode: warning
  save-time: 1d
  save-path: /tmp/rest-uploader/reports
  zookeeper-path: rest-uploader/conditions
  rest-timeout: 60s
  save_group_time: 1d
  validation:
    validation-timeout: 1h
    poll-timeout: 30s
    max-concurrent-handle: 1
    batch-size: 1
    charset-check-enabled: true
  health-check:
    timeout-active: 3m
    publish-period: 30s
```

Параметры настроек

- `mode` - включение ФЛК и поведение при обнаружении ошибок, например `warning`;
- `save-time` - период хранения журналов ошибок, например `1d`;
- `save-path` - путь хранения журналов ошибок на общем диске, например `/tmp/rest-uploader/reports`;
- `zookeeper-path` - путь к хранению правил в Zookeeper, например `rest-uploader/conditions`;
- `rest-timeout` - таймаут обработки запроса, например `60s`, `0` - таймаут отключен;
- `save_group_time` - период жизни группы, например `1d`;
- `validation-timeout` - таймаут выполнения асинхронной проверки, например `1h`;
- `poll-timeout` - таймаут получения сообщений из redis, например `30s`;
- `max-concurrent-handle` - количество корутин асинхронной валидации, например `1`;
- `batch-size` - размер порции вычитки сообщений из redis, например `1`;
- `charset-check-enabled` - признак выполнения проверки кодировки, например `true`;
- `timeout-active` - период жизни флага активности, например `3m`;
- `publish-period` - период обновления статуса, например `30s`;

Для настройки `mode` реализованы режимы:

`skip_string`

1. При необходимости пропуска строк формируется обновленный файл и сохраняется в queue с прежним ключом, после проверки всего файла;
2. Подтверждается чтение комплексной записи из stream `validation_queue`, которая затем удаляется из стрима;
3. Размещается запись в list с именем `validation_complete`;
4. В Redis записывается статус 0 – Буферизирован;
5. В зависимости от наличия `group_id` в комплексной записи:
   - при отсутствии `group_id` – завершается обработка файла;
   - при наличии `group_id` – обновляется статус в groupContent_[group_id] и выполняется проверка комплектности группы в соответствии с шагами 3-6 Журнал по группе файлов.

warning

1. Подтверждается чтение комплексной записи из stream validation_queue и удаляется из стрима;
2. Размещается запись в list с именем validation_complete;
3. В Redis записывается статус 0- Буферизирован;
4. В зависимости от наличия group_id в комплексной записи:
   - при отсутствии group_id - завершает обработку файла;
   - при наличии group_id - обновляет статус в groupContent_[group_id] и выполняет проверка комплектности группы в соответствии с шагами 3-6 Журнал по группе файлов.

skip_file/ skip_on_first_error:

1. Подтверждается чтение комплексной записи из stream validation_queue, которая затем удаляется из стрима;
2. Удаляется файл из hset queue по ключу;
3. В Redis записывает статус 7 – Ошибки ФЛК;
4. В зависимости от наличия group_id в комплексной записи:
   - при отсутствии group_id – завершается обработка файла;
   - при наличии group_id – обновляется статус в groupContent_[group_id] и проверяется комплектность группы в соответствии с шагами 3-6 Журнал по группе файлов.

skip_string_except_last пропускаются строки не прошедшие ФЛК по уникальности кроме последней

1. При необходимости пропуска строк формируется обновленный файл и сохраняется в queue с прежним ключом, после проверки всего файла;
2. Подтверждается чтение комплексной записи из stream validation_queue, которая затем удаляется из стрима;
3. Размещается запись в list с именем validation_complete;
4. В Redis записывается статус 0 – Буферизирован;
5. В зависимости от наличия group_id в комплексной записи:
   - при отсутствии group_id – завершается обработка файла;
   - при наличии group_id – обновляется статус в groupContent_[group_id] и выполняется проверка комплектности группы в соответствии с шагами 3-6 Журнал по группе файлов.

2.2.13.2.7 Секция zookeeper

В секции zookeeper указываются параметры настроек к **Zookeeper**.

Например:

```
zookeeper:
  connect-string: ${ZOOKEEPER_DS_ADDRESS:localhost}
  connection-timeout-ms: ${ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000}
  session-timeout-ms: ${ZOOKEEPER_DS_SESSION_TIMEOUT_MS:40000}
  retry-count: 3
  retry-base-sleep-time-ms: 1_000
```

Параметры настроек

- connect-string - Подключение к Zookeeper DS, например ZOOKEEPER_DS_ADDRESS:localhost;
- connection-timeout-ms - Zookeeper DS таймаут подключения, например ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000;

– `session-timeout-ms` - Zookeeper DS таймаут сессии, например `ZOOKEEPER_DS_SESSION_TIMEOUT_MS:40000`;
– `retry-count` - количество попыток подключения, например `3`.

2.2.13.2.8 Секция persistence-mode

В секции `persistence-mode` указывается настройка хранения данных: или в Prostore или в Zookeeper. в случае выбора Prostore автоматически создаются необходимые таблицы.

Например:

```
persistence-mode: ${PERSISTENCE_MODE:prostore} # prostore | zookeeper
```

Параметры настроек

– `persistence-mode` - настройка хранения данных, например `PERSISTENCE_MODE:prostore`.

2.2.13.2.9 Секция prostore-rest-client

В секции `prostore-rest-client` реализован блок параметров конфигурирования взаимодействия с **Prostore**.

Например:

```
prostore-rest-client: # требуется синхронно менять в приложениях rest-uploader и data-uploader !!!
  persistence-datamart: ${PERSISTENCE_DATAMART:persistence}
  datasource: ${PERSISTENCE_DATASOURCE:} # по умолчанию пусто, тогда берется единственный датасорс из настроек Простора
  table-conditions: ${PERSISTENCE_TABLE_CONDITIONS:rest_uploader_conditions}
  table-ids: ${PERSISTENCE_TABLE_IDS:rest_uploader_ids}
  # Таблица активных экземляров сервиса rest-uploader (используется при persistence-mode: prostore)
  table-rest-uploader-health-check:
${PERSISTENCE_TABLE_HEALTH_CHECK:rest_uploader_health_check}
  # Таблица групп файлов (используется при persistence-mode: prostore)
  table-group-info: ${PERSISTENCE_TABLE_GROUP_INFO:group_info}
  # Таблица данных о файлах в группе (используется при persistence-mode: prostore)
  table-group-content: ${PERSISTENCE_TABLE_GROUP_CONTENT:group_content}
  # Таблица уникальных значений полей в группе (используется при persistence-mode: prostore)
  table-group-uniq: ${PERSISTENCE_TABLE_GROUP_UNIQ:group_uniq}
  # Таблица очереди файлов на ФЛК (используется при persistence-mode: prostore)
  table-validation-queue: ${PERSISTENCE_TABLE_VALIDATION_QUEUE:validation_queue}
  # Таблица очереди файлов на загрузку (используется при persistence-mode: prostore)
  table-validation-complete:
${PERSISTENCE_TABLE_VALIDATION_COMPLETE:validation_complete}
  # Таблица статусов запросов (используется при persistence-mode: prostore)
  table-requests-status: ${PERSISTENCE_TABLE_STATUS:status}
  # Таблица с файлами (используется при persistence-mode: prostore)
  table-files: ${PERSISTENCE_TABLE_FILES:files}
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9090}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}
```

Параметры настроек

– `persistence-datamart` - датамарт, где будут располагаться таблицы хранения данных, например `PERSISTENCE_DATAMART:persistence`. Используется при persistence-mode = `prostore`.
– `datasource` - по умолчанию пусто, в таком случае берется единственный датасорс

из настроек Prostore;

- `table-conditions` - условия таблиц, например `PERSISTENCE_TABLE_CONDITIONS:rest_uploader_conditions`;
- `table-ids` - настройки идентификаторов таблиц, например `PERSISTENCE_TABLE_IDS:rest_uploader_ids`;
- `table-rest-uploader-health-check` - таблица с heath-check, например `PERSISTENCE_TABLE_HEALTH_CHECK:rest_uploader_health_check`;
- `table-group-info` - таблица групп файлов, например `PERSISTENCE_TABLE_GROUP_INFO:group_info`;
- `table-group-content` - таблица данных о файлах в группе, например `PERSISTENCE_TABLE_GROUP_CONTENT:group_content`;
- `table-group-uniq` - таблица уникальных значений полей в группе, например `PERSISTENCE_TABLE_GROUP_UNIQ:group_uniq`;
- `table-validation-queue` - таблица очереди файлов на ФЛК, например `PERSISTENCE_TABLE_VALIDATION_QUEUE:validation_queue`;
- `table-validation-complete` - таблица очереди файлов на загрузку, например `PERSISTENCE_TABLE_VALIDATION_COMPLETE:validation_complete`;
- `table-requests-status` - таблица хранения статусов запросов. например `PERSISTENCE_STATUS_TABLE:status`;
- `table-files` - таблица хранения данных файлов, например `PERSISTENCE_FILES_TABLE:files`;
- `host` - адрес Prostore, например `PS_HOST:t5-prostore-01.ru-central1.internal`;
- `port` - порт Prostore, например `PS_PORT:9195`;
- `max-pool-size` - максимальное число подключений к Prostore, например `PS_MAX_POOL_SIZE:8`.

2.2.13.2.10 Секция response

В секции `response` указывается период хранения информации о статусе запроса. Например:

```
response:
  time-to-live: ${RESPONSE_TTL:10h}
```

Параметры настроек

- `time-to-live` - период хранения информации о статусе запроса, например `ESPONSE_TTL:10h`.

2.2.13.2.11 Секция control

Секция `control` определяет возможности управления дельтами от клиента. Управление дельтами от клиента допускается только при настройках delta-> `creating-delta-on-upload-request=disable` у модулей DATA-Uploader и podd-adapter-mppw.

Например:

```
control:
  delta:
    enable: ${CONTROL_DELTA_ENABLE:true}
```

Параметры настроек

- `enable` - подключение возможности управления дельтами от клиента, например `CONTROL_DELTA_ENABLE:true`.

2.2.13.2.12 Секция redis

Секция `redis` определяет настройки подключения к **Redis**.

Например:

```
redis:
  type: ${REDIS_TYPE:STANDALONE}
  connection-string: ${REDIS_CONNECTION_STRING:redis://localhost:6379}
  password: ${REDIS_PASSWORD:eYVX7EwVmmxKPCDmwMtyKVge8oLd2t81}
  max-pool-size: ${REDIS_MAX_POOL_SIZE:6}
  max-pool-waiting: ${REDIS_MAX_POOL_WAITING:24}
  max-waiting-handlers: ${REDIS_MAX_WAITING_HANDLERS:32}
  net-client-options:
    tcp-user-timeout: 30
    idle-timeout: 30
```

Параметры настроек

- `type` - тип подключения к **Redis** (STANDALONE/CLUSTER);
- `connection-string` - указывается список серверов для подключения (перечисление через запятую);
- `password` - пароль для подключения;
- `max-pool-size` - устанавливается максимальный размер пула;
- `max-pool-waiting` - устанавливается максимальный размер пула ожидающих команд;
- `max-waiting-handlers` - устанавливается максимальный размер ожидающих обработчиков;
- `net-client-options` - параметры **Redis** клиента:
- `tcp-user-timeout` - таймаут на соединение;
- `idle-timeout` - таймаут ожидания ответа от редиса.

2.2.13.2.13 Секция auth

Секция `auth` служит для хранения секрета валидации токена.

Например:

```
auth:
  secret: ${AUTH_SECRET:gPHaT%ACXGQaQ30%1id%K7@C}
  enabled: ${AUTH_ENABLED:true}
  access-list-path: rest-uploader/ids
```

Параметры настроек

- `secret` - секрет для валидации токенов, например `AUTH_SECRET:gPHaT%ACXGQaQ30%1id%K7@C`;
- `enabled` - включение/отключение аутентификации, например `AUTH_ENABLED:true`;
- `access-list-path` - путь к списку доступов, например `rest-uploader/ids`.

2.2.13.2.14 Секция metrics

Секция `metrics` предназначена для настройки параметров метрик.

Например:

```
metrics:
  port: ${METRICS_PORT:9837}
```

Параметры конфигурации

- `port` - Порт для метрик, например `METRICS_PORT:9837`.

2.2.13.2.15 Секция csv-parser

> Внимание:
>
> При загрузке файлов с форматно-логическим контролем, важно, чтобы настройки секции csv-parser были одинаковы в модулях CSV-Uploader(если используется его UI),REST-Uploader и DATA-Uploader.

Секция `csv-parser` - настройка парсинга CSV.

Например:

```
csv-parser:
  # Символ разделителя значений
  separator: ${CSV_PARSER_SEPARATOR:;}
  # Символ кавычки
  quote-char: ${CSV_PARSER_QUOTE_CHAR:"}
  # Символ экранирования значений
  escape-char: ${CSV_PARSER_ESCAPE_CHAR:'}
  # Настройка интерпретации значений как null. Допустимые значения:
  #  - EMPTY_SEPARATORS - пустое значение между двумя разделителями, например ;;
  #  - EMPTY_QUOTES - пустые кавычки, например ;"";
  #  - BOTH - оба варианта
  #  - NEITHER - никогда. Пустая строка всегда определяется как пустая строка
  field-as-null: ${CSV_PARSER_FIELD_AS_NULL:EMPTY_SEPARATORS}
```

Параметры конфигурации

- `separator` - Символ разделителя значений, например `CSV_PARSER_SEPARATOR:;`;
- `quote-char` - символ кавычки, например `CSV_PARSER_QUOTE_CHAR:"`;
- `escape-char` - Символ экранирования значений, например `CSV_PARSER_ESCAPE_CHAR:'`;

Настройка интерпретации значений как null. Допустимые значения:

  - EMPTY_SEPARATORS - пустое значение между двумя разделителями, например ;;
  - EMPTY_QUOTES - пустые кавычки, например ;"";
  - BOTH - оба варианта
  - NEITHER - никогда. Пустая строка всегда определяется как пустая строка

- `field-as-null` - способ определения null поля, например `CSV_PARSER_FIELD_AS_NULL:EMPTY_SEPARATORS`;
- `multiline-limit` - параметры парсинга мультистроковых значений.

> Внимание:
>
> Парсер может зависать если в данных встречаются вложенные неэкранированные кавычки.

1. Параметр `CSV_PARSER_ESCAPE_CHAR` работает следующим образом: если символ экранирования и символ кавычки равны `"`, то будет использован **RFC4180Parser**, который считывает все символы между двумя двойными кавычками, при этом двойная кавычка в тексте поля должна быть экранирована двойной кавычкой (Например `"поле, ""содержащее двойную кавычку"""` будет считано как `поле, "содержащее двойную кавычку"`). В противном случае будет использован **CSVParser**, использующий символ экранирования для обозначения «непечатаемых символов».
2. Параметр `CSV_PARSER_FIELD_AS_NULL` может принимать следующие значения:
   - `EMPTY_SEPARATORS` - два разделителя полей (см. csv-parser/separator) подряд считаются null. Например: строка [aaa,,ccc] содержит значения [«aaa», null, «bbb»], а строка [aaa,»»,ccc] содержит значения [«aaa», «», «bbb»].

- EMPTY_QUOTES - два «ограничителя строки» (см. csv-parser/escape-char) подряд считаются null. Например: строка [aaa,»»,ccc] содержит значения [«aaa», null, «bbb»], а строка [aaa,,ccc] содержит значения [«aaa», «», «bbb»].
- BOTH - оба варианта (см. EMPTY_SEPARATORS и EMPTY_QUOTES) считаются null. Например: обе строки [aaa,»»,ccc] и [aaa,,bbb] содержат одинаковое значение [«aaa», null, «bbb»].
- NEITHER - ни один из вариантов (см. EMPTY_SEPARATORS и EMPTY_QUOTES) не считается null. Например: обе строки [aaa,»»,ccc] и [aaa,,bbb] содержат одинаковое значение [«aaa», «», «bbb»].

2.2.13.2.16 Секция backup

Секция backup предназначена для настроек бекапирования модуля.

Например:

```
backup:
  zk-path: ${COUNTER_BACKUP_ZK_PATH:/${environment.name}/counter-provider/counters}
  commandTopic: ${BACKUP_COMMAND_TOPIC:adapter.command}
  backupTopic: ${BACKUP_TOPIC:adapter.backup}
  statusTopic: ${STATUS_TOPIC:adapter.status}
  kafka:
    consumer:
      property:
        bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
        group.id: ${COUNTER_BACKUP_GROUP_ID:counter_provider_adapter_command}
        auto.offset.reset: latest
    producer:
      property:
        bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
```

Параметры настроек

- zk-path - путь к корневой ноде zookeeper для бэкапирования, например {COUNTER_BACKUP_ZK_PATH:/${environment.name}/counter-provider/counters};
- commandTopic - топик команд бэкапирования, например: {BACKUP_COMMAND_TOPIC:adapter.command};
- backupTopic - топик для отправки забэкапированных данных, например: {BACKUP_TOPIC:adapter.backup};
- statusTopic - топик для отправки статусов бэкапирования, например: {STATUS_TOPIC:adapter.status}.

2.2.13.2.17 Секция component-info

В секции component-info указываются настройки модуля сбора информации о компонентах витрины.

Например:

```
component-info:
  enabled: true
  datasource: ''
  datamart: component_info
  table-name: component_info
  create-table-period: 60s
  publish-period: 60s
  timeout-active: 300s
  secrets:
    - redis.password
```

Параметры настроек

- `datasource` - датасорс из настроек Prostore;
- `datamart` - схема Prostore;
- `table-name` - имя таблицы для записи информации о компоненте;
- `create-table-period` - период попыток создания схемы, в случае не успешного создания;
- `publish-period` - период публикации health-check;
- `timeout-active` - период, после которого компонент считается неактивным при отсутствии health-check;
- `secrets` - список элементов конфига маскируемых при отправке, если указан узловой элемент, то маскируются все вложенные в него элементы.

2.2.13.3 Проверка форматно-логического контроля

Проверка форматно-логического контроля включает в себя:

- обязательные проверки, выполняющиеся вне зависимости от настроек модуля в синхронном режиме;
- необязательные проверки, индивидуальные для каждой таблицы, которыми управляет администратор Системы, выполняющиеся в асинхронном режиме.

Таблица 2.15 Список реализованных проверок

Наименование проверки	**Код ошибки**	**Кирилическое описание**
Проверка уникальности	`dublicate`	Дубликат файла/группы
Проверка парсинга файла	`parsingErr`	Ошибка парсинга: *текст ошибки*
Проверка кодирования	`encodingErr`	Кодировка файла не соответствует кодировке UTF-8
Проверка превышения предельного размера файла (больше 512 Мб)	`tooLargeFile`	Слишком большой файл
Проверка наличия данных в файле	`emptyFile`	Пустой файл
Проверка соответствия заголовков инфосхеме	`wrongMetadata`	Структура файла не соответствует схеме
Проверка соответствия числа столбцов в строке	`wrongFieldsCount`	Некорректное число столбцов в строке
Проверка соответствия типам полей	`wrongFieldType`	Значение не соответствует типу *требуемый тип*
Проверка уникальности полей	`nonUniq`	Значение не отвечает требованиям уникальности
Проверка регулярных выражений	`nonMatchRegex`	Значение не соответствует регулярному выражению *регулярное выражение*
Проверка соответствия условию	`nonMatchConstant`	Значение не соответствует условию *условие*
Таймаут валидации	`validationTimeout`	Истек таймаут валидации файла

2.2.13.3.1 Синхронная проверка ФЛК

Примечание:

- синхронные проверки выполняются вне зависимости от настроек модуля REST-Uploader;
- синхронные проверки являются блокирующими;
- ошибки синхронных проверок возвращаются в теле ответа по REST-API.

К синхронным проверкам относятся:

- проверка соответствия инфосхеме:
  - проверка соответствия имен и количества полей в заголовках;
  - проверка типа данных;
  - проверка экранирования данных: проверка соответствия числа столбцов по каждой строке;
- проверка соответствия файла кодировке UTF-8 , отсутствие BOM (при наличии BOM при загрузке удаляются начальные байты `ef bb bf`);
- проверка размера файла и наличия данных:
  - проверка предельного размера загружаемого файла 512Мб;
  - проверка наличия данных в файле.

2.2.13.3.2 Асинхронная проверка

Примечание:

- асинхронные проверки выполняются в зависимости от настроек модуля;
- проверки не являются блокирующими (поведение при их наличии определяется конфигурацией модуля);
- список проверок уникален для каждой таблицы и хранится в Zookeeper в виде отдельного YAML файла.

К асинхронным проверкам относятся:

- проверка уникальности полей:
  - по сочетанию атрибутов (для комплексных ключей);
  - по заданному атрибуту;
- сравнение значения с константой;
- соответствие регулярному выражению.

Для одного поля возможно создать не более одной проверки одного типа, при этом у каждого поля может быть несколько проверок разных типов.

2.2.13.3.2.1 Проверка уникальности по одному или по сочетанию полей

Проверка уникальности проводится:

- в рамках группы файлов, если заданы **headers** - для проверки в рамках группы обязательно заполнения всех полей:
  - `group_id`;
  - `group_file_num`;
  - `group_file_count`.
- при проверке в рамках группы значения ключей для проверки уникальности в рамках группы хранится в **Redis**:
  - по всем файлам в рамках группы при наличии групповых атрибутов
  - по одному файлу, при отсутствии групповых атрибутов

Пример запроса для проверки уникальности по группе файлов:

```
--проверка по сочетанию полей
fields:
  id:
    uniq: true
    uniq-with: [type,region]
--проверка уникальности по одному полю
  snils:
    match: "/^[-¥s¥d]{11}$/"
    uniq: true
```

2.2.13.3.2.2 Проверка соответствия заданному значению

- проверка соответствия заданному значению проводится для каждого файла вне зависимости от наличия group_id;
- проверка осуществляется для значений каждого поля в соответствии с заданным правилом;
- проверка соответствия заданному значению включает в себя:
-
  - проверку сравнения с константой (>, <, >=, <=, =, !=);
  - проверку соответствия регулярному выражению (должна выполняться на основе Java Util Regexp https://docs.oracle.com/javase/7/docs/api/java/util/regex/package-summary.html)

2.2.13.3.2.3 Поведение в случае таймаута валидации

Период выполнения асинхронных проверок определяется конфигурационным параметром `validation-timeout` и по умолчанию составляет 60 минут.

В случае, если за указанное в настройках время асинхронные проверки не были выполнены, файл удаляется из очереди с обогащением отчета о найденных ошибках ошибкой `validationTimeout`.

В случае возникновения подобной ошибки рекомендуется:

- проверить регулярные выражения, по которым происходит проверка, так как неверно заданное регулярное выражение кратно увеличивает скорость проверки;
- увеличить значение `validation-timeout` и повторить загрузку данных.

2.2.13.4 Статусная модель

Статус запроса к модулю REST-Uploder можно получить, выполнив запрос с передачей идентификатора запроса `GET '/v2/requests/:requestId/status'`

В ответ возвращается три поля:

- code - код статуса;
- errorMessage - сообщение об ошибке, заполняется лишь в случае ошибочного статуса;
- description - описание ошибки, заполняется лишь в случае ошибочного статуса.

Пример ответа на запрос статуса

```
{"code":2,"description":null,"errorMessage":null}
{"code":3,"description":"Успешно обработан","errorMessage":null}
```

```
{"code":4,"description":"Ошибка обработки
запроса","errorMessage":"ru.itone.dtm.data.uploader.upload.UploadException:
ru.itone.dtm.prostore.rest.api.ProstoreRestException: Error executing query [insert
into univer.slots select
resource_id,slot_id,tag_type,tag_age,available_date,duration,slot_create_ts,slot_update
_ts,slot_status,sys_op from univer.slots_upload_ext]: All of the connectors are
failed¥n¥tat"}
{"code":5,"description":"Идентификатор запроса не обнаружен","errorMessage":null}
{"code":7,"description":"Ошибки ФЛК","errorMessage":null}
```


В свою очередь, идентификатор запроса ранее был получен в ответ от REST-Uploader:

- POST“/v2/datamarts/{datamart_name}/tables/{table_name}/upload;
- POST“/v2/datamarts/{datamart_name}/tables/{table_name}/delete.

Таблица 2.16 Статусная модель

Код статуса	**Описание статуса**	**Действия при получении данного статуса**
-1	Загрузка данных в буффер	Данный статус клиентскому приложению не возвращается, ответ вернется после того как загружаемые данные будут сохранены приложением REST-Uploader для дальнейшей загрузки.
0	Запрос буфферизирован	Выполнить повторный запрос статуса с некоторой задержкой. Рекомендуемая задержка 30сек.
1	Ожидает открытия дельты	Выполнить повторный запрос статуса с некоторой задержкой. Рекомендуемая задержка 30сек.
2	В обработке (модулем DATA-Uploader)	Выполнить повторный запрос статуса с некоторой задержкой. Рекомендуемая задержка 30сек.
3	Успешно обработан	Дополнительных действий не требуется
4	Ошибка обработки запроса	Необходимо: – изучить содержимое вернувшегося поля description – если суть проблемы не ясна из предыдущего пункта, изучить содержимое логов приложений на предмет наличия ошибок: ○ REST-Uploader ○ DATA-Uploader ○ podd-adapter-mppw ○ query-execution-core ○ используемого коннектора (kafka-postgres-writer или kafka-jet-writer)
5	Идентификатор запроса не обнаружен	Использовать действующий идентификатор запроса
6	Форматно-логический контроль	Выполнить повторный запрос статуса с некоторой задержкой. Рекомендуемая задержка 30сек.
7	Ошибки ФЛК	В процессе ФЛК выявлены ошибки, необходимо запросить отчет ФЛК, обратившись к REST-Uploader с запросом `GET '/v2/requests/{request_id}/report/'` Далее проанализировать и устранить выявленные недочеты в загружаемых данных или скорректировать проверки ФЛК.

2.2.13.5 Спецификация модуля асинхронной загрузки данных из сторонних источников

Данная спецификация описывает возможность загрузки данных в витрину, получение статуса запроса, удаление данных из витрины.

Метод	**URL**	**Назначение**
POST	`v2/datamarts/{datamart_name}/tables/{table_name}/upload`	Загрузка данных в витрину с учетом реализации ФЛК
GET	`v2/requests/{request_id}/status`	Получение статуса запроса

Метод	URL	Назначение
DELETE	v2/datamarts/{datamart_name}/tables/{table_name}/delete	Удаление данных из витрины
POST	v2/conditions/{datamart}/{table}	запрос для загрузки списка правил для таблицы, для сохранения в **Zookeeper**
PUT	v2/conditions/{datamart}/{table}	запрос для добавления правил для таблицы, для сохранения в **Zookeeper**
GET	v2/conditions/{datamart}/{table}	запрос для получения списка проверок для таблицы, хранящийся в **Zookeer**
DELETE	v2/conditions/{datamart}/{table}	запрос для удаления всего списка проверок по таблице
GET	v2/requests/{request_id}/report	Возвращает отчет по форматно логическом контроле загружаемых данных в формате .csv
GET	v2/group/{group_id}/report	Запрос возвращает отчет по комплектности группы загружаемых файлов в формате .csv

```yaml
openapi: 3.0.1
info:
  title: Rest-uploader
  description: This is a rest-uploader service for datamart filling
  version: 2.0.0
servers:
  - url: 'http://localhost:8081'
paths:
  '/v2/datamarts/{datamart_name}/tables/{table_name}/upload':
    post:
      summary: Add a new data to the datamart
      operationId: post-v2-datamarts-datamart_name-tables-table_name-upload
      parameters:
        - name: datamart_name
          in: path
          required: true
          schema:
            type: string
        - name: table_name
          in: path
          required: true
          schema:
            type: string
        - name: groupId
          in: header
          description: 'идентификатор группы'
          required: false
          schema:
            type: string
        - name: groupFileNum
          in: header
          description: 'номер файла в группе'
          required: false
          schema:
            type: integer
        - name: groupFileCount
```

```
        in: header
        description: 'число файлов в группе'
        required: false
        schema:
          type: integer
      requestBody:
        $ref: '#/components/requestBodies/uploadData'
      responses:
        '200':
          description: successful operation
          headers:
            requestId:
              schema:
                type: string
          content:
            text/plain:
              schema:
                type: string
        '400':
          description: Bad request
          headers:
            requestId:
              schema:
                type: string
          content:
            text/csv:
              schema:
                type: array
                items:
                  type: array
                  items:
                    type: string
        '401':
          description: Invalid token
          content: {}
        '403':
          description: Invalid idetifier
          content: {}
        '500':
          description: Internal server error
          content: {}
      security:
        - bearerAuth: []
      tags:
        - data
      description: Загрузка данных из внешних источников. К телу прикладывается файл
csv
    parameters:
      - schema:
          type: string
        name: datamart_name
        in: path
        required: true
      - schema:
          type: string
        name: table_name
        in: path
        required: true
  '/v2/requests/{request_id}/status':
    get:
      summary: Return request status
      description: Возвращение статуса запроса
```

```
    operationId: get-v2-request-request_id-status
    parameters:
      - name: request_id
        in: path
        description: Identifier of request
        required: true
        schema:
          type: string
    responses:
      '200':
        description: successful operation. При ошибочном ответе возвращается код и описание ошибки в соответствии с примером
        content:
          plain/text:
            schema:
              type: string
              example: "code: 4, description: Ошибка обработки запроса, errorMessage: Полный текст ошибки..."
      '401':
        description: Invalid token
        content: {}
      '403':
        description: Invalid idetifier
        content: {}
      '500':
        description: Internal server error
        content: {}
    security:
      - bearerAuth: []
    tags:
      - data
  parameters:
    - schema:
        type: string
      name: request_id
      in: path
      required: true
'/v2/requests/{request_id}/reportFLK':
  get:
    summary: Return report FLK check
    description: Возвращает отчет по формато логическом контроле загружаемых данных
    operationId: get-v2-request-request_id-reportFLK
    parameters:
      - name: request_id
        in: path
        description: Identifier of request
        required: true
        schema:
          type: string
    responses:
      '200':
        description: successful operation
        content:
          text/csv:
            schema:
              type: array
              items:
                type: array
                items:
                  type: string
      '401':
        description: Invalid token
```

```
        content: {}
      '403':
        description: Invalid idetifier
        content: {}
      '500':
        description: Internal server error
        content: {}
    security:
      - bearerAuth: []
    tags:
      - report
'/v2/group/{groupId}/report':
  get:
    summary: Return report group
    description: Возвращает отчет по комплектности группы загружаемых файлов
    operationId: get-v2-group-groupId-report
    parameters:
      - name: groupId
        in: path
        description: Identifier of group
        required: true
        schema:
          type: string
    responses:
      '200':
        description: successful operation
        content:
          text/csv:
            schema:
              type: array
              items:
                type: array
                items:
                  type: string
      '401':
        description: Invalid token
        content: {}
      '403':
        description: Invalid idetifier
        content: {}
      '404':
        description: Not Found
        content:
          json:
            schema:
              type: string
              description: Сообщение об ошибке
              example: 'Report not found'
      '500':
        description: Internal server error
        content: {}
    security:
      - bearerAuth: []
    tags:
      - report

'/v2/{datamart_name}/tables/{table_name}/delete':
  parameters:
    - schema:
        type: string
      name: datamart_name
      in: path
```

```
        required: true
      - schema:
          type: string
        name: table_name
        in: path
        required: true
      - name: groupId
        in: header
        description: 'идентификатор группы'
        required: false
        schema:
          type: string
      - name: groupFileNum
        in: header
        description: 'номер файла в группе'
        required: false
        schema:
          type: integer
      - name: groupFileCount
        in: header
        description: 'число файлов в группе'
        required: false
        schema:
          type: integer
    post:
      summary: Delete data by primary key array
      operationId: post-v2-datamart_name-tables-table_name-delete
      responses:
        '200':
          description: OK
        '400':
          description: Bad Request
        '401':
          description: Invalid token
        '403':
          description: Invalid identifier
        '500':
          description: Internal server error
          content: {}
      tags:
        - data
      description: Удаление данных по массиву первичных ключей. К телу прикладывается
файл csv
      security:
        - bearerAuth: []
      requestBody:
        $ref: '#/components/requestBodies/deleteData'
  '/v2/conditions/{datamart}/{table}':
    post:
      summary: Create verification conditions
      description: Формирование правил проверки для датамарта/таблицы
      operationId: post-v2-conditions
      parameters:
        - name: datamart
          in: path
          description: Name of datamart
          required: true
          schema:
            type: string
        - name: table
          in: path
          description: Name of table
```

```
        required: true
        schema:
          type: string
    requestBody:
      content:
        form-data/yaml: {}
        form-data/json: {}
        application/json:
          schema: {}
          example: '
          fields:
            birthday:
            # ограничение на формат даты ГГГГ-ММ-ДД: 4 цифры года, 2 цифры месяца
и 2 цифры дня
            # Уникальность по одному полю
              match: "(¥¥d{4})¥¥-(¥¥d{2})¥¥-(¥¥d{2})"
              uniq: true
            # Уникальность по сочетанию полей
              uniq-with: [code,passport]
            code:
              in: ["1","2","3","4","5","6","7","8","9","10"]
              uniq: true'
    responses:
      '200':
        description: successful operation
      '401':
        description: Invalid token
        content: {}
      '403':
        description: Invalid idetifier
        content: {}
      '500':
        description: Internal server error
        content: {}
    security:
      - bearerAuth: []
    tags:
      - conditions
  put:
    summary: Update verification conditions
    description: Обновление правил проверки для датамарта/таблицы
    operationId: put-v2-conditions
    parameters:
      - name: datamart
        in: path
        description: Name of datamart
        required: true
        schema:
          type: string
      - name: table
        in: path
        description: Name of table
        required: true
        schema:
          type: string
    requestBody:
      content:
        form-data/yaml: {}
        form-data/json: {}
        application/json:
          schema: {}
          example: '
```

```
          fields:
            birthday:
            # ограничение на формат даты ГГГГ-ММ-ДД: 4 цифры года, 2 цифры месяца
и 2 цифры дня
            # Уникальность по одному полю
              match: "(¥¥d{4})¥¥-(¥¥d{2})¥¥-(¥¥d{2})"
              uniq: true
            # Уникальность по сочетанию полей
              uniq-with: [code,passport]
            code:
              in: ["1","2","3","4","5","6","7","8","9","10"]
              uniq: true'
    responses:
      '200':
        description: successful operation
      '401':
        description: Invalid token
        content: {}
      '403':
        description: Invalid idetifier
        content: {}
      '500':
        description: Internal server error
        content: {}
    security:
      - bearerAuth: []
    tags:
      - conditions
  get:
    summary: Return verification conditions
    description: Возвращение правил проверки для датамарта/таблицы
    operationId: get-v2-conditions
    parameters:
      - name: datamart
        in: path
        description: Name of datamart
        required: true
        schema:
          type: string
      - name: table
        in: path
        description: Name of table
        required: true
        schema:
          type: string
    responses:
      '200':
        description: successful operation
        content:
          form-data/yaml: {}
          form-data/json: {}
      '401':
        description: Invalid token
        content: {}
      '403':
        description: Invalid idetifier
        content: {}
      '500':
        description: Internal server error
        content: {}
    security:
      - bearerAuth: []
```

```
    tags:
      - conditions
  delete:
    summary: Delete verification conditions
    description: Удаление правил проверки для датамарта/таблицы
    operationId: delete-v2-conditions
    parameters:
      - name: datamart
        in: path
        description: Name of datamart
        required: true
        schema:
          type: string
      - name: table
        in: path
        description: Name of table
        required: true
        schema:
          type: string
    responses:
      '200':
        description: successful operation
      '401':
        description: Invalid token
        content: {}
      '403':
        description: Invalid idetifier
        content: {}
      '500':
        description: Internal server error
        content: {}
    security:
      - bearerAuth: []
    tags:
      - conditions
components:
  requestBodies:
    uploadData:
      description: 'загружаемые данные'
      required: true
      content:
        text/csv:
          schema:
            type: array
            items:
              type: array
              items:
                type: string
        multipart/form-data:
          schema:
            required:
              - uploadData
            properties:
              uploadData:
                type: string
                description: Data for uploading
                format: binary
    deleteData:
      description: 'Удаляемые данные, важны лишь ключевые поля, остальные могут
отсутствовать или будут проигнорированы'
      required: true
      content:
```

```
        application/json:
          schema:
            type: object
            properties:
              primaryKeys:
                type: array
                description: Массив первичных ключей
                items:
                  type: array
                  items:
                    type: string
        text/csv:
          schema:
            type: array
            items:
              type: array
              items:
                type: string
        multipart/form-data:
          schema:
            required:
              - uploadData
            properties:
              uploadData:
                type: string
                description: Data for uploading
                format: binary
  securitySchemes:
    bearerAuth:
      type: http
      scheme: bearer
      bearerFormat: JWT
  examples: {}

security:
  - bearerAuth: []

tags:
  - name: data
    description: Загрузка и удаление данных из Витрины, получение статуса запроса
  - name: conditions
    description: Управление правилами проверок для таблицы
  - name: report
    description: Получение отчетов
```

2.2.14 Настройка СМЭВ4-адаптера – Модуль подписок

2.2.14.1 Конфигурация модуля СМЭВ4-адаптер - Модуль подписок (application.yml)

Файл `application.yml` – основной конфигурационный файл модуля, в котором задана его логика и порядок работы модуля:

- настройка подключения к **Prostore** (секция: `prostore`);
- подключение к **Брокеру сообщений Kafka**, **Zookeeper**;
- порядок обработки запросов между Получателем и Поставщиком данных (секция: `kafka`);
- настройка метрик (секция: `metrics`) и другие настройки необходимые для корректной работы адаптера.

2.2.14.1.1 Пример файла application.yml

В конфигурационном файле следует задавать только те настройки, которые необходимы для решения текущих бизнес-задач.

```yaml
environment:
  name: ${ENVIRONMENT_NAME:test}

http-server:
  port: ${HTTP_PORT:8085}

executor:
  reader-pool-size: ${EXECUTOR_READER_POOL_SIZE:20}

zookeeper:
  connection-string: ${ZOOKEEPER_DS_ADDRESS:localhost}
  connection-timeout-ms: ${ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000}
  session-timeout-ms: ${ZOOKEEPER_DS_SESSION_TIMEOUT_MS:86400000}
  chroot: ${ZOOKEEPER_DS_CHROOT:/adapter}

migration:
  enabled: ${MIGRATION_ENABLE:false}
  old-connection-string: ${OLD_ZOOKEEPER_DS_ADDRESS:localhost}

table-metadata:
  cache:
    enabled: false

prostore-rest-client:
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9195}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}

prostore:
  status-event-topic:
    topic: ${PS_STATUS_EVENT_TOPIC:status.event}
    property:
      bootstrap.servers: ${kafka.internal.bootstrap.servers}
      group.id: ${kafka.external.topic.prefix}replicator-status-event
      auto.offset.reset: earliest
      enable.auto.commit: false

subscription:
  consumer:
    # режим отмены подписки на потребителе. Возможные значения rename, drop, none
    cancel-mode: none
    # Продолжительность исключения подписки в случае ошибки применения
    exclusion-duration: 24h
    # Периодичность срабатывания механизма очистки для ошибочных подписок
    cleanup-interval: 5s

replication:
  max-deltas-in-batch: 10 # максимальное число дельт в выгружаемой пачке

# Массив описания standalone таблиц, участвующих в репликации
#standalone-tables: []
# Пример описания
#standalone-tables:
#  - table: "misdm05.readable_book"
#    anchor: "update_at"
#    soft-delete: "delete_at"
```

```
storage: # блок настроек хранения чанков данных репликации. При изменении параметров необходимо синхронизировать аналогичные параметры в сервисе mppw
  type: ${STORAGE_TYPE:postgres} # тип, postgres|kafka
  postgres: # параметры подключения к базе. Используется только при type=postgres
    connection:
      database: ${STORAGE_DATABASE:replication}
      schema: ${STORAGE_SCHEMA:public}
      host: ${STORAGE_HOST:localhost}
      port: ${STORAGE_PORT:5432}
      user: ${STORAGE_USER:postgres}
      password: ${STORAGE_PASSWORD:postgres}
    pool:
      max-size: ${STORAGE_POOL_SIZE:30}


kafka:
  agent.topic.prefix: ${AGENT_TOPIC_PREFIX:}
  max-concurrent-handle: ${KAFKA_MAX_CONCURRENT_HANDLE:10}
  commit-interval: ${KAFKA_COMMIT_INTERVAL:5s}
  external:
    bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
    topic.prefix: ${EXTERNAL_TOPIC_PREFIX:${kafka.agent.topic.prefix}}
  internal:
    bootstrap.servers: ${PS_KAFKA:localhost:9092}
    topic.prefix: ${INTERNAL_TOPIC_PREFIX:${kafka.agent.topic.prefix}}
  consumer:
    subscription-request:
      topic: ${kafka.external.topic.prefix}replication.rq
      max-concurrent-handle: ${kafka.max-concurrent-handle}
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.external.bootstrap.servers}
        group.id: ${kafka.external.topic.prefix}replicator-subscription-request
        auto.offset.reset: earliest
        enable.auto.commit: false
    subscription-cancel-request:
      topic: ${kafka.external.topic.prefix}replication.cancel.rq
      max-concurrent-handle: ${kafka.max-concurrent-handle}
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.external.bootstrap.servers}
        group.id: ${kafka.external.topic.prefix}replicator-subscription-cancel-request
        auto.offset.reset: earliest
        enable.auto.commit: false
    subscription-consumer-cancel-request:
      topic: ${kafka.external.topic.prefix}replication.cancel.in.rq
      max-concurrent-handle: ${kafka.max-concurrent-handle}
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.external.bootstrap.servers}
        group.id: ${kafka.external.topic.prefix}}replicator-subscription-consumer-
cancel-request
        auto.offset.reset: earliest
        enable.auto.commit: false
    subscription-storage-request:
      topic: ${kafka.external.topic.prefix}replication.in.rq
      max-concurrent-handle: ${kafka.max-concurrent-handle}
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.external.bootstrap.servers}
        group.id: ${kafka.external.topic.prefix}replicator-subscription-storage-request
```

```
      auto.offset.reset: earliest
      enable.auto.commit: false
    delta-request:
      topic: ${kafka.external.topic.prefix}delta.rq
      max-concurrent-handle: ${kafka.max-concurrent-handle}
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.external.bootstrap.servers}
        group.id: ${kafka.external.topic.prefix}replicator-delta-request
        auto.offset.reset: earliest
        enable.auto.commit: false
    delta-apply-notification:
      topic: ${kafka.internal.topic.prefix}subscription.in
      max-concurrent-handle: ${kafka.max-concurrent-handle}
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
        group.id: ${kafka.internal.topic.prefix}replicator-delta-apply-notification
        auto.offset.reset: earliest
        enable.auto.commit: false
    mppw-delta-apply-result:
      topic: ${kafka.internal.topic.prefix}mppw.delta.in.rs
      max-concurrent-handle: ${kafka.max-concurrent-handle}
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
        group.id: ${kafka.internal.topic.prefix}replicator-delta-apply-result
        auto.offset.reset: earliest
        enable.auto.commit: false

  producer:
    subscription-result: ${kafka.external.topic.prefix}replication.rs
    subscription-error: ${kafka.external.topic.prefix}replication.err
    subscription-cancel-error: ${kafka.external.topic.prefix}replication.cancel.rs
    subscription-cancel-result: ${kafka.external.topic.prefix}replication.cancel.rs
    subscription-consumer-cancel-result:
${kafka.external.topic.prefix}replication.cancel.in.rs
    subscription-storage-result: ${kafka.external.topic.prefix}replication.in.rs
    subscription-storage-error: ${kafka.external.topic.prefix}replication.in.err
    delta-error: ${kafka.external.topic.prefix}delta.err
    delta-apply-error: ${kafka.external.topic.prefix}delta.in.err
    delta-apply-result: ${kafka.external.topic.prefix}delta.in.rs
    delta-notification: ${kafka.external.topic.prefix}delta.notification
    property:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
    internal:
      mppr-delta-request: ${kafka.internal.topic.prefix}mppr.delta.rq
      mppw-delta-apply-request: ${kafka.internal.topic.prefix}mppw.delta.in.rq
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}

metrics:
  port: ${METRICS_PORT:9837}

log:
  replRequest: ${REPL_REQUEST_LOG_ENABLED:false}
  replResponse: ${REPL_RESPONSE_LOG_ENABLED:false}

backup:
  zk-path: ${REPLICATOR_BACKUP_ZK_PATH:/${environment.name}/podd-adapter-replicator}
  commandTopic: ${BACKUP_COMMAND_TOPIC:adapter.command}
  adapterCommandBroadcast:
```

```
${REPLICATOR_COMMAND_BROADCAST_TOPIC:adapter.command.broadcast}
  backupTopic: ${BACKUP_TOPIC:adapter.backup}
  statusTopic: ${STATUS_TOPIC:adapter.status}
  timeout: ${BACKUP_TIMEOUT:PT180s}
  idleDelay: ${BACKUP_DELAY:500}
  kafka:
    consumer:
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
        group.id: ${REPLICATOR_BACKUP_GROUP_ID:podd_adapter_replicator_adapter_command}
        auto.offset.reset: latest
    producer:
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
```

2.2.14.2 Параметры конфигурации

Настройка конфигурации **СМЭВ4-адаптера - Модуль подписок** осуществляется путем редактирования параметров настроек в файле `application.yml`, где настраиваются секции:

- `environment` - указывается название окружения (`test`, `prod` и т.д.);
- `http-server` - настройки порта подключения;
- `executor` - масштабирования нагрузки на модуль;
- `zookeeper` – параметры подключения к **Zookeeper**;
- `migration` - настройки миграции;
- `prostore-api-client` - блок параметров конфигурирования взаимодействия с **Prostore**;
- `subscription` - настройки подписки;
- `replication` - настройки репликации;
- `storage` - блок настроек хранения чанков данных репликации. При изменении параметров необходимо синхронизировать аналогичные параметры в сервисе MPPW.
- `kafka` - настройки параметров подключения к шине данных **Apache Kafka**;
- `log` - настройка сохранения лог-файла;
- `metrics` - настройка получения метрик;
- `backup` - настройки бекапирования.

2.2.14.2.1 Секция environment

В секции `environment` указывается среда разработки (dev, test, stable, prod).

Например:

```
environment:
  name: ${ENVIRONMENT_NAME:test}
```

Параметры настроек

- `name` - среда разработки, например `ENVIRONMENT_NAME:test`.

2.2.14.2.2 Секция http-server

Секция `http-server` предназначена для настройки порта и протокола передачи данных (одно из значений `http` или `https`).

Например:

```
http-server:
  port: ${HTTP_PORT:8085}
```

Параметры настроек

– `port` - порт веб-сервера, например: `HTTP_PORT:8085`.

2.2.14.2.3 Секция executor

Секция `executor` предназначена для масштабирования нагрузки (увеличения / уменьшения) на модуль.

Например:

```
executor:
  reader-pool-size: ${EXECUTOR_READER_POOL_SIZE:20}
```

Параметры настроек

– `reader-pool-size` - размер пула для чтения **Kafka**, например `EXECUTOR_READER_POOL_SIZE:20`.

2.2.14.2.4 Секция zookeeper

Секция `zookeeper` предназначена для настройки параметров подключения к **Zookeeper**.

Например:

```
zookeeper:
  connection-string: ${ZOOKEEPER_DS_ADDRESS:t5-adsp-01.ru-central1.internal}
  connection-timeout-ms: ${ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000}
  session-timeout-ms: ${ZOOKEEPER_DS_SESSION_TIMEOUT_MS:86400000}
  chroot: ${ZOOKEEPER_DS_CHROOT:/adapter}
```

Параметры настроек

– `ZOOKEEPER_DS_ADDRESS` - адрес сервера Zookeeper DS;

– `ZOOKEEPER_DS_SESSION_TIMEOUT_MS` - таймаут подключения к Zookeeper DS,максимальное время ожидания для выявления сбоев потребителей, указывается в миллисекундах (MS.);

– `ZOOKEEPER_DS_SESSION_TIMEOUT_MS` - таймаут сессии, максимальное время ожидания подключения к Zookeeper. Если ответ не получен до истечения установленного значения, клиент повторно отправляет запрос при необходимости, указывается в миллисекундах (MS.);

– `ZOOKEEPER_DS_CHROOT` - Zookeeper DS chroot path.

2.2.14.2.5 Секция migration

Секция `migration` реализована настройка миграции **Zookeeper** для задачи бекапирования.

Например:

```
migration:
  enabled: ${MIGRATION_ENABLE:false}
  old-connection-string: ${OLD_ZOOKEEPER_DS_ADDRESS:localhost}
```

Параметры настроек

– `enabled` - подключение миграции, например `{MIGRATION_ENABLE:false}`;

– `old-connection-string` - адрес **Zookeeper**, например `{OLD_ZOOKEEPER_DS_ADDRESS:localhost}`.

2.2.14.2.6 Секция prostore-rest-client

В секции `prostore-rest-client` настраиваются параметры конфигурирования взаимодействия с **Prostore**.

Например:

```
prostore-rest-client:
  host: ${PS_HOST:t5-prostore-01.ru-central1.internal}
  port: ${PS_PORT:9195}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}
```

Параметры настроек

- `host` - адрес **Prostore**, например `PS_HOST:t5-prostore-01.ru-central1.internal`;
- `port` - порт **Prostore**, например `PS_PORT:9195`;
- `max-pool-size` - максимальное число подключений к **Prostore**, например `PS_MAX_POOL_SIZE:8`.

2.2.14.2.7 Секция subscription

В секции `subscription` настраиваются подписки на потребителе.

Например

```
subscription:
  consumer:
    # режим отмены подписки на потребителе. Возможные значения rename, drop, none
    cancel-mode: none
    # Продолжительность исключения подписки в случае ошибки применения
    exclusion-duration: 24h
    # Периодичность срабатывания механизма очистки для ошибочных подписок
    cleanup-interval: 5s
```

Параметры конфигурации

- `cancel-mode` - режим отмены подписки на потребителе. Возможные значения: `rename`, `drop`, `none`;
- `exclusion-duration` - продолжительность исключения подписки в случае ошибки применения (в часах), например `exclusion-duration: 24h`;
- `cleanup-interval` - периодичность срабатывания механизма очистки для ошибочных подписок (в секундах), например `cleanup-interval: 5s`.

2.2.14.2.8 Секция replication

В секции `subscription` указываются настройки репликации.

Например

```
replication:
  max-deltas-in-batch: 10
```

Параметры конфигурации

- `max-deltas-in-batch` - максимальное число дельт в выгружаемой пачке, например `max-deltas-in-batch: 10`.

2.2.14.2.9 Секция storage

В секции `storage` указываются настройки хранения чанков данных репликации.

ри изменении параметров необходимо синхронизировать аналогичные параметры в сервисе MPPW.

Например:

```
storage:
  type: ${STORAGE_TYPE:postgres}
  postgres:
    connection:
      database: ${STORAGE_DATABASE:replication}
      schema: ${STORAGE_SCHEMA:public}
      host: ${STORAGE_HOST:localhost}
      port: ${STORAGE_PORT:5432}
      user: ${STORAGE_USER:postgres}
      password: ${STORAGE_PASSWORD:postgres}
    pool:
      max-size: ${STORAGE_POOL_SIZE:30}
```

Параметры настроек

- `type` - тип, postgres|kafka, например: `STORAGE_TYPE:postgres`;
- `connection` - параметры подключения к базе.

2.2.14.2.10 Секция kafka

В секции `kafka` настраиваются параметры подключения к шине данных Apache Kafka (используется для взаимодействия с СМЭВ4-адаптером) и настройки взаимодействия через топики модуля **СМЭВ4-адаптер - Модуль исполнения запросов**.

Модуль взаимодействует через топики:

- `replication.rq/rs/err` - запрос создания подписки (Поставщик данных);
- `replication.cancel.rq/rs/err` - запрос отмены подписки (Поставщик данных);
- `delta.rq/mppr.delta.rq` - запрос дельты (Поставщик данных);
- `replication.in.rq/rs/err` - запрос создания структуры по подписке (Получатель данных);
- `subscription.in/delta.in.rs/delta.in.err` - запрос применения дельты (Получатель данных);
- `status.event/delta.notification` - статусы с Prostore (Поставщик данных).

Например:

```
kafka:
  agent.topic.prefix: ${AGENT_TOPIC_PREFIX:}
  max-concurrent-handle: ${KAFKA_MAX_CONCURRENT_HANDLE:10}
  commit-interval: ${KAFKA_COMMIT_INTERVAL:5s}
  external:
    bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
    topic.prefix: ${EXTERNAL_TOPIC_PREFIX:${kafka.agent.topic.prefix}}
  internal:
    bootstrap.servers: ${PS_KAFKA:localhost:9092}
    topic.prefix: ${INTERNAL_TOPIC_PREFIX:${kafka.agent.topic.prefix}}
  consumer:
    subscription-request:
      topic: ${kafka.external.topic.prefix}replication.rq
      max-concurrent-handle: ${kafka.max-concurrent-handle}
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.external.bootstrap.servers}
        group.id: ${kafka.external.topic.prefix}replicator-subscription-request
        auto.offset.reset: earliest
        enable.auto.commit: false
    subscription-cancel-request:
      topic: ${kafka.external.topic.prefix}replication.cancel.rq
      max-concurrent-handle: ${kafka.max-concurrent-handle}
      commit-interval: ${kafka.commit-interval}
      property:
```

```
      bootstrap.servers: ${kafka.external.bootstrap.servers}
      group.id: ${kafka.external.topic.prefix}replicator-subscription-cancel-request
      auto.offset.reset: earliest
      enable.auto.commit: false
    subscription-consumer-cancel-request:
      topic: ${kafka.external.topic.prefix}replication.cancel.in.rq
      max-concurrent-handle: ${kafka.max-concurrent-handle}
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.external.bootstrap.servers}
        group.id: ${kafka.external.topic.prefix}}replicator-subscription-consumer-
cancel-request
        auto.offset.reset: earliest
        enable.auto.commit: false
    subscription-storage-request:
      topic: ${kafka.external.topic.prefix}replication.in.rq
      max-concurrent-handle: ${kafka.max-concurrent-handle}
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.external.bootstrap.servers}
        group.id: ${kafka.external.topic.prefix}replicator-subscription-storage-request
        auto.offset.reset: earliest
        enable.auto.commit: false
    delta-request:
      topic: ${kafka.external.topic.prefix}delta.rq
      max-concurrent-handle: ${kafka.max-concurrent-handle}
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.external.bootstrap.servers}
        group.id: ${kafka.external.topic.prefix}replicator-delta-request
        auto.offset.reset: earliest
        enable.auto.commit: false
    delta-apply-notification:
      topic: ${kafka.internal.topic.prefix}subscription.in
      max-concurrent-handle: ${kafka.max-concurrent-handle}
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
        group.id: ${kafka.internal.topic.prefix}replicator-delta-apply-notification
        auto.offset.reset: earliest
        enable.auto.commit: false
    mppw-delta-apply-result:
      topic: ${kafka.internal.topic.prefix}mppw.delta.in.rs
      max-concurrent-handle: ${kafka.max-concurrent-handle}
      commit-interval: ${kafka.commit-interval}
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
        group.id: ${kafka.internal.topic.prefix}replicator-delta-apply-result
        auto.offset.reset: earliest
        enable.auto.commit: false

  producer:
    subscription-result: ${kafka.external.topic.prefix}replication.rs
    subscription-error: ${kafka.external.topic.prefix}replication.err
    subscription-cancel-error: ${kafka.external.topic.prefix}replication.cancel.rs
    subscription-cancel-result: ${kafka.external.topic.prefix}replication.cancel.rs
    subscription-consumer-cancel-result:
${kafka.external.topic.prefix}replication.cancel.in.rs
    subscription-storage-result: ${kafka.external.topic.prefix}replication.in.rs
    subscription-storage-error: ${kafka.external.topic.prefix}replication.in.err
    delta-error: ${kafka.external.topic.prefix}delta.err
    delta-apply-error: ${kafka.external.topic.prefix}delta.in.err
```

```
    delta-apply-result: ${kafka.external.topic.prefix}delta.in.rs
    delta-notification: ${kafka.external.topic.prefix}delta.notification
    property:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
    internal:
      mppr-delta-request: ${kafka.internal.topic.prefix}mppr.delta.rq
      mppw-delta-apply-request: ${kafka.internal.topic.prefix}mppw.delta.in.rq
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
```

Параметры конфигурации

- `AGENT_TOPIC_PREFIX` - значение префикса для топиков. Топики взаимодействия с **СМЭВ4-адаптером - Модуль исполнения запросов**
- (см. Спецификация Модуля исполнения запросов).

2.2.14.2.11 Секция metrics

В секции `metrics` настраиваются параметры метрик.

Например:

```
metrics:
  port: ${METRICS_PORT:9837
```

Параметры конфигурации

- `port` - порт для получения метрик, например `9837`.

2.2.14.2.12 Секция log

В секции `log` настраиваются параметры логирования.

Например:

```
log:
  replRequest: ${REPL_REQUEST_LOG_ENABLED:false}
  replResponse: ${REPL_RESPONSE_LOG_ENABLED:false}
```

Параметры конфигурации

- `repl-request` - журналировать запросы к модулю подписок, например `REPL_REQUEST_LOG_ENABLED:false`;
- `repl-response` - журналировать ответы модуля подписок, например `REPL_RESPONSE_LOG_ENABLED:false`.

2.2.14.2.13 Секция backup

В секции `backup` настраиваются параметры бекапирования модуля.

Например:

```
backup:
  zk-path: ${REPLICATOR_BACKUP_ZK_PATH:/${environment.name}/podd-adapter-replicator}
  commandTopic: ${BACKUP_COMMAND_TOPIC:adapter.command}
  adapterCommandBroadcast:
${REPLICATOR_COMMAND_BROADCAST_TOPIC:adapter.command.broadcast}
  backupTopic: ${BACKUP_TOPIC:adapter.backup}
  statusTopic: ${STATUS_TOPIC:adapter.status}
  timeout: ${BACKUP_TIMEOUT:PT180s}
  idleDelay: ${BACKUP_DELAY:500}
  kafka:
    consumer:
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
        group.id: ${REPLICATOR_BACKUP_GROUP_ID:podd_adapter_replicator_adapter_command}
        auto.offset.reset: latest
    producer:
      property:
        bootstrap.servers: ${kafka.internal.bootstrap.servers}
```

Параметры настроек

- `zk-path` - путь к корневой ноде **Zookeeper** для бэкапирования, например `{COUNTER_BACKUP_ZK_PATH:/${environment.name}/counter-provider/counters}`;
- `commandTopic` - топик команд бэкапирования, например: `{BACKUP_COMMAND_TOPIC:adapter.command}`;
- `backupTopic` - топик для отправки сохраненных данных, например: `{BACKUP_TOPIC:adapter.backup}`;
- `statusTopic` - топик для отправки статусов бэкапирования, например: `{STATUS_TOPIC:adapter.status}`.

2.2.15 Настройка BLOB-адаптера

2.2.15.1 Конфигурация BLOB-адаптера (application.yml)

Файл `application.yml` – основной конфигурационный файл **BLOB-адаптер**, в котором задана логика и порядок работы модуля:

- получение и обработка входящих запросов;
- настройка подключения к СМЭВ3-адаптеру, СМЭВ4-адаптеру и Хранилищу BLOB-объектов, и другие настройки необходимые для корректной работы модуля.

2.2.15.1.1 Пример файла application.yml

В конфигурационном файле следует задавать только те настройки, которые необходимы для решения текущих бизнес-задач.

```
http-server:
  enabled: ${SERVER_ENABLED:true}
  port: ${SERVER_PORT:8081}

executor:
  reader-pool-size: ${EXECUTOR_READER_POOL_SIZE:20}

vertx:
  web-client:
    max-pool-size: 100

kafka:
  agent.topic.prefix: ${AGENT_TOPIC_PREFIX:}
  max-concurrent-handle: ${KAFKA_MAX_CONCURRENT_HANDLE:100}
```

```
  commit-interval: ${KAFKA_COMMIT_INTERVAL:5s}
  external:
    bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
    topic.prefix: ${EXTERNAL_TOPIC_PREFIX:${kafka.agent.topic.prefix}}
  enabled: ${KAFKA_ENABLED:true}
  consumer:
    blob-request: ${kafka.external.topic.prefix}blob.rq
    # максимальное количество обработчиков входящих запросов
    max-concurrent-handle: ${kafka.max-concurrent-handle}
    # периодичность фиксации оффсета обработанных сообщений
    commit-interval: ${kafka.commit-interval}
    property:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
      group.id: ${AGENT_TOPIC_PREFIX:}blob-consumer
      auto.offset.reset: earliest
      enable.auto.commit: false
  producer:
    blob-result: ${kafka.external.topic.prefix}blob.rs
    blob-error: ${kafka.external.topic.prefix}blob.err
    property:
      bootstrap.servers: ${kafka.external.bootstrap.servers}

blob:
  chunk-size: ${CHUNK_SIZE:524288}
  storage:
    client: vertx # vertx/awssdk
    protocol: ${BLOB_STORAGE_PROTOCOL:http} # use only for client vertx
    host: ${BLOB_STORAGE_HOST:localhost}
    port: ${BLOB_STORAGE_PORT:8888}
    path-prefix: ${BLOB_STORAGE_PATH_PREFIX:}
    path-postfix: ${BLOB_STORAGE_PATH_POSTFIX:}
    bucket: ${BUCKET:} # use only for client awssdk
    auth:
      type: ${BLOB_STORAGE_AUTH_TYPE:NONE}  # BASIC/TOKEN/AUTH/NONE/AWSBASICCREDENTIALS
      user: ${BLOB_STORAGE_AUTH_USER:user}  # use only for auth.type BASIC
      password: ${BLOB_STORAGE_AUTH_PASSWORD:pass} # use only for auth.type BASIC
      access-token: ${ACCESS_TOKEN:}  # use only for auth.type AWSBASICCREDENTIALS
      secret-token: ${SECRET_TOKEN:}  # use only for auth.type AWSBASICCREDENTIALS
      token: ${BLOB_STORAGE_AUTH_TOKEN:token}  # use only for auth.type TOKEN
      authorization-server:   # use only for auth.type AUTH
        protocol: ${AUTH_SERVER_PROTOCOL:http}
        host: ${AUTH_SERVER_HOST:localhost}
        port: ${AUTH_SERVER_PORT:80}
        path: ${AUTH_SERVER_PATH:oauth2/token}
        client-id: ${AUTH_SERVER_CLIENT_ID:}
        client-secret: ${AUTH_SERVER_CLIENT_SECRET:}

logging:
  request-response:
    blob-request: ${BLOB_REQUEST_LOG_ENABLED:false}
    blob-response: ${BLOB_RESPONSE_LOG_ENABLED:false}

metrics:
  port: ${METRICS_PORT:9837}
```

2.2.15.2 Параметры конфигурации

Настройка конфигурации **BLOB-адаптера** осуществляется путем редактирования параметров настроек в файле `application.yml`, где настраиваются секции:

- `http-server` - указывается порт сервера;
- `executor` - настраивается размер пула для запросов;

- `vertx` - настраиваются значения вертиклов;
- `kafka` - настраиваются параметры подключения к шине данных **Apache Kafka**;
- `blob` - настраивается подключение к Хранилищу BLOB-объектов;
- `logging` - настраивается логирование работы модуля;
- `metrics` - настраивается получение метрик.

2.2.15.2.1 Секция http-server

Секция `http-server` позволяет настраивать взаимодействие с BLOB-объектами через модуль **СМЭВ3-адаптер** по протоколу `http/https` и задавать порт, на котором будет открыт доступ к серверу.

Например:

```
http-server:
  enabled: ${SERVER_ENABLED:true}
  port: ${SERVER_PORT:8081}
```

- `enabled` - флаг активации работы с сервером;
- `port` - порт, на котором будет открыт доступ к серверу.

2.2.15.2.2 Секция executor

Секция `executor` предназначена для масштабирования нагрузки (увеличения / уменьшения) на модуль.

Например:

```
executor:
  reader-pool-size: ${EXECUTOR_READER_POOL_SIZE:20}
```

Параметры настроек

- `reader-pool-size` - размер пула для чтения **Kafka**, например `EXECUTOR_READER_POOL_SIZE:20`.

2.2.15.2.3 Секция vertx

Секция `vertx` определяет настройки количества вертиклов. Например:

```
vertx:
  web-client:
    max-pool-size: 20
```

Параметры настроек

- `max-pool-size` - максимальное значение для веб-клиента.

2.2.15.2.4 Секция kafka

Секция `kafka` предназначена для настройки параметров подключения к шине данных **Apache Kafka** (используется для взаимодействия с **СМЭВ4-адаптером**).

Например:

```
kafka:
  agent.topic.prefix: ${AGENT_TOPIC_PREFIX:}
  max-concurrent-handle: ${KAFKA_MAX_CONCURRENT_HANDLE:1000}
  commit-interval: ${KAFKA_COMMIT_INTERVAL:5s}
  external:
    bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
    topic.prefix: ${EXTERNAL_TOPIC_PREFIX:${agent.topic.prefix}}
  enabled: ${KAFKA_ENABLED:true}
  consumer:
    blob-request: ${kafka.external.topic.prefix}blob.rq
    # максимальное количество обработчиков входящих запросов
```

```
    max-concurrent-handle: ${kafka.max-concurrent-handle}
    # периодичность фиксации оффсета обработанных сообщений
    commit-interval: ${kafka.commit-interval}
    property:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
      group.id: ${AGENT_TOPIC_PREFIX:}blob-consumer
      auto.offset.reset: earliest
      enable.auto.commit: false
  producer:
    blob-result: ${kafka.external.topic.prefix}blob.rs
    blob-error: ${kafka.external.topic.prefix}blob.err
    property:
      bootstrap.servers: ${kafka.external.bootstrap.servers}
```

Параметры конфигурации

- `enabled` - флаг включения чтения из **Kafka**, например `KAFKA_ENABLED:true`;
- `blob-request` - максимальное количество обработчиков входящих запросов, например `${kafka.external.topic.prefix}blob.rq`;
- `max-concurrent-handle` - периодичность фиксации оффсета обработанных сообщений, например `${kafka.max-concurrent-handle}`

blob.rq, blob.rs, blob.err - топики взаимодействия с **Агентом СМЭВ4**.

2.2.15.2.5 Секция blob

Секция `blob` предназначена для настройки:

- размера выгружаемого чанка BLOB;
- пути к Хранилищу BLOB-объектов (GET-запрос);
- метода аутентификации для модуля BLOB-адаптера;
- получения токена;
- повторной аутентификаций при истечении времени жизни токена.

Например:

```
blob:
  chunk-size: ${CHUNK_SIZE:524288}
  storage:
    client: vertx # vertx/awssdk
    protocol: ${BLOB_STORAGE_PROTOCOL:http} # use only for client vertx
    host: ${BLOB_STORAGE_HOST:localhost}
    port: ${BLOB_STORAGE_PORT:8888}
    path-prefix: ${BLOB_STORAGE_PATH_PREFIX:}
    path-postfix: ${BLOB_STORAGE_PATH_POSTFIX:}
    bucket: ${BUCKET:} # use only for client awssdk
    auth:
      type: ${BLOB_STORAGE_AUTH_TYPE:NONE}  # BASIC/TOKEN/AUTH/NONE/AWSBASICCREDENTIALS
      user: ${BLOB_STORAGE_AUTH_USER:user}  # use only for auth.type BASIC
      password: ${BLOB_STORAGE_AUTH_PASSWORD:pass} # use only for auth.type BASIC
      access-token: ${ACCESS_TOKEN:}  # use only for auth.type AWSBASICCREDENTIALS
      secret-token: ${SECRET_TOKEN:}  # use only for auth.type AWSBASICCREDENTIALS
      token: ${BLOB_STORAGE_AUTH_TOKEN:token}  # use only for auth.type TOKEN
      authorization-server:   # use only for auth.type AUTH
        protocol: ${AUTH_SERVER_PROTOCOL:http}
        host: ${AUTH_SERVER_HOST:localhost}
        port: ${AUTH_SERVER_PORT:80}
        path: ${AUTH_SERVER_PATH:oauth2/token}
        client-id: ${AUTH_SERVER_CLIENT_ID:}
        client-secret: ${AUTH_SERVER_CLIENT_SECRET:}
```

Параметры конфигурации

- `chunk-size` - размер выгружаемого чанка BLOB, например `${CHUNK_SIZE:524288}`;
- `protocol` - протокол обмена с сервером хранилища **BLOB-объектов** (одно из значений `http` или `https`), например `BLOB_STORAGE_PROTOCOL:http`;
- `host` - имя сервера хранилища **BLOB-объектов**, например `BLOB_STORAGE_HOST:localhost`;
- `port` - порт сервера хранилища **BLOB-объектов**, если отсутствует, то следует использовать следующие порты `80` для **HTTP**, `443` для **HTTPS**, например `BLOB_STORAGE_PORT:8888`;
- `path-prefix` - путь до места хранилища **BLOB-объектов**, путь до места хранения BLOB-объекта на сервере хранилища BLOB-объектов, например `BLOB_STORAGE_PATH_POSTFIX:`;
- `path-postfix` - окончание пути, начало списка параметров, например `BLOB_STORAGE_PATH_PREFIX:`;
- `bucket` - наименование хранилища, только для клиентов AWSSDK;
- `auth` - параметры аутентификации **BLOB-адаптера**;
- `type` -тип аутентификации (`NONE` - нет, `BASIC` - по имени/паролю, `TOKEN` - по токену, `AUTH` - через сервер аутентификации), например `BLOB_STORAGE_AUTH_TYPE:NONE`;
- `user` - имя пользователя (для аутентификации `BASIC`), например `BLOB_STORAGE_AUTH_USER:user`;
- `password` - пароль (для аутентификации `BASIC`), например `BLOB_STORAGE_AUTH_PASSWORD:pass`;
- `token` - токен (для аутентификации `TOKEN`), например `BLOB_STORAGE_AUTH_TOKEN:token`;
- `protocol` - имя протокола **HTTP** или **HTTPS** (для аутентификации `AUTH`), например `AUTH_SERVER_PROTOCOL:http`;
- `host` - строка с IP или FQDN сервера авторизации (для аутентификации `AUTH`), например `AUTH_SERVER_HOST:localhost`;
- `port` - TCP-порт (для аутентификации `AUTH`), например `AUTH_SERVER_PORT:80`;
- `path` - путь на сервере (для аутентификации `AUTH`), например `AUTH_SERVER_PATH:oauth2/token`;
- `client-id` - идентификатор клиента, присвоенный **BLOB-адаптеру** в сервере авторизации (для аутентификации `AUTH`), например `AUTH_SERVER_CLIENT_ID:`;
- `client-secret` - секретный код клиента, присвоенный **BLOB-адаптеру** в сервере авторизации (для аутентификации `AUTH`), например `AUTH_SERVER_CLIENT_SECRET:`.

Пример cURL-запроса к серверу аутентификации

cURL (windows)

```
curl -X POST http://t5-avanpost-01.ru-central1.internal/oauth2/token ^
  -H "Accept: application/json"^
  -H "Content-type: application/x-www-form-urlencoded"^
  --data "grant_type=client_credentials&client_id=b0fd0f28-4b99-40d7-8dd3-
e663a6cc77d1&client_secret=Zaq1sd!sa2" ^
  -o result.txt ^
  --trace-ascii result.log
```

cURL (linux)

```
curl --request POST ¥
  --url http://t5-avanpost-01.ru-central1.internal/oauth2/token ¥
  --header 'Accept: application/json' ¥
  --header 'Content-type: application/x-www-form-urlencoded' ¥
  --data 'grant_type=client_credentials&client_id=b0fd0f28-4b99-40d7-8dd3-
e663a6cc77d1&client_secret=Zaq1sd!sa2' ¥
  -o result.txt
```

Пример cURL-запроса к Хранилищу BLOB-объектов

cURL (windows)

```
curl -X GET http://vmserv1.internal.example.com:8080/datamart/data/v1/blobs/1234567 ^
  -H "Authorization: bearer
eyJhbGciOiJSUzI1NiIsImtpZCI6IjEifQ.eyJqdGkiOiI5YmQ3MzdjZi05MDNmLTQxZTktYjI5Mi1mZmUwM2Qz
NDhkNWIiLCJleHAiOjE2NTQxMDI5NDgsImlhdCI6MTY1NDEwMTE0OCwiaXNzIjoiY2VydC5pc3N1ZXIuaG9zdCI
sImF1ZCI6IiIsImV4cGlyZXNfaW4iOjE4MDAsImNsaWVudF9pZCI6ImIwZmQwZjI4LTRiOTktNDBkNy04ZGQzLW
U2NjNhNmNjNzdkMSJ9.qi8JKlQAdMsK3fTq4H88Z5-FppaUP95OH-
rmPtCxEMmlPnyhNCRJe34aKMR5mXVldEzY1clV87-
qjWCyPLH_Zkqji1C7aQz7fMbgZixhY2wrQnXAXRfslkRe5Ph3GYYd26GvWOG1xl99AHvfDWIfI1SGcJyd0z_iOl
1GbghLvSV38MquZ8ugBdKaDjV-Ww3U_sWlJVO-
oF8xjUMYuhOSsCNxhxMng1oVwUdAUbbgoB5ldyoGTbqmbQMYvBmKBT0eZqOR6RnJEAjmfOC9YeWwADKwovFybvG
OaQZsjlaoJ2XxpmS79U7UO_6KXK1cnHfshVuB5_yUwubrRh6tRxt0CA"^
  -o result.bin ^
  --trace-ascii result.log
```

Пример настройки динамической ссылки на файлы с содержимым BLOB-полей

Пример 1

Если в Витрине поле с типом `LINK` содержит текст `12345678` и для получения содержимого BLOB надо использовать строку вызова:

```
http://aa.bb.cc:8080/api/v1/blobs/12345678
```

Настройки файла `application.yml` должны иметь следующий вид:

```
blob-storage:
  protocol: http
  host: aa.bb.cc
  port: 8080
  path-prefix: api/v1/blobs/
```

Пример 2

Если в Витрине поле с типом `LINK` содержит текст `12345678` и для получения содержимого BLOB надо использовать строку вызова:

```
https://aa.bb.cc/api/v1/blobs/12345678/data?format=jpg&size=low&backgraund=#000000,
```

Настройки файла `application.yml` должны иметь следующий вид:

```
blob-storage:
  protocol: https
  host: aa.bb.cc
  path-prefix: api/v1/blobs/
  path-postfix: /data
  params:
    format: jpg
    size: low
    backgraund: "#000000"
```

Пример 3

Если в Витрине поле с типом `LINK` содержит текст `12345678` и для получения содержимого BLOB надо использовать строку вызова:

```
http://aa.bb.cc:8080/api/v1/blobs/12345678
```

Настройки файла `application.yml` должны иметь следующий вид:

```
blob-storage:
  protocol: ${PROT:http}
  host: ${HOST:aa.bb.cc}
  port: ${PORT:80}
  path-prefix: ${PREFIX:api/v1/blobs/}
```

Пример 4

Если требуется получить строку вида:

```
https://aa.bb.cc:8080/app/{link}/download?requester_id={value}&user=fdsfs&zip=true
```

Необходимо настроить:

1. В Витрине данных поле с типом `LINK` должно содержать текст:

```
12345678/download?requester_id=ABCDEFGH
```

2. В файл `application.yml` добавить следующие настройки:

```
blob-storage:
  protocol: https
  host: aa.bb.cc
  port: 8080
  path-prefix: api/
  params:
    user: fdsfs
    zip: true
```

Указание дополнительных параметров к Хранилищу BLOB-объектов

Например

```
blob-storage:
  params:
    name1: value1
    name2: value2
```

Пример запроса

```
/files/test?name1=value1&name2=value2
```

2.2.15.2.6 Секция logging

В секции `logging` настраивается логирование работы модуля.

Например:

```
logging:
  request-response:
    blob-request: ${BLOB_REQUEST_LOG_ENABLED:false}
    blob-response: ${BLOB_RESPONSE_LOG_ENABLED:false}
```

Параметры настроек

- `blob-request` - журналировать запросы, например `BLOB_REQUEST_LOG_ENABLED:false`;
- `blob-request` - журналировать ответы, например `BLOB_RESPONSE_LOG_ENABLED:false`.

2.2.15.2.7 Секция metrics

Секция `metrics` предназначена для настройки параметров метрик.

Например:

```
metrics:
  port: ${METRICS_PORT:9837}
```

Параметры конфигурации

- `port` - Порт для метрик, например `METRICS_PORT:9837`.

2.2.16 Настройка Сервиса формирования документов

2.2.16.1 Конфигурация Сервиса Формирования документов (application.yml)

Файл `application.yml` – основной конфигурационный файл **Сервиса Формирования документов**, в котором задана логика и порядок работы сервиса:

- настройка и обработка документов;
- путь к pebble-шаблонам документов (секция `printable-forms`);
- настройка сервиса формирования подписи (`sign-service`);
- настройка подключения к базе данных (секция: `datasource`);
- настройка проверки состояния БД (секция `health`) и другие настройки необходимые для корректной работы сервиса.

2.2.16.1.1 Пример файла application.yml

В конфигурационном файле следует задавать только те настройки, которые необходимы для решения текущих бизнес-задач.

```
http-server:
  port: ${HTTP_PORT:8080}

executor:
  reader-pool-size: ${EXECUTOR_READER_POOL_SIZE:20}

prostore-rest-client:
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9195}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}

metrics:
  port: ${METRICS_PORT:9837}

vertx:
  web-client:
    max-pool-size: 20

counter-service:
  host: ${COUNTER_SERVICE_HOST:localhost}
  port: ${COUNTER_SERVICE_PORT:9000}
  serviceName: ${COUNTER_SERVICE_NAME:printableform}
  timeout: ${COUNTER_SERVICE_TIMEOUT:30}


sign-service:
  url: ${SIGN_SERVICE_URL:http://localhost:8192}
  timeout: ${SIGN_SERVICE_TIMEOUT:30}
  pool-size: ${SIGN_SERVICE_POOL_SIZE:5}

notarius:
  host: ${NOTARUIS_HOST:localhost}
  port: ${NOTARUIS_PORT:8192}
```

```
  enabled: ${NOTARIUS_ENABLED:FALSE}
  props:
    maxContentLength: ${NOTARUIS_MAX_CONTENT_LENGTH:104857600}
    signatureURI:
${NOTARUIS_SIGNATURE_URI:urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34102012-
gostr34112012-256}
    digestMethod: ${NOTARUIS_DIGEST_METHOD:http://www.w3.org/2001/04/xmldsig-
more#gostr3411}

printable-forms:
  reports:
    # имя документа
    - report: document_1
      # настройки по извлечению данных
      extract:
        # путь pebble шаблона, который будет извлекать данные
        template: extract_static.peb
      # настройки по формированию xml документа
      xml:
        # путь pebble шаблона, который будет формировать xml документ
        template: generate_xml_1.peb
        # Id подписываемого элемента, если не указано, то подписывается весь xml
        elementId: elementToSign
        # имя элемента, куда добавлять ЭП, если не указано, то в корень
        elementName: signature
        # имя файла, если не указано, то <Id_ПФ + ".xml">
        fileName: document_1.xml
      # настройки по формированию xml документа с открепленной подписью
      xml_detached_sig:
        # имя файла, если не указано, то <Id_ПФ + ".xml">
        fileName: document_1.xml
        # имя файла p7s, если не указано, то <Id_ПФ + ".p7s">
        fileSign: xmlSign.p7s
      # настройки по формированию pdf документа
      pdf:
        # путь pebble шаблона, который будет формировать pdf документ
        template: generate_pdf_1.peb
        # имя файла, если не указано, то <Id_ПФ + ".pdf">
        fileName: pdfFileName.pdf
      # настройки по формированию pdf документа с открепленной подписью
      pdf_sig:
        # путь pebble шаблона, который будет формировать подписываемый pdf
документ, задается в .pdf.template
        #  (для генерации "pdf без ЭП" и "pdf с ЭП" используется единый pebble-шаблон)
        # имя файла, если не указано, то <Id_ПФ + ".pdf">
        fileName: pdfFileName.pdf
        # имя файла p7s, если не указано, то <Id_ПФ + ".p7s">
        fileSign: pdfSign.p7s
```

2.2.16.2 Параметры конфигурации

Настройка конфигурации **Сервиса Формирования документов** осуществляется путем редактирования параметров настроек в файле `application.yml`, где настраиваются секции:

- `http-server` - указывается порт веб-сервера;
- `executor` - предназначена для указания размера пула для запросов;
- `prostore-rest-client` - настраивается блок параметров взаимодействия с **Prostore**;
- `metrics` - указывается порт для получения метрик;
- `counter-service` - указываются настройки подключения к сервису генерации номера;

- `sign-service` - указываются настройки подключения к сервису подписания документа;
- `notarius` - указываются настройки сервиса подписания и проверки подписи «Нотариус»;
- `printable-forms` - указываются настройки сервиса формирования документов

2.2.16.2.1 Секция http-server

В секции `http-server` указывается порт веб-сервера.

Например:

```
http-server:
  port: ${HTTP_PORT:8080}
```

Параметры настроек

- `port` - порт веб-сервера, например: `HTTP_PORT:8080`.

2.2.16.2.2 Секция executor

В секции `executor` указывается размер пула для запросов.

Например:

```
executor:
  reader-pool-size: ${EXECUTOR_READER_POOL_SIZE:20}
```

Параметры настроек

- `reader-pool-size` - Размер пула для чтения запросов, например `EXECUTOR_READER_POOL_SIZE:20`

2.2.16.2.3 Секция prostore-rest-client

В секции `prostore-rest-client` настраивается блок параметров взаимодействия с **Prostore**.

Например:

```
prostore-rest-client:
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9195}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}
```

Параметры настроек

- `host` - адрес Prostore, например `PS_HOST:localhost`;
- `port` - порт Prostore, например `PS_PORT:9195`;
- `max-pool-size` - максимальное число подключений к Prostore, например `PS_MAX_POOL_SIZE:8`.

2.2.16.2.4 Секция metrics

В секции `metrics` указывается порт для получения метрик.

Например:

```
metrics:
  port: ${METRICS_PORT:9837}
```

Параметры настроек

- `port` - порт для получения метрик, например `METRICS_PORT:9837`

2.2.16.2.5 Секция counter-service

В секции `counter-service` настраивается подключение к сервису генерации номера.

Например:

```
counter-service:
  host: ${COUNTER_SERVICE_HOST:localhost}
  port: ${COUNTER_SERVICE_PORT:9000}
  serviceName: ${COUNTER_SERVICE_NAME:printableform}
  timeout: ${COUNTER_SERVICE_TIMEOUT:30}
```

Параметры настроек

- `host` - адрес сервиса генерации номера, например `COUNTER_SERVICE_HOST:t5-printable-form-01.ru-central1.internal`;
- `port` - порт сервиса генерации номера, например `COUNTER_SERVICE_PORT:9000`;
- `serviceName` - значение имени сервиса, например `COUNTER_SERVICE_NAME:printableform`;
- `timeout` - таймаут на генерации номера, например `COUNTER_SERVICE_TIMEOUT:30`.

2.2.16.2.6 Секция sign-service

В секции `sign-service` настраивается подключение к сервису подписания документа.
Например:

```
sign-service:
  url: ${SIGN_SERVICE_URL:http://dev-dtm-poddagent01.ru-central1.internal:8192}
  timeout: ${SIGN_SERVICE_TIMEOUT:30}
  pool-size: ${SIGN_SERVICE_POOL_SIZE:5}
```

Параметры настроек

- `url` - URL сервиса подписания документа, например `SIGN_SERVICE_URL:http://dev-dtm-poddagent01.ru-central1.internal:8192`;
- `timeout` - таймаут на подписание документа (сек), например `SIGN_SERVICE_TIMEOUT:30`;
- `pool-size` - размер пула для сервиса подписания, например `SIGN_SERVICE_POOL_SIZE:5`.

2.2.16.2.7 Секция notarius

В секции `notarius` указываются настройки сервиса Нотариус.
Например:

```
notarius:
  host: ${NOTARUIS_HOST:dev-dtm-poddagent01.ru-central1.internal}
  port: ${NOTARUIS_PORT:8192}
  enabled: ${NOTARIUS_ENABLED:FALSE}
  props:
    maxContentLength: ${NOTARUIS_MAX_CONTENT_LENGTH:104857600}
    signatureURI:
${NOTARUIS_SIGNATURE_URI:urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34102012-
gostr34112012-256}
    digestMethod: ${NOTARUIS_DIGEST_METHOD:http://www.w3.org/2001/04/xmldsig-
more#gostr3411}
```

Параметры настроек

- `host` - адрес сервиса Нотариус, например `NOTARUIS_HOST:dev-dtm-poddagent01.ru-central1.internal`;
- `port` - порт сервиса Нотариус, например `NOTARUIS_PORT:8192`;
- `enabled` - выбор сервиса подписания между schloussler и notarius, например `NOTARIUS_ENABLED:FALSE`;

- `maxContentLength` - максимальный размер отправляемого объекта, например `NOTARUIS_MAX_CONTENT_LENGTH:1048576005`;
- `signatureURI` - URI алгоритма хэширования, например `NOTARUIS_SIGNATURE_URI:urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34102012-gostr34112012-256`;
- `digestMethod` - алгоритм хэширования, например `NOTARUIS_DIGEST_METHOD:http://www.w3.org/2001/04/xmldsig-more#gostr3411`.

2.2.16.2.8 Секция printable-forms

В секции `printable-forms` настраивается сервис формирования документов. Например:

```yaml
printable-forms:
  reports:
    # имя документа
    - report: document_1
      # настройки по извлечению данных
      extract:
        # путь pebble шаблона, который будет извлекать данные
        template: ./src/main/resources/extract_static.peb
      # настройки по формированию xml документа
      xml:
        # путь pebble шаблона, который будет формировать xml документ
        template: ./src/main/resources/generate_xml_1.peb
        # Id подписываемого элемента, если не указано, то подписывается весь xml
        elementId: elementToSign
        # имя элемента, куда добавлять ЭП, если не указано, то в корень
        elementName: signature
        # имя файла, если не указано, то <Id_ПФ + ".xml">
        fileName: document_1.xml
      # настройки по формированию pdf документа
      pdf:
        # путь pebble шаблона, который будет формировать pdf документ
        template: ./src/main/resources/generate_pdf_1.peb
        # имя файла, если не указано, то <Id_ПФ + ".pdf">
        fileName: pdfFileName.pdf
      # настройки по формированию pdf документа с откреплённой подписью
      pdf_sig:
        # путь pebble шаблона, который будет формировать подписываемый pdf
документ, задается в .pdf.template
        #  (для генерации "pdf без ЭП" и "pdf с ЭП" используется единый pebble-шаблон)

        # имя файла, если не указано, то <Id_ПФ + ".pdf">
        fileName: pdfFileName.pdf
        # имя файла p7s, если не указано, то <Id_ПФ + ".p7s">
        fileSign: pdfSign.p7s
```

> Внимание:
>
> В конфигурационном файле `application.yml` пути к файлам pebble-шаблонов должны быть либо: относительно директории запуска, либо абсолютные пути.

2.2.16.3 Примеры pebble-шаблонов для Сервиса Формирования документов

2.2.16.3.1 Возможность вызова REST-сервисов из шаблона Сервиса Формирования документов

Для вызова REST-сервисов из шаблона Сервиса Формирования документов

используется функция `callRest`.

Пример вызова функции из pebble-шаблона:

```
{% set host = "smevql-dtm-smevqlserver01.ru-central1.internal" %}
{% set port = "8080" %}
{% set route = "data" %}
    {% set rq = "${jsonRequest.replace("¥"", "¥¥¥"")}" %}

{% set varName = callRest(
        method = "POST",
        url = "http://#¥{host}:#{port}/#{route}",
        headers = "Content-Type=application/json",
        body = rq,
        responseType = "JSON"
      )
%}

{{ varName["response"]["ticket"][0]["id"]}}
```

Для асинхронного вызова (без ожидания ответа), необходимо выставить параметр `async=true`.

2.2.16.3.2 Pebble-шаблон для обработки поступившего запроса и формирования json-файла

```
{# формируем sql запрос в переменную passengersquery#}
{% var passengersquery %}
    {% if params[0] is empty %}
        select * from smart.all_types limit 10
    {% else %}
        select * from smart.all_types limit {{ params[0] }}
    {% endif %}
{% endvar %}
{# выполняем sql запрос и помещаем результат выполнения в переменную
rows.searchpassenger #}
{{ sql("searchpassenger", passengersquery) }}

{% var json_data %}
{
    "passengers": [
    {% for p in rows.searchpassenger %}
    {#  формируем json динамически  #}
        {% if loop.first %}
        {% else %}
            ,
        {% endif %}
        {
            "id": "{{ p.id }}",
            "firstname": "{{ p.firstname }}",
            "middlename": "{{ p.middlename }}",
            "lastname": "{{ p.lastname }}",
            "birthday": "{{ p.birthday }}"
        }
    {% endfor %}
    ]
}
{% endvar %}

{#выведем полученный json в неэкранированной форме#}
{{ json_data | raw }}
```

2.2.16.3.3 Pebble-шаблон для формирования xml-документа

```
<root>
    <passengers Id="elementToSign">
    {% for p in data.passengers %}
        <passenger id="{{ p.id }}">
            <firstname>{{ p.firstname }}</firstname>
            <middlename>{{ p.middlename }}</middlename>
            <lastname>{{ p.lastname }}</lastname>
            <birthday>{{ p.birthday }}</birthday>
        </passenger>
        <cert>
            <organization>{{ certification_info.subjectDN.commonName }}</organization>
            <serial>{{ dec_to_hex(certification_info.serialNumber) }}</serial>
            <issuer>{{ certification_info.issuerDN.commonName }}</issuer>
            <valid-from>{{ certification_info.notBefore | date("dd.MM.yyyy") }}</valid-
from>
            <valid-until>{{ certification_info.notAfter | date("dd.MM.yyyy") }}</valid-
until>
        </cert>
    {% endfor %}
    </passengers>
    <signature/>
</root>
```

2.2.16.3.4 Pebble-шаблон для формирования pdf-документа

```
<html>
    <head>
        <style>
            table, th, td {
            border: 1px solid black;
            }
        </style>
    </head>
    <body>
    <h3>Certificate</h3>
    <table>
        <tr>
            <th>Организация</th>
            <th>Сертификат</th>
            <th>Издатель</th>
            <th>Действителен с</th>
            <th>Действителен по</th>
        </tr>
        <tr>
            <td>{{ certification_info.subjectDN.commonName }}</td>
            <td>{{ certification_info.serialNumber }}</td>
            <td>{{ certification_info.issuerDN.commonName }}</td>
            <td>{{ certification_info.notBefore }}</td>
            <td>{{ certification_info.notAfter }}</td>
        </tr>
    </table>
    <h3>Passengers</h3>
    <table>
        <tr>
            <th>id</th>
            <th>firstname</th>
            <th>middlename</th>
            <th>lastname</th>
            <th>birthday</th>
        </tr>
```

```
        {% for p in data.passengers %}
        <tr>
            <td>{{ p.id }}</td>
            <td>{{ p.firstname }}</td>
            <td>{{ p.middlename }}</td>
            <td>{{ p.lastname }}</td>
            <td>{{ p.birthday }}</td>
        </tr>
        {% endfor %}
    </table>
    </body>
</html>
```

2.2.17 Настройка REST-адаптера

2.2.17.1 Конфигурация REST-адаптера

Файл `application.yml` – основной конфигурационный файл модуля, в котором задана его логика и порядок работы модуля

2.2.17.1.1 Пример файла application.yml

```yaml
http-server:
  port: 8082

prostore-rest-client:
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9195}
  default-schema: ${PS_SCHEMA:demo_dev}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}

swagger:
  file-path: sample.yaml
  #Map operationId -> templatefile
  templates:
    execquery_get: sample.peb
    execquery_post: sample.peb
    execquery_get_params: sample.peb
    execquery_post_params: sample.peb

metrics:
  port: ${METRICS_PORT:9837}

vertx:
  web-client:
    max-pool-size: 20

  # Параметры подключения к сервису печатных форм. Указывается при использовании функции toSpf
spf:
  host: ${SPF_HOST:localhost}
  port: ${SPF_PORT:8080}
  # Дополнительные параметры. Указываются ключ-значения сертификатов, необходимых для сервиса ПФ
  params: {}
```

2.2.17.2 Параметры конфигурации

Настройка конфигурации **REST-адаптера** осуществляется путем редактирования параметров настроек в файле `application.yml`, где настраиваются секции:

- `http-server` - порт сервера;
- `prostore-rest-client` - блок параметров конфигурирования взаимодействия с **Prostore**;
- `swagger` - пример файла конфигурации конечных точек;
- `metrics` - настраивается получение метрик;
- `vertx` - настраиваются значения вертиклов;
- `spf` - Параметры подключения к сервису печатных форм. Указывается при использовании функции toSpf.

2.2.17.2.1 Секция http-server

Секция `http-server` предназначена для настройки порта и протокола передачи данных (одно из значений `http` или `https`).

Например:

```
http-server:
  port: 8082
```

Параметры настроек

- `port` - порт веб-сервера, например: `8082`.

2.2.17.2.2 Секция prostore-rest-client

В секции `prostore-rest-client` настраиваются параметры конфигурирования взаимодействия с **Prostore**.

Например:

```
prostore-rest-client:
  host: ${PS_HOST:t5-prostore-01.ru-central1.internal}
  port: ${PS_PORT:9195}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}
```

Параметры настроек

- `host` - адрес **Prostore**, например `PS_HOST:t5-prostore-01.ru-central1.internal`;
- `port` - порт **Prostore**, например `PS_PORT:9195`;
- `max-pool-size` - максимальное число подключений к **Prostore**, например `PS_MAX_POOL_SIZE:8`.

2.2.17.2.3 Секция swagger

В секции `swagger` приведен пример файла конфигурации конечных точек.

Например:

```
swagger:
  file-path: sample.yaml
  #Map operationId -> templatefile
  templates:
    execquery_get: sample.peb
    execquery_post: sample.peb
    execquery_get_params: sample.peb
    execquery_post_params: sample.peb
```

Параметры настроек

- `file-path` - пример файла конфигурации конечных точек;
- `templates` - шаблоны парсинга и обогащения запросов.

2.2.17.2.4 Секция metrics

В секции `metrics` настраиваются параметры метрик.

Например:

```
metrics:
  port: ${METRICS_PORT:9837
```

Параметры конфигурации

- `port` - порт для получения метрик, например `9837`.

2.2.17.2.5 Секция spf

В секции spf указываются (при использовании функции toSpf) параметры подключения к сервису печатных форм.

Например:

```
spf:
  host: ${SPF_HOST:localhost}
  port: ${SPF_PORT:8080}
  # Дополнительные параметры. Указываются ключ-значения сертификатов, необходимых для сервиса ПФ
  params: {}
```

Параметры настроек

- `host` - адрес Сервиса печатных форм, например `SPF_HOST:localhost`;
- `port` - порт Сервиса печатных форм, например `SPF_PORT:8080`;
- `params` - Дополнительные параметры (ключ-значения сертификатов, необходимых для сервиса ПФ).

2.2.17.3 Конфигурационный файл с конечными точками

Для доступа к конечным точкам отредактируйте файл `sample.yaml`, описав все необходимые API в соответствии с приведенным в файле шаблоном (шаблон соответствует спецификации OpenAPI 3.0 https://swagger.io/specification/).

Пример файла `sample.yaml` со всеми конфигурируемыми атрибутами:

```
openapi: 3.0.0
info:
  title: Sample API
  version: 0.1.9
servers:
  - url: /
paths:
  /test/query:
    get:
      summary: Returns some value
      operationId: execquery_get
      responses:
        '200': # status code
          description: A JSON array of user names
          content:
            application/json:
              schema:
                type: string

    post:
      summary: Returns some value
      operationId: execquery_post
      responses:
        '200':    # status code
```

```
        description: A JSON array of user names
        content:
          application/json:
            schema:
              type: string
/test/query/{id}:
  post:
    summary: Returns some value
    operationId: execquery_post_params
    parameters:
      - name: id
        in: path
        required: true
        schema:
          type: string
          format: utf8
    responses:
      '200': # status code
        description: A JSON array of user names
        content:
          application/json:
            schema:
              type: string
  get:
    summary: Returns some value
    operationId: execquery_get_params
    parameters:
      - name: id
        in: path
        required: true
        schema:
          type: string
          format: utf8
    responses:
      '200': # status code
        description: A JSON array of user names
        content:
          application/json:
            schema:
              type: string
```

Секция `servers`

- `url: /` - корневой путь сервера.

Секция: `paths`

- `/test/query` - путь запроса;
- `get` - тип запроса (`get`, `post` и т.д.);
- `operationId` - определение `operationId` для связки с файлом шаблона;
- `responses` - описание параметров ответа.

2.2.17.4 Шаблоны

Для парсинга и обогащения запросов используются шаблоны.

В качестве языка шаблонов используется `pebble` с дополнениями:

- функция `xpath(expression)` - возвращает вычисленное выражение на входящем документе;
- тэг `{% mtom %} some content {% endmtom %}` - отправляет внутренность вложением;
- функция `sql(var, sql, param1, param2, ...)` - выполняет sql-запрос с параметрами, результат записывается в переменную `var`.

Пример формирования шаблона приведен в файле `sample.peb`.

```
{% var restrictionsquery %}
   {% if id is empty %}
      select * from passenger
   {% else %}
      select * from passenger where passenger.id = '{{ id }}'
   {% endif %}
{% endvar %}
{{ sql("searchingpassenger", restrictionsquery) }}


{% for ts in rows.searchingpassenger %}
   id = {{ ts.id }} and name = {{ ts.firstname }}
{% endfor %}
```

2.2.18 Настройка Counter-provider - Сервиса генерации уникального номера

2.2.18.1 Конфигурация модуля Counter-Provider (application.yml)

Файл `application.yml` – основной конфигурационный файл модуля, в котором задана его логика и порядок работы сервиса:

- среда разработки;
- настройка счетчика;
- настройка подключения к **Zookeeper**, а также другие настройки необходимые для корректной работы сервиса.

2.2.18.2 Пример файла application.yml

В конфигурационном файле следует задавать только те настройки, которые необходимы для решения текущих бизнес-задач.

```
environment:
  name: ${ENVIRONMENT_NAME:test}

http-server:
  port: ${HTTP_PORT:9000}

counter:
  start-number: ${COUNTER_START_NUMBER:1}
  retry-after-failure: ${COUNTER_RETRY_AFTER_FAILURE:3}
  update-timeout: ${COUNTER_UPDATE_TIMEOUT:}
  reset-period: ${COUNTER_RESET_PERIOD:}
  increment-gap: ${INCREMENT_GAP:1}

zookeeper:
  connection-string: ${ZOOKEEPER_DS_ADDRESS:localhost}
  connection-timeout-ms: ${ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000}
  session-timeout-ms: ${ZOOKEEPER_DS_SESSION_TIMEOUT_MS:86400000}
  chroot: ${ZOOKEEPER_DS_CHROOT:/adapter}

persistence-mode: ${PERSISTENCE_MODE:prostore} # prostore | zookeeper

prostore-rest-client:
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9195}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:5}
  persistence-datamart: ${PERSISTENCE_DATAMART:persistence}
```

```
  datasource: ${PERSISTENCE_DATASOURCE:} # по умолчанию пусто, тогда берется
единственный датасорс из настроек Простора
  counter-prefix: ${COUNTER_PREFIX:counter_}


migration:
  enabled: ${MIGRATION_ENABLE:false}
  old-connection-string: ${OLD_ZOOKEEPER_DS_ADDRESS:localhost}

backup:
  mode: ${BACKUP_MODE:rest} # kafka | rest
  rest:
    uri: ${BACKUP_URI:/backup}
  zk-path: ${COUNTER_BACKUP_ZK_PATH:/${environment.name}/counter-provider/counters}
  commandTopic: ${BACKUP_COMMAND_TOPIC:adapter.command}
  backupTopic: ${BACKUP_TOPIC:adapter.backup}
  statusTopic: ${STATUS_TOPIC:adapter.status}
  kafka:
    consumer:
      property:
        bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
        group.id: ${COUNTER_BACKUP_GROUP_ID:counter_provider_adapter_command}
        auto.offset.reset: latest
    producer:
      property:
        bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}


metrics:
  port: ${METRICS_PORT:9837}
```

2.2.18.3 Параметры конфигурации

Настройка конфигурации **Сервиса генерации уникального номера** осуществляется путем редактирования параметров настроек в файле `application.yml`, где настраиваются секции:

- `environment` - указывается среда разработки;
- `http-server` - указывается порт веб-сервера;
- `counter` - указываются настройки счетчика;
- `zookeeper` - определяет настройки подключения к **Zookeeper**;
- `persistence-mode` - настройки хранения данных;
- `prostore-rest-client` - блок параметров конфигурирования взаимодействия с **ProStore**;
- `migration` - настройки миграции;
- `backup` - настройки бекапирования;
- `metrics` - настройка порта для получения метрик.

2.2.18.3.1 Секция environment

В секции `environment` указывается среда разработки (dev, test, stable, prod)
Например:

```
environment:
  name: ${ENVIRONMENT_NAME:test}
```

Параметры настроек

- `name` - среда разработки, например `ENVIRONMENT_NAME:test`.

2.2.18.3.2 Секция http-server

В секции `http` указывается порт веб-сервера.

Например:

```
http-server:
  port: ${HTTP_PORT:9000}
```

Параметры настроек

- `port` - порт веб-сервера, например: `HTTP_PORT:9000`.

2.2.18.3.3 Секция counter

В секции `counter` можно настраивать начальный номер счетчика, а также количество попыток записи счетчика после ошибки обновления.

Например:

```
counter:
  start-number: ${COUNTER_START_NUMBER:1}
  retry-after-failure: ${COUNTER_RETRY_AFTER_FAILURE:3}
  update-timeout: ${COUNTER_UPDATE_TIMEOUT:}
  reset-period: ${COUNTER_RESET_PERIOD:}
  increment-gap: ${INCREMENT_GAP:1}
```

Параметры настроек

- `start-number` - начальный номер счетчика, например `COUNTER_START_NUMBER:1`;
- `retry-after-failure`- количество попыток записи счетчика после ошибки обновления, например `COUNTER_RETRY_AFTER_FAILURE:3`;
- `update-timeout` - таймаут обновления счетчика, например `COUNTER_UPDATE_TIMEOUT:`;
- `reset-period` - период сброса счетчика, например `COUNTER_RESET_PERIOD:`;
- `increment-gap` - шаг инкремента, например `INCREMENT_GAP:1`.

2.2.18.3.4 Секция zookeeper

В секции `zookeeper` настраиваются параметры подключения к **Zookeeper**.

Например:

```
zookeeper:
  connection-string: ${ZOOKEEPER_DS_ADDRESS:t5-adsp-01.ru-central1.internal}
  connection-timeout-ms: ${ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000}
  session-timeout-ms: ${ZOOKEEPER_DS_SESSION_TIMEOUT_MS:86400000}
  chroot: ${ZOOKEEPER_DS_CHROOT:/adapter}
```

Параметры настроек

- `connection-string` - Подключение к Zookeeper DS, например `ZOOKEEPER_DS_ADDRESS:t5-adsp-01.ru-central1.internal`;
- `connection-timeout-ms` - Zookeeper DS таймаут подключения, например `ZOOKEEPER_DS_CONNECTION_TIMEOUT_MS:30000`;
- `session-timeout-ms` - Zookeeper DS таймаут сессии, например `ZOOKEEPER_DS_SESSION_TIMEOUT_MS:86400000`;
- `chroot` - Zookeeper DS chroot path, например `ZOOKEEPER_DS_CHROOT:/adapter`.

2.2.18.3.5 Секция persistence-mode

В секции `persistence-mode` указывается настройка хранения данных: или в Prostore или в Zookeeper. В случае выбора Prostore автоматически создаются необходимые таблицы.

Например:

```
persistence-mode: ${PERSISTENCE_MODE:prostore} # prostore | zookeeper
```

Параметры настроек

- `persistence-mode` - настройка хранения данных, например `PERSISTENCE_MODE:prostore`.

2.2.18.3.6 Секция prostore-rest-client

В секции `prostore-rest-client` реализован блок параметров конфигурирования взаимодействия с **ProStore**.

Например:

```
prostore-rest-client:
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9195}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:5}
  persistence-datamart: ${PERSISTENCE_DATAMART:persistence}
  datasource: ${PERSISTENCE_DATASOURCE:}
  counter-prefix: ${COUNTER_PREFIX:counter_}
```

Параметры настроек

- `host` - адрес **ProStore**, например `PS_HOST:localhost`;
- `port` - порт **ProStore**, например `PS_PORT:9195`;
- `max-pool-size` - максимальное число подключений к **ProStore**, например `PS_MAX_POOL_SIZE:8`;
- `persistence-datamart` - настройка хранения данных, например `{PERSISTENCE_DATAMART:persistence}`;
- `datasource` - источник данных, например `PERSISTENCE_DATASOURCE:`, по умолчанию пусто, в этом случае берется единственный датасорс из настроек Простора.

2.2.18.3.7 Секция migration

В секции `migration` настраиваются миграции **Zookeeper** для задачи бекапирования.

Например:

```
migration:
  enabled: ${MIGRATION_ENABLE:false}
  old-connection-string: ${OLD_ZOOKEEPER_DS_ADDRESS:localhost}
```

Параметры настроек

- `enabled` - подключение миграции, например `{MIGRATION_ENABLE:false}`;
- `old-connection-string` - адрес **Zookeeper**, например `{OLD_ZOOKEEPER_DS_ADDRESS:localhost}`.

2.2.18.3.8 Секция backup

В секции `backup` настраивается бекапирования модуля.

Например:

```
backup:
  mode: ${BACKUP_MODE:rest} # kafka | rest
  rest:
    uri: ${BACKUP_URI:/backup}
  zk-path: ${COUNTER_BACKUP_ZK_PATH:/${environment.name}/counter-provider/counters}
  commandTopic: ${BACKUP_COMMAND_TOPIC:adapter.command}
  backupTopic: ${BACKUP_TOPIC:adapter.backup}
  statusTopic: ${STATUS_TOPIC:adapter.status}
  kafka:
    consumer:
      property:
        bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
        group.id: ${COUNTER_BACKUP_GROUP_ID:counter_provider_adapter_command}
        auto.offset.reset: latest
    producer:
      property:
        bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:localhost:9092}
```

Параметры настроек

- `mode` - режим бекапирования, например `BACKUP_MODE:rest`;
- `uri` - путь к файлу бекапирования в случае выбора REST-сервисов для режима бэкапирования;
- `zk-path` - путь к корневой ноде zookeeper для бэкапирования, например `{COUNTER_BACKUP_ZK_PATH:/${environment.name}/counter-provider/counters}`;
- `commandTopic` - топик команд бэкапирования, например: `{BACKUP_COMMAND_TOPIC:adapter.command}`;
- `backupTopic` - топик для отправки сохраненных данных, например: `{BACKUP_TOPIC:adapter.backup}`;
- `statusTopic` - топик для отправки статусов бэкапирования, например: `{STATUS_TOPIC:adapter.status}`.

2.2.18.3.9 Секция metrics

В секции `metrics` настраивается порт получения метрик.

Например:

```
metrics:
  port: ${METRICS_PORT:9837}
```

Параметры настроек

- `port` - Порт для получения метрик, например `{METRICS_PORT:9837}`.

2.2.19 Настройка утилиты Backup manager

2.2.19.1 Конфигурация утилиты Backup manager (application.yml)

Файл `application.yml` – основной конфигурационный файл утилиты, в котором задана её логика и порядок работы:

В конфигурации утилиты сохранены:

- путь к **DTM-tools** и параметры необходимые для работы утилиты **DTM-tools**:
  - маска топиков для бекапирования данных СУБД backend.backup.(datamart);
  - адрес Prostore (с переопределением через команду запуска);
- адрес Kafka и имена топиков для:
  - передачи команд модулям адаптера;
  - получения статусов модулей адаптера;

- бекапирования данных бекенда Витрины;
- бекапирования данных модулей адаптеров;

– путь к рабочей директории резервного копирования (доступный файловому коннектору) с переопределением через команду запуска;
– хранимый объем топика (значение по умолчанию 1Тб);
– время хранения топиков бекапа (значение по умолчанию 3 суток).

2.2.19.2 Пример файла application.yml

В конфигурационном файле следует задавать только те настройки, которые необходимы для решения текущих бизнес-задач.

```
logging:
  level:
    org.springframework: OFF
    org.apache: OFF
    ru.itone.dtm.BackupApplicationKt: OFF

spring:
  main:
    banner-mode: OFF

config:
  dtmToolsPath: ${BM_DTM_TOOLS_PATH:./}
  prostorePath: ${BM_PROSTORE_PATH:localhost:9094}
  kafkaBrokers: ${BM_KAFKA_BROKERS:localhost1:9092, localhost2:9092}
  backupDir: ${BM_BACKUP_DIR:./backup}
  backupArchive: ${BM_BACKUP_ARCHIVE:./backup.zip}
  adapterBackupTimeout: ${BM_ADAPTER_BACKUP_TIMEOUT:1}s
  topic:
    adapterCommand: ${BM_ADAPTER_COMMAND_TOPIC:adapter.command}
    adapterCommandBroadcast:
${BM_ADAPTER_COMMAND_BROADCAST_TOPIC:adapter.command.broadcast}
    adapterStatus: ${BM_ADAPTER_STATUS_TOPIC:adapter.status}
    adapterBackup: ${BM_ADAPTER_BACKUP_TOPIC:adapter.backup}
    backendBackupPrefix: ${BM_BACKEND_BACKUP_TOPIC_PREF:backend.backup.}
    backendBackupRetentionBytes:
${BM_BACKEND_BACKUP_TOPIC_RETENTION_BYTES:1000000000000}
    backendBackupRetentionMs: ${BM_BACKEND_BACKUP_TOPIC_RETENTION_MS:259200000}
    initializationStatusTimeout: ${BM_MODULES_WAIT_INIT_TIMEOUT:15}s
    executionStatusTimeout: ${BM_MODULES_WAIT_EXECUTE_TIMEOUT:120}s

backup:
  mode: [rest, kafka] # kafka, rest
  rest-services:
    counter-provider: localhost:9000/backup
    rest-uploader: localhost:8081/backup
```

2.2.19.3 Параметры конфигурации

Настройка конфигурации **Backup manager** осуществляется путем редактирования параметров настроек в файле `application.yml`, где настраиваются секции:

– `logging` - настраивается логирование работы модуля;
– `spring` - настройки фреймворка Spring;
– `config` - указываются настройки утилиты;
– `backup` - бекапирования данных СУБД.

2.2.19.3.1 Секция logging

В секции `logging` настраивается логирование работы модуля

Например:

```
logging:
  level:
      org.springframework: OFF
      org.apache: OFF
      ru.itone.dtm.BackupApplicationKt: OFF
```

Параметры настроек

- `org.springframework` - логирование фреймворка Spring, например `OFF`;
- `org.apache` - логирование Apache, например `OFF`;
- `ru.itone.dtm.BackupApplicationKt` - логирование утилиты, например `OFF`.

2.2.19.3.2 Секция spring

В секции `spring` указываются настройки фреймворка Spring.

Например:

```
spring:
  main:
      banner-mode: OFF
```

Параметры настроек

- `banner-mode` - режим баннера, например: `OFF`.

2.2.19.3.3 Секция config

В секции `config` указываются настройки утилиты.

Например:

```
config:
  dtmToolsPath: ${BM_DTM_TOOLS_PATH:./}
  prostorePath: ${BM_PROSTORE_PATH:localhost:9094}
  kafkaBrokers: ${BM_KAFKA_BROKERS:localhost1:9092, localhost2:9092}
  backupDir: ${BM_BACKUP_DIR:./backup}
  backupArchive: ${BM_BACKUP_ARCHIVE:./backup.zip}
  adapterBackupTimeout: ${BM_ADAPTER_BACKUP_TIMEOUT:1}
  topic:
      adapterCommand: ${BM_ADAPTER_COMMAND_TOPIC:adapter.command}
      adapterCommandBroadcast:
${BM_ADAPTER_COMMAND_BROADCAST_TOPIC:adapter.command.broadcast}
      adapterStatus: ${BM_ADAPTER_STATUS_TOPIC:adapter.status}
      adapterBackup: ${BM_ADAPTER_BACKUP_TOPIC:adapter.backup}
      backendBackupPrefix: ${BM_BACKEND_BACKUP_TOPIC_PREF:backend.backup.}
      backendBackupRetentionBytes:
${BM_BACKEND_BACKUP_TOPIC_RETENTION_BYTES:1000000000000}
      backendBackupRetentionMs: ${BM_BACKEND_BACKUP_TOPIC_RETENTION_MS:259200000}
      initializationStatusTimeout: ${BM_MODULES_WAIT_INIT_TIMEOUT:15}
      executionStatusTimeout: ${BM_MODULES_WAIT_EXECUTE_TIMEOUT:120}
```

Параметры настроек

- `dtmToolsPath` - адрес dtmTools, например: `BM_DTM_TOOLS_PATH:./`;
- `prostorePath` - адрес Prostore, например: `BM_PROSTORE_PATH:localhost:9094`;
- `kafkaBrokers` - адрес брокеров Kafka, например: `BM_KAFKA_BROKERS:localhost1:9092, localhost2:9092`;
- `backupDir` - директория хранения файла бэкапа, например: `BM_BACKUP_DIR:./backup`;
- `backupArchive` - архив бэкапа, например: `BM_BACKUP_ARCHIVE:./backup.zip`;
- `adapterBackupTimeout` - тайм-аут резервного копирования, например: `BM_ADAPTER_BACKUP_TIMEOUT:1`;

- `topic` - имена топиков, например:
  - `adapterCommandBroadcast` - передачи команд модулям адаптера, например: `BM_ADAPTER_COMMAND_BROADCAST_TOPIC:adapter.command.broadcast`;
  - `adapterStatus` - получения статусов модулей адаптера, например: `BM_ADAPTER_STATUS_TOPIC:adapter.status`;
  - `aadapterBackup` - бекапирования данных модулей адаптеров, например: `BM_ADAPTER_BACKUP_TOPIC:adapter.backup`.

2.2.19.3.4 Секция backup

В секции `backup` указываются настройки бэкапа Kafka или REST-сервисов.

Например:

```
backup:
  mode: [rest, kafka] # kafka, rest
  rest-services:
    counter-provider: localhost:9000/backup
    rest-uploader: localhost:8081/backup
```

Параметры настроек

- `mode` - режим бэкапирования;
- `counter-provider` местонахождения файла бэкапа для сервиса Counter-provider;
- `rest-uploader` - местонахождения файла бэкапа для сервиса Rest-uploader.

2.2.20 Настройка Arenadata Cluster Manager (ADCM)

Подробная инструкция по настройке Arenadata Cluster Manager (ADCM) приведена в официальной документации разработчика на странице https://docs.arenadata.io/adcm/.

2.2.21 Настройка Arenadata Streaming (ADS)

Инструкция по настройке Arenadata Streaming (ADS) через Arenadata Cluster Manager (ADCM) приведена в официальной документации к Arenadata Cluster Manager (ADCM) на странице https://docs.arenadata.io/ads/AdminGuide/Config_ADCM.html.

2.2.22 Настройка сервиса журналирования

Сервис журналирования позволяет работать с логами прикладных модулей запущенных в средах WildFly и Kubernetes/OpenShift.

Настройка сервиса журналирования должна быть применена к каждому модулю.

Ниже приведены шаги по настройке сервиса журналирования.

1. Добавьте зависимости в проект:

```
<parent>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-parent</artifactId>
   <version>2.3.4.RELEASE</version>
   <relativePath/> <!-- lookup parent from repository -->
</parent>
<modelVersion>4.0.0</modelVersion>

<artifactId>logger-test</artifactId>

<properties>
   <java.version>11</java.version>
   <spring-cloud.version>Hoxton.SR5</spring-cloud.version>
</properties>
```

```
<dependencies>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
   </dependency>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-validation</artifactId>
   </dependency>

   <dependency>
      <groupId>ch.qos.logback</groupId>
      <artifactId>logback-classic</artifactId>
      <version>1.2.3</version>
   </dependency>

   <dependency>
      <groupId>ch.qos.logback.contrib</groupId>
      <artifactId>logback-json-classic</artifactId>
      <version>0.1.5</version>
   </dependency>

   <dependency>
      <groupId>ch.qos.logback.contrib</groupId>
      <artifactId>logback-jackson</artifactId>
      <version>0.1.5</version>
   </dependency>
   <dependency>
      <groupId>net.logstash.logback</groupId>
      <artifactId>logstash-logback-encoder</artifactId>
      <version>6.3</version>
   </dependency>
   <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <version>1.18.12</version>
      <scope>provided</scope>
   </dependency>
   <dependency>
      <groupId>io.springfox</groupId>
      <artifactId>springfox-boot-starter</artifactId>
      <version>3.0.0</version>
   </dependency>
   <dependency>
      <groupId>org.codehaus.janino</groupId>
      <artifactId>janino</artifactId>
      <version>3.0.6</version>
   </dependency>
   <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-sleuth</artifactId>
   </dependency>
   <dependency>
      <groupId>org.springframework.kafka</groupId>
      <artifactId>spring-kafka</artifactId>
      <version>2.5.9.RELEASE</version>
   </dependency>

</dependencies>
<dependencyManagement>
   <dependencies>
```

```
      <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-dependencies</artifactId>
            <version>${spring-cloud.version}</version>
            <type>pom</type>
            <scope>import</scope>
      </dependency>
   </dependencies>
</dependencyManagement>
```

2. Подключите FluentBit к приложению как отдельный контейнер в OpenShift:

```
containers:
 - name: fluent-bit
   image: fluent/fluent-bit:latest
   volumeMounts: #перенос конфигураций, и лог файла из проекта в контейнер
     - name: logger
       mountPath: /fluent-bit/logger/
     - name: fluent-bit-config
       mountPath: /fluent-bit/etc/
   terminationMessagePolicy: File
   envFrom:
     - configMapRef:
         name: logger-fluent-bit-config-env
```

3. Создайте файл `logback.xml` для логирования приложения (подробнее см. документацию):

> **Внимание:**
>
> Обязательно в appender FILE_FLUENT прописать путь до конфигураций FluentBit, иначе в контейнер логи не подтянутся. Например, `<file>name-project/docker/fluentbit/conf/log.log</file>`.

```
<configuration debug="true">
   <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
      <layout class="ch.qos.logback.classic.PatternLayout">
            <pattern>
               <Pattern>
                  %d{yyyy-MM-dd HH:mm:ss}%-5level %logger{36} - %msg%n
               </Pattern>
            </pattern>
      </layout>
   </appender>

   <appender name="FILE_FLUENT" class="ch.qos.logback.core.FileAppender">
      <file>docker/fluentbit/conf/log.log</file>
      <append>false</append>
      <layout class="ch.qos.logback.classic.PatternLayout">
            <pattern>
               <Pattern>
                  x-b3-traceid=%X{X-B3-TraceId:-} x-b3-spanid=%X{X-B3-SpanId:-} x-b3-
parentspanid=%X{X-B3-ParentSpanId:-} x-b3-sampled=%X{X-B3-Sampled:-} x-b3-flags=%X{X-
B3-Flags:-} caller_file_name=%file serverEventDatetime="%d" logLevel=%level
threadName=%thread  message="%replace(%replace(%m){'¥n','¥u2028'}){'¥"','¥''}"
exception="%replace(%replace(%ex){'¥"','¥u2028'}){'¥n','¥u2028'}%nopex" callerLine=%L
callerMethod="%replace(%caller){'¥n','¥u2028'}" loggerName="%10.10logger"
callerClass=%logger{40} levelStr="%level" levelInt="%level" mdc= ¥n
               </Pattern>
            </pattern>
      </layout>
   </appender>
```

```
   <root level="debug" additivity="false">
      <appender-ref ref="STDOUT"/>
      <appender-ref ref="FILE_FLUENT"/>
   </root>
</configuration>
```

4. Добавьте описание конфигурации для FluentBit (подробнее см документацию):

```
[SERVICE]
      Flush        1
      Log_Level    info
      Daemon       off
      Parsers_File /fluent-bit/etc/parsers.conf
[INPUT]
      Name   tail
      Path       /fluent-bit/logger/log.log
      Tag    kafka-efs
      Buffer_Chunk_Size 400k
      Buffer_Max_Size 6MB
      Mem_Buf_Limit 6MB
      Parser logfmt
      Refresh_Interval 20

[FILTER]
      Name record_modifier
      Match      kafka-efs
      Record     subsystem ${SUBSYSTEM}
      Record     distribVersion ${DISTRIBVERSION}
      Record     deploymentUnit ${DEPLOYMENTUNIT}
      Record     hostName ${HOSTNAME}
      Record     ipAddress ${IPADDRESS}

[FILTER]
      Name       modify
      Match      kafka-efs
      Hard_copy callerClass className

[FILTER]
      Name       record_modifier
      Match      kafka-efs
      Whitelist_key serverEventDatetime
      Whitelist_key subsystem
      Whitelist_key distribVersion
      Whitelist_key deploymentUnit
      Whitelist_key hostName
      Whitelist_key ipAddress
      Whitelist_key logLevel
      Whitelist_key className
      Whitelist_key threadName
      Whitelist_key message
      Whitelist_key x-b3-traceid
      Whitelist_key x-b3-spanid
[FILTER]
      Name    lua
      Match   kafka-efs
      script  convert_date.lua
      call    convert_date_efs

[OUTPUT]
      Name  stdout
      Match kafka-efs
      Format json
```

```
      json_date_key time

[OUTPUT]
      Name  http
      Match kafka-efs
      Host logstash-service-gt-tatarstan-test-efs.apps.ocp-public.sbercloud.ru
      Port 80
      Format json
      json_date_key time
```

Для правильной работы подключите `parsers.conf`.

```
[PARSER]
      Name        logfmt
      Format      logfmt
```

5. Настройте форматирование даты:

```
function convert_date_efs(tag, timestamp, record)
  local pattern = "(%d+)-(%d+)-(%d+) (%d+):(%d+):(%d+),(%d+)"
  dt_str = record["serverEventDatetime"]
  local year, month, day, hour, minute, seconds, milliseconds = dt_str:match(pattern)
  ts = os.time{year = year, month = month, day = day, hour = hour, min = minute, sec =
seconds }
  ts = (ts * 1000) + milliseconds
  record["serverEventDatetime"] = ts
  return 2, timestamp, record
end

function convert_date_pprb(tag, timestamp, record)
  local pattern = "(%d+)-(%d+)-(%d+) (%d+):(%d+):(%d+),(%d+)"
  dt_str = record["serverEventDatetime"]
  local year, month, day, hour, minute, seconds, milliseconds = dt_str:match(pattern)
  ts = os.time{year = year, month = month, day = day, hour = hour, min = minute, sec =
seconds }
  ts = (ts * 1000) + milliseconds
  record["timestamp"] = ts
  return 2, timestamp, record
end

function convert_date_logstash(tag, timestamp, record)
  local pattern = "(%d+)-(%d+)-(%d+) (%d+):(%d+):(%d+),(%d+)"
  dt_str = record["serverEventDatetime"]
  local year, month, day, hour, minute, seconds, milliseconds = dt_str:match(pattern)
  ts = os.time{year = year, month = month, day = day, hour = hour, min = minute, sec =
seconds }
  ts = (ts * 1000) + milliseconds
  record["time"] = ts
  return 2, timestamp, record
end
```

6. Добавьте параметры модуля:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: logger-fluent-bit-config-env
data:
  MODULEID: 1.0.0
  MODULEVERSION: 1.0.0
  NODEID: 12345
  HOSTADDRESS: 0.0.0.0
  SUBSYSTEM: LOGGER-TEST
  DISTRIBVERSION: 1.0.0
  DEPLOYMENTUNIT: TEST-UNIT
  IPADDRESS: 0.0.0.1
```

7. Настройте конфигурацию для OpenShift. Для того, чтобы Prometheus мог идентифицировать сервис и собирать с него информацию, создайте **Service** и укажите путь, на котором располагаются метрики приложения.

```
apiVersion: v1
kind: Service
metadata:
name: monitoring-rest
annotations:
   description: 'Exposes Prometheus App by CLuster Ip'
   prometheus.io.scrape: 'true'
   prometheus.io.path: '/monitoring-rest/actuator/prometheus'
   prometheus.io.port: '8081'
labels:
   app: monitoring-rest
spec:
type: LoadBalancer
ports:
   - name: http
     port: 9837
     targetPort: 9837
selector:
   app: monitoring-rest
```

2.2.23 Установка компонента сбора данных запросов и ответов Витрины данных

Компонент сбора данных запросов и ответов Витрины данных реализован с целью проведения бизнес-мониторинга ИЭП процессов обработки запросов типовым ПО витрины данных, как в целом, так и в части функционирования отдельных витрин для последующей передачи данных в СЦЛ.

2.2.23.1 Процесс установки

Общий процесс установки состоит из следующих действий:

1. Настройка логирования модулей.
2. Установка и настройка Vector.
3. Установка и настройка HaProxy.
4. Установка и настройка fluentbit.

2.2.23.1.1 Настройка логирования модулей

Необходимо настроить формирование логов в формате JSON на стороне модулей:

- **СМЭВ4-адаптер - Модуль исполнения запросов**;

- **СМЭВ4-адаптер – Модуль MPPR**;
- **BLOB-адаптер**;
- **СМЭВ4-адаптер-Модуль подписок**;
- **Сервис формирования документов**

Для этого в файле `logback.xml` включите параметр `net.logstash.logback.encoder.LogstashEncoder`.

Пример logback.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
    <configuration>
    <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>logs/application.log</file>
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <!-- daily rollover -->
            <fileNamePattern>logs/application.%d{yyyy-MM-dd}.log</fileNamePattern>
            <!-- keep 30 days' worth of history capped at 3GB total size -->
            <maxHistory>30</maxHistory>
            <totalSizeCap>3GB</totalSizeCap>
        </rollingPolicy>
        <encoder class="net.logstash.logback.encoder.LogstashEncoder" />
    </appender>

    <root level="INFO">
        <appender-ref ref="FILE" />
    </root>
</configuration>
```

Подробная информация об encoder: https://github.com/logfellow/logstash-logback-encoder

Чтобы включить генерацию СЦЛ в секции **logging** файла настроек **application.yaml** установите значения `true`.

2.2.23.1.2 Установка и настройка Vector

Установка производится по официальной документации: https://vector.dev/docs/setup/installation/

Пример настройки source:

```
json_source:
    type: fluent
    address: 0.0.0.0:24226
```

Пример фильтрации сообщений, имеющих флаг `scl`:

```
scl_tags_filter:
type: filter
inputs:
  - json_source
condition:
  type: "vrl"
  source: |-
    exists(.tags) && includes(array!(.tags), "TYPE_SCL")
```

Пример парсинга scl-сообщений:

```
scl_message_remap:
    type: remap
    inputs:
    - scl_tags_filter
    source: |-
    . = parse_json!(.message)
```

Пример отправки scl-сообщений в Kafka:

```
podd_agent_sink:
    type: kafka
    inputs:
      - scl_message_remap
    bootstrap_servers: kafka:9092
    topic: "<префикс>.scl.signal"
    acknowledgements: true
    compression: "gzip"
    encoding:
      codec: json
    healthcheck: true
```

2.2.23.1.3 Установка и настройка HaProxy

Установка производится по официальной документации: http://docs.haproxy.org/

Для настройки HaProxy в секции `backend` перечислите список установленных инстансов **Vector**.

Пример файла `haproxy.cfg`:

```
global
    log 127.0.0.1 local2

    chroot /var/lib/haproxy
    pidfile /var/run/haproxy.pid
    maxconn 4000
    user haproxy
    group haproxy
    daemon

    stats socket /var/lib/haproxy/stats

defaults
    mode tcp
    log global
    retries 3

    maxconn 3000

listen stats
    bind                0.0.0.0:1936
    mode                http
    stats               enable
    stats               uri /

frontend services
    bind 0.0.0.0:24226
    default_backend services
    mode tcp

backend services
    balance roundrobin
    mode tcp
    server vector01 vector-01:24226
    server vector02 vector-02:24226
```

2.2.23.1.4 Установка и настройка Fluent bit

Установка производится по официальной документации: (https://docs.fluentbit.io/manual/installation/getting-started-with-fluent-bit).

Fluent bit должен быть настроен на чтение файлов логов приложений.

Пример файла конфигурации `fluent-bit.conf`:

```
[SERVICE]
    flush        5
    daemon       off
    log_level    info
    parsers_file parsers.conf
[INPUT]
    name tail
    path <путь до лог файла приложения>
    tag *
    parser json
[OUTPUT]
    name forward
    match *
    host haproxy
    port 24226
```

Пример файла `parsers.conf`:

```
[PARSER]
    Name        json
    Format      json
```

На этом настройка fluentbit завершена.

2.2.23.1.5 Работа с БД ClickHouse

В рамках технического решения по хранению протоколируемых запросов и ответов с возможностью извлечение данных по уникальному идентификатору реализовано использование колоночной аналитической базы данных **ClickHouse**.

Ключевые функциональные особенности базы данных **ClickHouse**:

- **движок базы данных**: по умолчанию **ClickHouse** использует движок Atomic;
- **движок таблиц**: MergeTree;
- **версия ClickHouse**: LTS;
- запрос на создание таблицы хранения логов в **ClickHouse**: **CREATE TABLE**.

Пример создания таблицы:

```
CREATE TABLE {Название БД}.logs
(
   logger String,
   timestamp DateTime,
   level String,
   requestId String,
   message String,
   messageType String,
   customerId String,
   customerOgrn String,
   queryMnemonic String
)
ENGINE = MergeTree()
PARTITION BY timestamp
ORDER BY timestamp
SETTINGS index_granularity = 8192;
```

Пример задания конфигурационных настроек:

```
clickhouse_default_config:
clickhouse:
   logger:
      level: trace
      log: /var/log/clickhouse-server/clickhouse-server.log
      errorlog: /var/log/clickhouse-server/clickhouse-server.err.log
      size: 1000M
      count: 10
   http_port: 8123
   tcp_port: 9000
   listen_host: 0.0.0.0
   max_connections: 4096
   keep_alive_timeout: 3
   user_directories:
      users_xml:
      path: users.xml
      local_directory:
      path: "{{ clickhouse_root_data_folder }}/access/"
   path: "{{ clickhouse_root_data_folder | add_slash }}"
```

2.2.23.1.6 Включение / выключение отправки сообщений в СЦЛ

Отправка логов в СЦЛ осуществляется автоматически после корректной настройки компонента.

Для выключения отправки логов закомментируйте блок отправки сообщений `podd_agent_sink` в Kafka в настройках Vector.

2.2.24 Настройка Агента СМЭВ4

Порядок установки и описание настроек Агента СМЭВ4 см. в документе: **«Руководство администратора СМЭВ4»**.

Описание формата взаимодействия между Агентом СМЭВ4 и СМЭВ4-адаптером (название топиков, формат сообщений, схема взаимодействия) описан в разделе Спецификация Модуля исполнения запросов.

2.2.24.1 Настройка взаимодействия программы с Агентом СМЭВ4

После установки программы и Агента СМЭВ4 настройте их взаимодействие между собой. Для этого:

1. Настройте Агента СМЭВ4 и СМЭВ4-адаптер на работу с одним и тем же брокером сообщения Kafka:
   - Если вместе с Агентом СМЭВ4 устанавливается брокер сообщений Kafka, а Агент СМЭВ4 преднастроен на работу именно с этим экземпляром брокера сообщений, то укажите адрес этого брокера сообщений в конфигурационном файле СМЭВ4-адаптера (`application.yml`), параметр `kafkaUrl`.
   - Если вместе с Агентом СМЭВ4 не устанавливается брокер сообщений Kafka, то в Агенте СМЭВ4 согласно его документации настройте работу с брокером сообщений Kafka, установленным с программой. Для этого используйте адрес сервера Kafka из конфигурационного файла СМЭВ4-адаптера (`application.yml`), параметр `kafkaUrl`.
2. Настройте названия топиков (см. Таблица 2.17) для обмена сообщениями в конфигурационном файле СМЭВ4-адаптера (`application.yml`).

Таблица 2.17 Название топиков для обмена сообщениями между СМЭВ4-адаптером и

Агентом СМЭВ4

№	Назначение	Настройка	Значение по умолчанию
1	Получение запросов	`client.kafka.query.consumer.rqTopicName`	`query.rq`
2	Ответы на запросы	`client.kafka.query.producer.rsTopicName`	`query.rs`
3	Ошибки запросов	`client.kafka.query.producer.errTopicName`	`query.err`
4	Результат запроса оценки	`client.kafka.query.estimateTopicName`	`query.query.estimation.rs`

Формат обмена электронными сообщениями с СМЭВ4-адаптером описан в разделе Спецификация Модуля исполнения запросов.

2.3 Настройка сервиса мониторинга

Для мониторинга состояния работы Компонента «Витрины данных» используется связка Grafana + Prometheus.

Prometheus — система мониторинга, обладающая возможностями тонкой настройки метрик. Prometheus используется для отслеживания состояния работы компонентов системы на низком уровне.

Grafana — инструмент с открытым исходным кодом для визуализации данных из различных систем сбора статистики. Grafana используется для представления в графическом виде временных рядов и текстовых данных.

Примечание:

Описание установки системы мониторинга приведено в разделе Установка системы мониторинга документа «Руководство по установке Компонента «Витрина данных»».

2.3.1 Предоставление источника данных

Существует два способа предоставления источника данных:

- с помощью конфигурационного файла;
- с помощью интерфейса Grafana.

Настройка в конфигурационном файле

Каждый файл конфигурации предоставления источника данных содержит *манифест*, в котором указывается желаемое состояние набора подготовленных источников данных.

При запуске Grafana загружает файлы конфигурации и предоставляет источники данных, перечисленные в манифестах.

Ниже приведен пример настройки источника данных **TestData** , который можно использовать для информационных панелей.

1. В директории `provisioning/datasources/` создайте файл `dtm.yml` со следующим содержимым:

```
apiVersion: 1

datasources:
  - name: Prometheus
    type: prometheus
    access: proxy
    url: <ip:port>
    jsonData:
      httpMethod: POST
      manageAlerts: false
      prometheusType: Prometheus
```

2. Перезапустите Grafana, чтобы загрузить новые изменения.
3. На боковой панели наведите курсор на значок « Конфигурация» (шестеренка) и нажмите «Источники данных». TestData появится в списке источников данных.

Примечание:

Папка `provisioning/datasources/` находится в `/etc/grafana/`.

Настройка в интерфейсе Grafana

Для работы Prometheus с Grafana необходимо добавить Prometheus в качестве источника получения данных в Grafana.

1. Откройте Grafana — Configuration — Data sources, нажмите **Add data source** и выберите **Prometheus**.

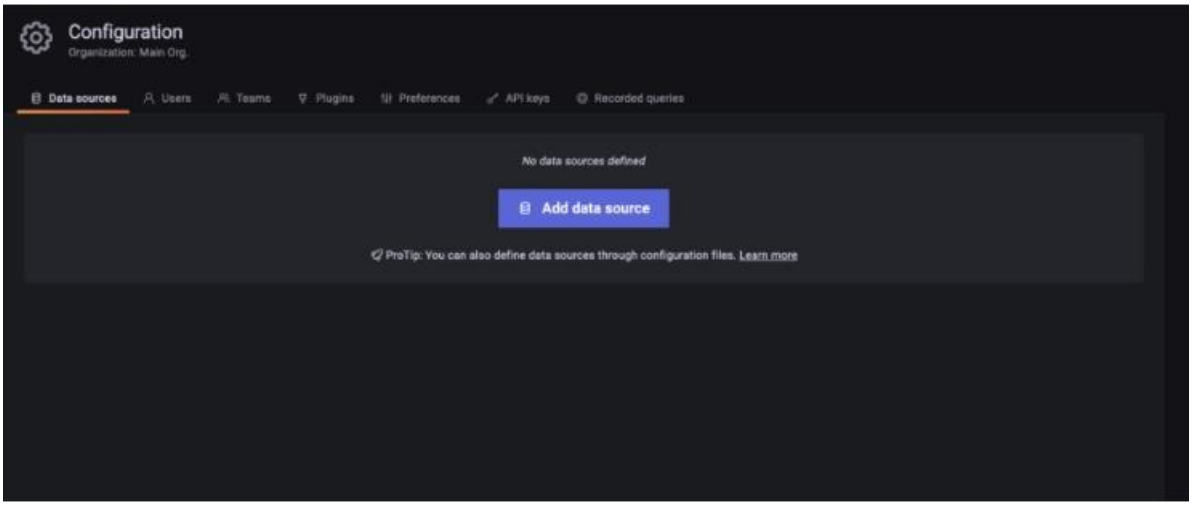

Рисунок - 2.13 Выбор Prometheus в качестве источника получения данных

2. В поле URL введите адрес и порт, по которому доступен Prometheus.

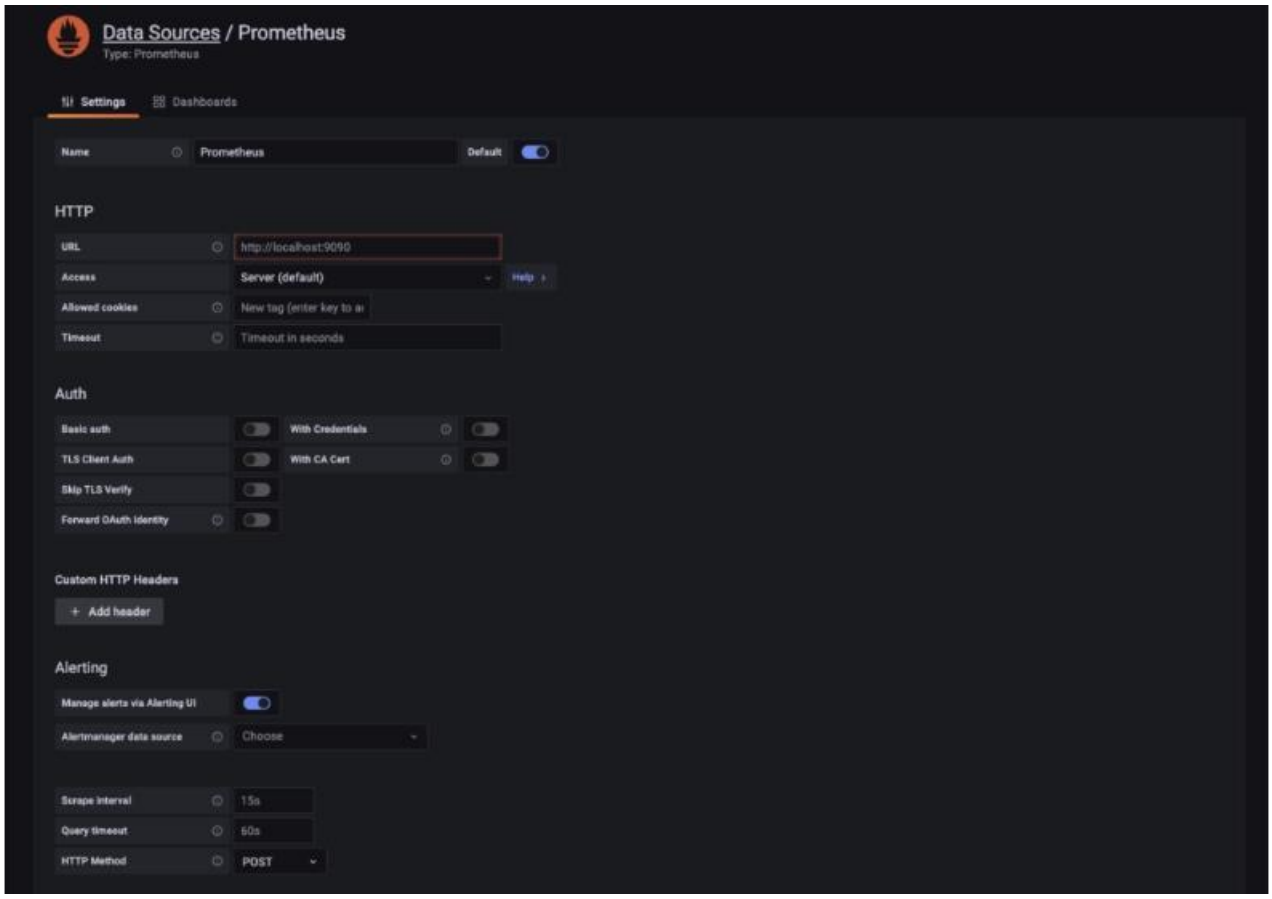

Рисунок - 2.14 Ввод URL Prometheus

3. Внизу страницы нажмите кнопку **Save & test**

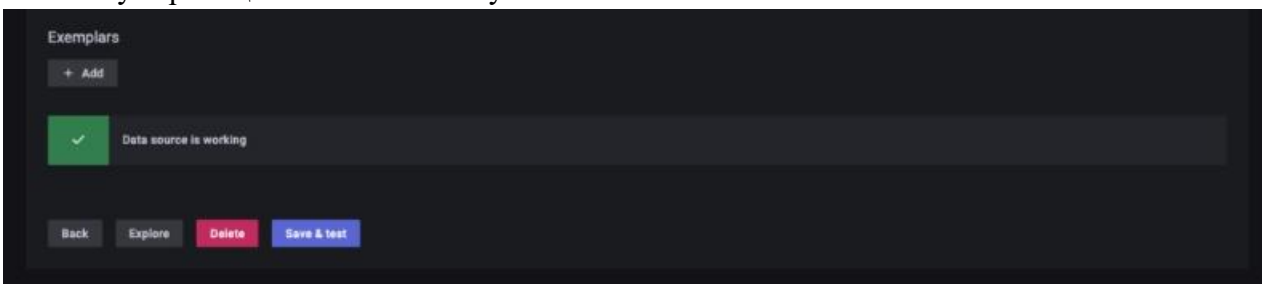

Рисунок - 2.15 Применение настроек

После успешной проверки Prometheus будет добавлен в Grafana.

2.3.2 Предоставление информационной панели

Определить поставщика информационных панелей можно также двумя способами:

- с помощью конфигурационного файла;
- с помощью интерфейса Grafana.

Настройка в конфигурационном файле

Каждый файл конфигурации информационной панели содержит *манифест*, который определяет желаемое состояние набора *поставщиков информационной панели*.

Поставщик информационной панели сообщает Grafana, где найти и где разместить определения информационной панели.

Grafana регулярно проверяет изменения в определениях панели мониторинга (по

умолчанию каждые 10 секунд).

В директории `provisioning/dashboards/` создайте файл `dtm.yml` со следующим содержимым:

```
apiVersion: 1

providers:
  - name: 'DTM Metrics' # Уникальное идентифицируемое имя поставщика
    folder: 'dtm-metrics' # Папка, в которую помещаются дашборды
    type: file
    disableDeletion: false
    updateIntervalSeconds: 10
    allowUiUpdates: false
    options:
      path: <path to dashboard definitions>
      foldersFromFilesStructure: false
```

> Примечание:
>
> Папка `provisioning/dashboards/` находится в `/etc/grafana/`.

Настройка в интерфейсе Grafana

1. Нажмите на иконку + и выберите «Import dashboard».

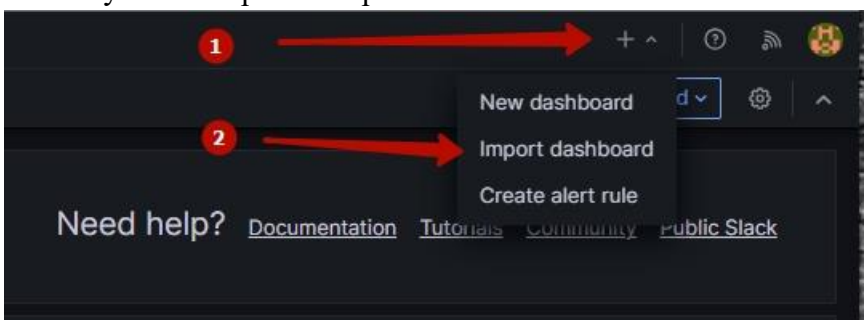

Рисунок - 2.16 Меню импорта Панели

2. В открывшемся окне нажмите на плашку «Upload dashboard JSON file» и загрузите файл нужной панели.

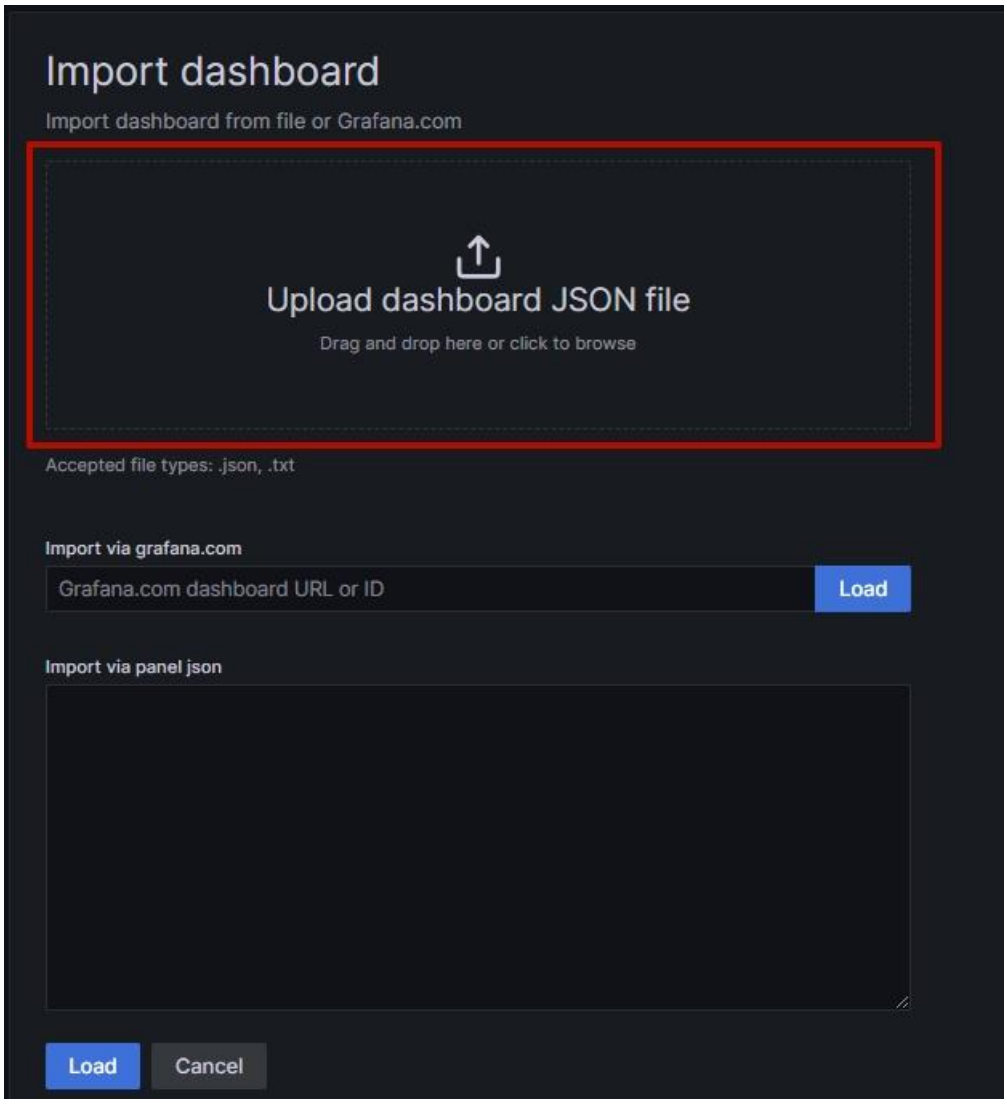

Рисунок - 2.17 Загрузка файла панели

2.3.2.1 Настройка конфигурационного файла Prometheus

Пример конфигурационного файла `prometheus.yml`:

```
global:
  scrape_timeout:     5s
  scrape_interval:    5s

scrape_configs:
  - job_name: 'smevql-server'
    static_configs:
      - targets: ['ip:8080']

  - job_name: 'blob-adapter'
    static_configs:
      # изменить стандартный порт в application.yml сервиса на кастомный, так как он совпадает с другими сервисами
      - targets: ['ip:9837']

  - job_name: 'counter-provider'
    static_configs:
      # изменить стандартный порт в application.yml сервиса на кастомный, так как он совпадает с другими сервисами
      - targets: ['ip:9837']
```

```
  - job_name: 'csv-uploader'
    static_configs:
      # изменить стандартный порт в application.yml сервиса на кастомный, так как
он совпадает с другими сервисами
      - targets: ['ip:9837']

  - job_name: 'data-uploader'
    static_configs:
      # изменить стандартный порт в application.yml сервиса на кастомный, так как
он совпадает с другими сервисами
      - targets: ['ip:9837']

  - job_name: 'podd-adapter-group-repl'
    static_configs:
      # изменить стандартный порт в application.yml сервиса на кастомный, так как
он совпадает с другими сервисами
      - targets: ['ip:9837']

  - job_name: 'podd-adapter-group-tp'
    static_configs:
      # изменить стандартный порт в application.yml сервиса на кастомный, так как
он совпадает с другими сервисами
      - targets: ['ip:9843']

  - job_name: 'podd-adapter-import-tp'
    static_configs:
      - targets: ['ip:19843']

  - job_name: 'podd-adapter-mppr'
    static_configs:
      # изменить стандартный порт в application.yml сервиса на кастомный, так как
он совпадает с другими сервисами
      - targets: ['ip:9843']

  - job_name: 'podd-adapter-mppw'
    static_configs:
      # изменить стандартный порт в application.yml сервиса на кастомный, так как
он совпадает с другими сервисами
      - targets: ['ip:9843']

  - job_name: 'podd-adapter-query'
    static_configs:
      # изменить стандартный порт в application.yml сервиса на кастомный, так как
он совпадает с другими сервисами
      - targets: ['ip:9837']

  - job_name: 'podd-adapter-replicator'
    static_configs:
      # изменить стандартный порт в application.yml сервиса на кастомный, так как
он совпадает с другими сервисами
      - targets: ['ip:9837']

  - job_name: 'podd-avro-defragmentator'
    static_configs:
      # изменить стандартный порт в application.yml сервиса на кастомный, так как
он совпадает с другими сервисами
      - targets: ['ip:9837']

  - job_name: 'printable-form-service'
    static_configs:
      # изменить стандартный порт в application.yml сервиса на кастомный, так как
```

```
он совпадает с другими сервисами
      - targets: ['ip:9837']

  - job_name: 'rest-uploader'
    static_configs:
      # изменить стандартный порт в application.yml сервиса на кастомный, так как
он совпадает с другими сервисами
      - targets: ['ip:9837']

  - job_name: 'smev3-adapter'
    static_configs:
      # изменить стандартный порт в application.yml сервиса на кастомный, так как
он совпадает с другими сервисами
      - targets: ['ip:9033']
```

3 ЗАПУСК И ОСТАНОВКА ПРОГРАММЫ

Программа не имеет графического интерфейса и запускается автоматически после запуска сервера.

Все компоненты Программы оформлены в виде системных служб, имеют отдельные файлы конфигурации, автоматически запускаются при старте сервера и автоматически останавливаются при его выключении.

При необходимости любой из сервисов/модулей можно остановить и перезапустить.

Данный раздел содержит описание запуска и остановки модулей ручным способом.

Внимание:
Программные средства настраиваются в зависимости от используемой конфигурации. Состав компонентов приведен в разделе Состав компонентов в дистрибутиве документа «Техническое описание системы».

3.1 Prostore

3.1.1 Запуск

Процесс запуска Prostore приведен в документации сервиса: https://prostore.datamart.ru/docs_prostore/maintenance/maintenance.html.

3.2 СМЭВ QL Сервер

Описание настроек модуля приведено в разделе Настройка программных средств.

3.2.1 Быстрый старт

3.2.1.1 Создание и конфигурация

Создать новое приложение СМЭВ QL Сервера командой:

```
java -jar smevql-server-all.jar new <new-app-name>
```

Данная команда создаст структуру папок сервера и исполняемый файл `smevql` внутри `<new-app-name>`.

3.2.1.2 Запуск и управление

Запуск СМЭВ QL Сервера осуществляется командой:

```
./smevql start -e <environment>
```

- `environment` - задается название среды разработки. Без указания окружения сервер будет запущен в `development`.

В момент запуска приложения выполняются проверки наличия и корректности заполнения конфигурационного файла и файлов моделей.

Типовые ошибки представлены ниже:

Тип ошибки: Некорректный формат или отсутствие файла модели

Пример лог-записи:

{«@timestamp»:»2024-01-10T16:25:55.460659+03:00»,»@version»:»1»,»message»:»Ошибка старта сервера»,»logger_name»:»ru.gov.digital.smevql.server.RequestHandler»,»thread_name»:»main»,«level»:»ERROR»,»level_value»:40000,»stack_trace»:»java.lang.RuntimeException: Ошибка парсинга модели данных из файла

Тип ошибки: Некорректно заполнен конфигурационный файл

Пример лог-записи:

{«@timestamp»:»2024-01-10T16:27:01.202248+03:00»,»@version»:»1»,»message»:»Ошибка старта сервера»,»logger_name»:»ru.gov.digital.smevql.server.RequestHandler»,»thread_name»:»main»,«level»:»ERROR»,»level_value»:40000,»stack_trace»:»net.mamoe.yamlkt.YamlDecodingException:Top-level decoder: Yaml Block Map: bad indentation, baseIndent=0, newIndent=2n uri: smevql/api/v1

Остановка СМЭВ QL Сервера осуществляется командой:

```
./smevql stop
```

Перезапуск СМЭВ QL Сервера осуществляется командой:

```
./smevql restart
```

3.2.1.3 Работа с сервером

3.2.1.3.1 Генераторы

Генераторы создают папки и файлы-шаблоны с начальными значениями. Для запуска генератора можно использовать полную команду `./smevql generate` или короткий алиас `./smevql g`.

Новый пустой источник генерируется командой:

```
./smevql g source <source-name>
```

Пример источника на основе Prostore:

```
prostore_source:
type: rest
version: '1.0'
adapter: prostore
protocol: http
host: smevql-dtm-prostore01.ru-central1.internal
port: 9090
path: api/v1/datamarts/query?format=json
headers:
    - content-type: application/json
threads-count: 4
connection-timeout: 30
```

Новая модель генерируется командой:

```
./smevql g model <model-name>
```

Пример модели:

```
resources:
- mo: *base_model
    name: Медицинская организация
    description: Логическая таблица "Медицинская организация"
    fields:
    <<: *default_fields
    parent_id:
        <<: *ds
        name: parent_id
    update_ts:
        <<: *dts
        name: update_ts
    address:
        <<: *ds
        name: address
```

```
    address_fias_guid:
        <<: *ds
        name: address_fias_guid
    enabled:
        <<: *ds
        name: enabled
    name:
        <<: *ds
        name: name
    region_okato:
        <<: *ds
        name: region_okato
    create_ts:
        <<: *dts
        name: create_ts
    id:
        <<: *pks
        name: id
    rmis_id:
        <<: *ds
        name: rmis_id
    phone:
        <<: *ds
        name: phone
    connections:
    has_many: []
    belongs_to:
    - attachment:
        primary_key: [ mo_id ]
        foreign_key: [ id ]
    - resource:
        primary_key: [ mo_id ]
        foreign_key: [ id ]
    extract:
    source:
        - name: prostore
        table: misdm02.mo
- profilecode_resource: *base_model
- resource: *base_model
- observation: *base_model
- book: *base_model
- slot: *base_model
- monitoring: *base_model
- referral: *base_model
- attachment: *base_model
- patient: *base_model
- service: *base_model
- unaccessible_period: *base_model
```

Из существующего Prostore модель генерируется командой:

```
./smevql schema-gen test -h localhost -p 9090 -d demo_view
```

- `test` - имя директории, куда будет выгружена модель;
- `-h localhost -p 9090` - это хост и порт Prostore;
- `-d demo_view` - витрина (схема).

3.2.1.4 Сборка проекта

Собрать проект можно с помощью gradle:

```
./gradlew clean build
```

3.3 СМЭВ3-адаптер

3.3.1 Запуск модуля

Модуль может быть поставлен как контейнер, управляемый Docker или как JAR-файл.

Запуск из Docker

Запуск выполняется при помощи **Docker** команды:

```
docker start smev3-adapter
```

Запуск как JAR-file

Если модуль поставляется как JAR-файл, вводится команда

```
java
    [-Dconfig.location=<путь до application.yml> ]
    [-Dlogging.config=logback.xml]
    -jar <путь до smev3-adapter.jar>
```

команды заключенные в [] выполняются опционально.

3.3.2 Остановка модуля

Остановка модуля выполняется при помощи **Docker** команды:

```
docker stop smev3-adapter
```

Для ручной остановки необходимо подключиться по SSH на сервер, найти процесс, который содержит JAR-файл и остановить его.

Пример:

```
ps aux | grep smev3-adapter
```

3.4 CSV-Uploader

Описание настроек модуля приведено в разделе Настройка программных средств.

3.4.1 Запуск CSV-uploader

Модуль может быть поставлен как контейнер, управляемый Docker или как JAR-файл.

Запуск из Docker

Запуск выполняется при помощи **Docker** команды:

```
docker start csv-uploader
```

Запуск как JAR-file

Если модуль поставляется как JAR-файл, вводится команда

```
java
    [-Dconfig.location=<путь до application.yml> ]
    [-Dlogging.config=logback.xml]
    -jar <путь до csv-uploader.jar>
```

команды заключенные в [] выполняются опционально.

3.4.2 Остановка модуля

Остановка модуля выполняется при помощи **Docker** команды:

```
docker stop csv-uploader
```

Для ручной остановки необходимо подключиться по SSH на сервер, найти процесс, который содержит JAR-файл и остановить его.

Пример:

```
ps aux | grep csv-uploader
```

3.5 СМЭВ4-адаптер - Модуль исполнения запросов

Описание настроек модуля приведено в разделе Настройка программных средств.

3.5.1 Запуск модуля

Модуль может быть поставлен как контейнер, управляемый Docker или как JAR-файл.

Запуск из Docker

Запуск выполняется при помощи **Docker** команды:

```
docker start podd-adapter-query
```

Запуск как JAR-file

Если модуль поставляется как JAR-файл, вводится команда

```
java
    [-Dconfig.location=<путь до application.yml> ]
    [-Dlogging.config=logback.xml]
    -jar <путь до podd-adapter-query.jar>
```

команды заключенные в [] выполняются опционально.

3.5.2 Остановка модуля

Остановка модуля выполняется при помощи **Docker** команды:

```
docker stop podd-adapter-query
```

Для ручной остановки необходимо подключиться по SSH на сервер, найти процесс, который содержит JAR-файл и остановить его.

Пример:

```
ps aux | grep podd-adapter-query
```

3.6 СМЭВ4-адаптер – Модуль MPPR

Описание настроек модуля приведено в разделе Настройка программных средств.

3.6.1 Запуск модуля

Модуль может быть поставлен как контейнер, управляемый Docker или как JAR-файл.

Запуск из Docker

Запуск выполняется при помощи **Docker** команды:

```
docker start podd-adapter-mppr
```

Запуск как JAR-file

Если модуль поставляется как JAR-файл, вводится команда

```
java
    [-Dconfig.location=<путь до application.yml> ]
    [-Dlogging.config=logback.xml]
    -jar <путь до podd-adapter-mppr.jar>
```

команды заключенные в [] выполняются опционально.

3.6.2 Остановка модуля

Остановка модуля выполняется при помощи **Docker** команды:

```
docker stop podd-adapter-mppr
```

Для ручной остановки необходимо подключиться по SSH на сервер, найти процесс, который содержит JAR-файл и остановить его.

Пример:

```
ps aux | grep podd-adapter-mppr
```

3.7 СМЭВ4-адаптер-Модуль MPPW

Описание настроек модуля приведено в разделе Настройка программных средств.

3.7.1 Запуск модуля

Модуль может быть поставлен как контейнер, управляемый Docker или как JAR-файл.

Запуск из Docker

Запуск выполняется при помощи **Docker** команды:

```
docker start podd-adapter-mppw
```

Запуск как JAR-file

Если модуль поставляется как JAR-файл, вводится команда

```
java
    [-Dconfig.location=<путь до application.yml> ]
    [-Dlogging.config=logback.xml]
    -jar <путь до podd-adapter-mppw.jar>
```

команды заключенные в [] выполняются опционально.

3.7.2 Остановка модуля

Остановка модуля выполняется при помощи **Docker** команды:

```
docker stop podd-adapter-mppw
```

Для ручной остановки необходимо подключиться по SSH на сервер, найти процесс, который содержит JAR-файл и остановить его.

Пример:

```
ps aux | grep  podd-adapter-mppw
```

3.8 СМЭВ4-адаптер – Модуль импорта данных табличных параметров

Описание настроек модуля приведено в разделе Настройка программных средств.

3.8.1 Запуск модуля

Модуль может быть поставлен как контейнер, управляемый Docker или как JAR-файл.

Запуск из Docker

Запуск выполняется при помощи **Docker** команды:

```
docker start podd-adapter-import-tp
```

Запуск как JAR-file

Если модуль поставляется как JAR-файл, вводится команда

```
java
    [-Dconfig.location=<путь до application.yml> ]
    [-Dlogging.config=logback.xml]
    -jar <путь до podd-adapter-import-tp.jar>
```

команды заключенные в [] выполняются опционально.

3.8.2 Остановка модуля

Остановка модуля выполняется при помощи **Docker** команды:

```
docker stop podd-adapter-import-tp
```

Для ручной остановки необходимо подключиться по SSH на сервер, найти процесс, который содержит JAR-файл и остановить его.

Пример:

```
ps aux | grep podd-adapter-import-tp
```

3.9 СМЭВ4-адаптер – Модуль группировки данных табличных параметров

Описание настроек модуля приведено в разделе Настройка программных средств.

3.9.1 Запуск модуля

Модуль может быть поставлен как контейнер, управляемый Docker или как JAR-файл.

Запуск из Docker

Запуск выполняется при помощи **Docker** команды:

```
docker start podd-adapter-group-tp
```

Запуск как JAR-file

Если модуль поставляется как JAR-файл, вводится команда

```
java
    [-Dconfig.location=<путь до application.yml> ]
    [-Dlogging.config=logback.xml]
    -jar <путь до podd-adapter-group-tp.jar>
```

команды заключенные в [] выполняются опционально.

3.9.2 Остановка модуля

Остановка модуля выполняется при помощи **Docker** команды:

```
docker stop podd-adapter-group-tp
```

Для ручной остановки необходимо подключиться по SSH на сервер, найти процесс, который содержит JAR-файл и остановить его.

Пример:

```
ps aux | grep podd-adapter-group-tp
```

3.10 СМЭВ4-адаптер – Модуль дефрагментации чанков табличных параметров

Описание настроек модуля приведено в разделе Настройка программных средств.

3.10.1 Запуск модуля

Модуль может быть поставлен как контейнер, управляемый Docker или как JAR-файл.

Запуск из Docker

Запуск выполняется при помощи **Docker** команды:

```
docker start podd-adapter-defragmentator
```

Запуск как JAR-file

Если модуль поставляется как JAR-файл, вводится команда

```
java
    [-Dconfig.location=<путь до application.yml> ]
    [-Dlogging.config=logback.xml]
    -jar <путь до podd-adapter-defragmentator.jar>
```

команды заключенные в [] выполняются опционально.

3.10.2 Остановка модуля

Остановка модуля выполняется при помощи **Docker** команды:

```
docker stop podd-adapter-defragmentator
```

Для ручной остановки необходимо подключиться по SSH на сервер, найти процесс, который содержит JAR-файл и остановить его.

Пример:

```
ps aux | grep podd-adapter-defragmentator
```

3.11 DATA-uploader – Модуль исполнения асинхронных заданий

Описание настроек модуля приведено в разделе Настройка программных средств.

3.11.1 Запуск модуля

Модуль может быть поставлен как контейнер, управляемый Docker или как JAR-файл.

Запуск из Docker

Запуск выполняется при помощи **Docker** команды:

```
docker start data-uploader
```

Запуск как JAR-file

Если модуль поставляется как JAR-файл, вводится команда

```
java
    [-Dconfig.location=<путь до application.yml> ]
    [-Dlogging.config=logback.xml]
    -jar <путь до data-uploader.jar>
```

команды заключенные в [] выполняются опционально.

3.11.2 Остановка модуля

Остановка модуля выполняется при помощи **Docker** команды:

```
docker stop data-uploader
```

Для ручной остановки необходимо подключиться по SSH на сервер, найти процесс, который содержит JAR-файл и остановить его.

Пример:

```
ps aux | grep data-uploader
```

3.12 REST-uploader – Модуль асинхронной загрузки данных из сторонних источников

Описание настроек модуля приведено в разделе Настройка программных средств.

3.12.1 Запуск модуля

Модуль может быть поставлен как контейнер, управляемый Docker или как JAR-файл.

Запуск из Docker

Запуск выполняется при помощи **Docker** команды:

```
docker start rest-uploader
```

Запуск как JAR-file

Если модуль поставляется как JAR-файл, вводится команда

```
java
    [-Dconfig.location=<путь до application.yml> ]
    [-Dlogging.config=logback.xml]
    -jar <путь до rest-uploader.jar>
```

команды заключенные в [] выполняются опционально.

3.12.2 Остановка модуля

Остановка модуля выполняется при помощи Docker команды:

```
docker stop rest-uploader
```

Для ручной остановки необходимо подключиться по SSH на сервер, найти процесс, который содержит JAR-файл и остановить его.

Пример:

```
ps aux | grep rest-uploader
```

3.12.3 Добавление поставщика данных

Для добавления поставщика данных должен генерироваться токен авторизации, который передается поставщику.

Генерация токена осуществляется по следующим шагам:

1. Открыть web-страницу https://jwt.io/
2. Выбрать алгоритм HS256;
3. Ввести в payload следующие поля:

```
{
"sub": "1234567890",
"iss": "John Doe"
}
```

где:

– sub - идентификатор поставщика данных, для которого сформирован токен;
– iss - кем сформирован токен.

Подпись токена формируется методом получения хеш-функции SHA-256 с секретом. Для этого нужно в `verify signature` в поле `your-256-bit-secret` ввести значение из `test-secret` настроек сервиса REST-uploader.

Для добавления идентификатора поставщика данных в Базу данных Redis необходимо в структуре `set`, содержащую идентификаторы поставщика данных, выполнить операцию SADD:

```
SADD ids ProviderID
```

где:

– ids - ключ, по которому осуществляется доступ к набору элементов;
– ProviderID - идентификатор поставщика данных.

В случае, когда ожидание ответа на запрос превысило указанное количество времени, необходимо сделать повторный запрос.

В случае возникновения ошибок при обработке файлов сотрудникам, загружаюмщим данные необходимо изучить возврат REST-uploader. Если ошибка внутренняя, то нужно обратиться к администратору Витрины. Администратор изучит логи REST-uploader / Data-uploader.

3.13 СМЭВ4-адаптер-Модуль подписки

Описание настроек модуля приведено в «Руководстве администратора».

3.13.1 Запуск модуля

Модуль может быть поставлен как контейнер, управляемый Docker или как JAR-файл.

Запуск из Docker

Запуск выполняется при помощи **Docker** команды:

```
docker start podd-adapter-replicator
```

Запуск как JAR-file

Если модуль поставляется как JAR-файл, вводится команда

```
java
    [-Dconfig.location=<путь до application.yml> ]
    [-Dlogging.config=logback.xml]
    -jar <путь до podd-adapter-replicator.jar>
```

команды заключенные в [] выполняются опционально.

3.13.2 Остановка модуля

Остановка модуля выполняется при помощи Docker команды:

```
docker stop podd-adapter-replicator
```

Для ручной остановки необходимо подключиться по SSH на сервер, найти процесс, который содержит JAR-файл и остановить его.

Пример:

```
ps aux | grep  podd-adapter-replicator
```

3.14 BLOB-адаптер

Описание настроек модуля приведено в «Руководстве администратора».

3.14.1 Запуск модуля

Для ручного запуска необходимо подключиться по SSH на сервер и в командной строке запустить jar-файл, указав его расположение.

Например:

```
java
    -Dconfig.location=<путь до application.yml>
    -jar <путь до blob-adapter.jar> &
```

– `Dconfig.location` – путь до конфигурационного файла модуля (application.yml).

3.14.2 Остановка модуля

Для ручной остановки необходимо подключиться по ssh на сервер, найти процесс, который содержит jar-файл и остановить его.

Пример:

```
ps aux | grep   blob-adapter
```

`kill` «номер процесса».

3.15 Сервис формирования документов

Описание настроек модуля приведено в «Руководстве администратора».

3.15.1 Запуск модуля

Модуль может быть поставлен как контейнер, управляемый Docker или как JAR-файл.

Запуск из Docker

Запуск выполняется при помощи **Docker** команды:

```
docker start printable-form-service
```

Запуск как JAR-file

Если модуль поставляется как JAR-файл, вводится команда

```
java
    [-Dconfig.location=<путь до application.yml> ]
    [-Dlogging.config=logback.xml]
    -jar <путь до printable-form-service.jar>
```

команды заключенные в `[]` выполняются опционально.

3.15.2 Остановка модуля

Остановка модуля выполняется при помощи Docker команды:

```
docker stop printable-form-service
```

Для ручной остановки необходимо подключиться по ssh на сервер, найти процесс, который содержит jar-файл и остановить его.

Пример:

```
ps aux | grep  printable-form-service
```

3.16 Утилита Backup manager

Описание настроек утилиты приведено в разделе Бекапирование Витрины данных НСУД.

3.16.1 Запуск утилиты

Для запуска механизма резервного копирования, администратору системы нужно перейти в директорию backup-tools (утилиты **Backup Manager** и **DTM-Tools** должны быть расположены в одной директории) и выполнить в консоли команду `backup`:

```
java -jar backup-manager-1.10.0-SNAPSHOT.jar backup --dtmTools ./dtm-tools-1.12.0.jar --kafkaBrokers 10.81.7.90:12541 --prostore 10.81.7.90:12520
```

Backup Manager принимает команду в качестве аргумента, обеспечивает выполнение 2 операций:

- **backup** - резервное копирование, вторым аргументом указывается целевой архив;
- **restore** - восстановление из резервной копии.

В процессе бекапирования или восстановления из бекапа утилита **Backup Manager** выводит в консоли сообщения об основных шагах и выполняемых операциях вида:

- отправлена команда остановки обработки запросов;
- отправлена команда бекапирования датамарта (datamart).

3.17 ETL

3.17.1 Apache Airflow

Apache Airflow представляет собой набор контейнеров, управляемых **Docker**.

Ниже приведено описание запуска и остановки **Apache Airflow**.

3.17.1.1 Запуск

Для запуска **Apache Airflow** нужно перейти в директорию с файлом `docker-compose.yml`, созданным при установке **Apache Airflow**.

Например:

```
cd ~/<folder-name>
```

Выполните команду:

```
docker-compose start
```

3.17.1.2 Остановка

Для остановки **Apache Airflow** нужно перейти в директорию с файлом `docker-compose.yml`, созданным при установке **Apache Airflow**.

В папке, где расположен файл `docker-compose.yaml` выполните команду:

```
docker-compose stop
```

3.17.2 Apache Spark

Apache Spark представляет собой контейнер, управляемый **Docker**.
Ниже приведено описание запуска и остановки **Apache Spark**.

3.17.2.1 Запуск

Для запуска **Apache Spark** нужно перейти в директорию с файлом `docker-compose.yml`, созданным при установке **Apache Spark**.

Например:

```
cd ~/<folder-name>
```

Выполните команду:

```
docker-compose start
```

Для запуска отдельно мастера и воркера **Apache Spark** можно использовать команды **Docker**:

Пример команды:

```
docker start spark-master
docker start spark-worker-1
```

3.17.2.2 Остановка

Для остановки **Apache Spark** нужно перейти в директорию с файлом `docker-compose.yml`, созданным при установке **Apache Spark**.

В папке, где расположен файл `docker-compose.yaml` выполните команду:

```
docker-compose stop
```

Для остановки отдельно мастера и воркера Apache Spark можно использовать команды **Docker**:

Пример команды:

```
docker stop spark-master
docker stop spark-worker-1
```

3.17.3 Apache Hadoop

Apache Hadoop представляет собой набор контейнеров, управляемых **Docker**.
Ниже приведено описание запуска и остановки **Apache Hadoop**.

3.17.3.1 Запуск

Для запуска **Apache Hadoop** нужно перейти в директорию с файлом `docker-compose.yml`, созданным при установке **Apache Hadoop**.

Например:

```
cd ~/<folder-name>
```

Выполните команду:

```
docker-compose start
```

Для запуска отдельно каждого контейнера **Apache Hadoop** можно использовать

команды **Docker**:

Пример команды:

```
docker start namenode
docker start datanode
docker start resourcemanager
docker start nodemanager
docker start historyserver
```

3.17.3.2 Остановка

Для остановки **Apache Hadoop** нужно перейти в директорию с файлом `docker-compose.yml`, созданным при установке **Apache Hadoop**.

В папке, где расположен файл `docker-compose.yaml` выполните команду:

```
docker-compose stop
```

Для остановки отдельно каждого контейнера Apache Hadoop можно использовать команды **Docker**:

Пример команды:

```
docker stop namenode
docker stop datanode
docker stop resourcemanager
docker stop nodemanager
docker stop historyserver
```

3.17.4 Tarantool (Vynil)

Tarantool (Vynil) представляет собой контейнер, управляемый **Docker**.

Описание запуска и остановки Tarantool (Vynil) приведено в файле `docker-compose.yml` директории *Tarantool*.

3.17.4.1 Запуск

Для запуска Tarantool (Vynil) перейдите в директорию с файлом `docker-compose.yml`, созданным при установке **Tarantool (Vynil)**.

Например:

```
cd ~/direct
```

Выполните команду:

```
docker-compose start
```

Для запуска отдельно каждого контейнера **Tarantool (Vynil)** можно использовать команды **Docker**:

Пример команды:

```
docker start tarantool1
docker start tarantool2
```

3.17.4.2 Остановка

Для остановки **Tarantool (Vynil)** перейдите в директорию с файлом `docker-compose.yml`, созданным при установке **Tarantool (Vynil)**.

В папке, где расположен файл `docker-compose.yaml` выполните команду:

```
docker-compose stop
```

Для остановки отдельно каждого контейнера **Tarantool (Vynil)** можно использовать

команды **Docker**:

Пример команды:

```
docker stop tarantool1
docker stop tarantool2
```

3.18 REST-адаптер

REST-адаптер представляет собой контейнер, управляемый **Docker**. Для запуска и остановки **REST-адаптер** используются команды **Docker**.

Запуск REST-адаптер выполняется путём запуска **Docker** командой:

```
docker start rest-adapter
```

Остановка REST-адаптер выполняется путём остановки **Docker** командой:

```
docker stop rest-adapter
```

3.19 Counter-provider - Сервис генерации уникального номера

Описание настроек модуля приведено в Настройка программных средств.

3.19.1 Запуск модуля

Модуль может быть поставлен как контейнер, управляемый Docker или как JAR-файл.

Запуск из Docker

Запуск выполняется при помощи **Docker** команды:

```
docker start counter-provider
```

Запуск как JAR-file

Если модуль поставляется как JAR-файл, вводится команда

```
java
    [-Dconfig.location=<путь до application.yml> ]
    [-Dlogging.config=logback.xml]
    -jar <путь до counter-provider.jar>
```

команды заключенные в [] выполняются опционально.

3.19.2 Остановка модуля

Остановка модуля выполняется при помощи **Docker** команды:

```
docker stop counter-provider
```

Для ручной остановки необходимо подключиться по SSH на сервер, найти процесс, который содержит JAR-файл и остановить его.

Пример:

```
ps aux | grep counter-provider
```

3.20 Установка коннектора Kafka-Postgres

1. Скопировать файлы `kafka-postgres-writer-0.3.0.jar`, `kafka-postgres-reader-0.3.0.jar`, `kafka-postgres-avro-0.3.0.jar` из дистрибутива и загрузить в папку **/kafka-postgres-connector**.
2. Скопировать конфигурационные файлы ``application.yaml``сервисов KAFKA-POSTGRES-WRITER и KAFKA-POSTGRES-READER в папку **kafka-postgres-**

connector/config

Конфигурационный файл KAFKA-POSTGRES-WRITER `application.yaml`:

```
logging:
level:
    ru.datamart.kafka: ${LOG_LEVEL:DEBUG}
    org.apache.kafka: ${KAFKA_LOG_LEVEL:INFO}

http:
port: ${SERVER_PORT:8096}

vertx:
pools:
    eventLoopPoolSize: ${VERTX_EVENT_LOOP_SIZE:12}
    workersPoolSize: ${VERTX_WORKERS_POOL_SIZE:32}
verticle:
    query:
    instances: ${QUERY_VERTICLE_INSTANCES:12}
    insert:
    poolSize: ${INSERT_WORKER_POOL_SIZE:32}
    insertPeriodMs: ${INSERT_PERIOD_MS:1000}
    batchSize: ${INSERT_BATCH_SIZE:500}
    consumer:
    poolSize: ${KAFKA_CONSUMER_WORKER_POOL_SIZE:32}
    maxFetchSize: ${KAFKA_CONSUMER_MAX_FETCH_SIZE:10000}
    commit:
    poolSize: ${KAFKA_COMMIT_WORKER_POOL_SIZE:1}
    commitPeriodMs: ${KAFKA_COMMIT_WORKER_COMMIT_PERIOD_MS:1000}

client:
kafka:
    consumer:
    checkingTimeoutMs: ${KAFKA_CHECKING_TIMEOUT_MS:10000}
    responseTimeoutMs: ${KAFKA_RESPONSE_TIMEOUT_MS:10000}
    consumerSize: ${KAFKA_CONSUMER_SIZE:10}
    closeConsumersTimeout: ${KAFKA_CLOSE_CONSUMER_TIMEOUT:15000}
    property:
        bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:kafka.host:9092}
        group.id: ${KAFKA_CONSUMER_GROUP_ID:postgres-query-execution}
        auto.offset.reset: ${KAFKA_AUTO_OFFSET_RESET:earliest}
        enable.auto.commit: ${KAFKA_AUTO_COMMIT:false}
        auto.commit.interval.ms: ${KAFKA_AUTO_INTERVAL_MS:1000}

datasource:
postgres:
    database: ${POSTGRES_DB_NAME:test}
    user: ${POSTGRES_USERNAME:dtm}
    password: ${POSTGRES_PASS:dtm}
    hosts: ${POSTGRES_HOSTS:localhost:5432}
    poolSize: ${POSTGRES_POOLSIZE:10}
    preparedStatementsCacheMaxSize: ${POSTGRES_CACHE_MAX_SIZE:256}
    preparedStatementsCacheSqlLimit: ${POSTGRES_CACHE_SQL_LIMIT:2048}
    preparedStatementsCache: ${POSTGRES_CACHE:true}
```

Конфигурационный файл KAFKA-POSTGRES-READER `application.yaml`:

```
logging:
level:
    ru.datamart.kafka: ${LOG_LEVEL:DEBUG}
    org.apache.kafka: ${KAFKA_LOG_LEVEL:INFO}

http:
```

```
port: ${SERVER_PORT:8094}

vertx:
pools:
    eventLoopPoolSize: ${VERTX_EVENT_LOOP_SIZE:12}
    workersPoolSize: ${VERTX_WORKERS_POOL_SIZE:32}
verticle:
    query:
    instances: ${QUERY_VERTICLE_INSTANCES:12}

datasource:
postgres:
    database: ${POSTGRES_DB_NAME:test}
    user: ${POSTGRES_USERNAME:dtm}
    password: ${POSTGRES_PASS:dtm}
    hosts: ${POSTGRES_HOSTS:localhost:5432}
    poolSize: ${POSTGRES_POOLSIZE:10}
    preparedStatementsCacheMaxSize: ${POSTGRES_CACHE_MAX_SIZE:256}
    preparedStatementsCacheSqlLimit: ${POSTGRES_CACHE_SQL_LIMIT:2048}
    preparedStatementsCache: ${POSTGRES_CACHE:true}
    fetchSize: ${POSTGRES_FETCH_SIZE:1000}

kafka:
client:
    property:
    key.serializer: org.apache.kafka.common.serialization.ByteArraySerializer
    value.serializer: org.apache.kafka.common.serialization.ByteArraySerializer
```

3.21 Arenadata Cluster Manager (ADCM)

3.21.1 Запуск

ADCM представляет собой контейнер, управляемый **Docker**.

Для запуска **ADCM** выполните следующие команды:

1. Запустите **ADCM** введя команду **Docker**:

```
docker start adcm
```

2. Подключитесь через браузер к веб-интерфейсу по адресу

```
http://<ip_adress_of_server>:8000.
```

3. Авторизуйтесь в веб-интерфейсе.

3.21.2 Остановка

Остановка **ADCM** выполняется путём остановки **Docker** командой:

```
docker stop adcm
```

3.22 Arenadata Streaming (ADS)

Компонент запускается автоматически при установке как два `systemd` сервиса:

- *Kafka*;
- *Zookeeper*.

Включен автозапуск сервисов при перезапуске сервера.

Запуск и остановка ADS возможна как в консоли (см Раздел 3.22.3), так и через ADCM (см Раздел 3.22.4).

3.22.1 Kafka

3.22.1.1 Запуск

1. Авторизуйтесь в **Arenadata Cluster Manager (ADCM)**.
2. Перейдите в пункт кластер ADS.
3. В левом меню, выберите пункт **Services**.
4. В таблице со списком сервисов *ADS*, в строке *Kafka*, нажмите значок **Actions** и нажмите кнопку **Start** (см. Рисунок - 3.1).

3.22.1.2 Остановка

1. Авторизуйтесь в **Arenadata Cluster Manager (ADCM)**.
2. Перейдите в пункт кластер ADS.
3. В левом меню, выберите пункт **Services**.
4. В таблице со списком сервисов *ADS*, в строке *Kafka*, нажмите значок **Actions** и нажмите кнопку **Stop** (см. Рисунок - 3.1).

3.22.1.3 Перезапуск

Для перезапуска **Kafka** в **Arenadata Cluster Manager (ADCM)**, выполните следующие действия:

1. Авторизуйтесь в **Arenadata Cluster Manager (ADCM)**.
2. Перейдите в пункт кластер ADS.
3. В левом меню, выберите пункт **Services**.
4. В таблице со списком сервисов *ADS*, в строке *Kafka*, нажмите значок **Actions** и нажмите кнопку **Restart** (см. Рисунок - 3.1).

3.22.2 Zookeeper

3.22.2.1 Запуск

1. Авторизуйтесь в **Arenadata Cluster Manager (ADCM)**.
2. Нажмите вкладку **Cluster**.
3. Перейдите в пункт кластер ADS.
4. В левом меню, выберите пункт **Services**.
5. В таблице со списком сервисов *ADS*, в строке *Zookeeper*, нажмите значок **Actions** и нажмите кнопку **Start** (см. Рисунок - 3.1).
6. Подтвердите действие, для этого нажмите кнопку **Run**.

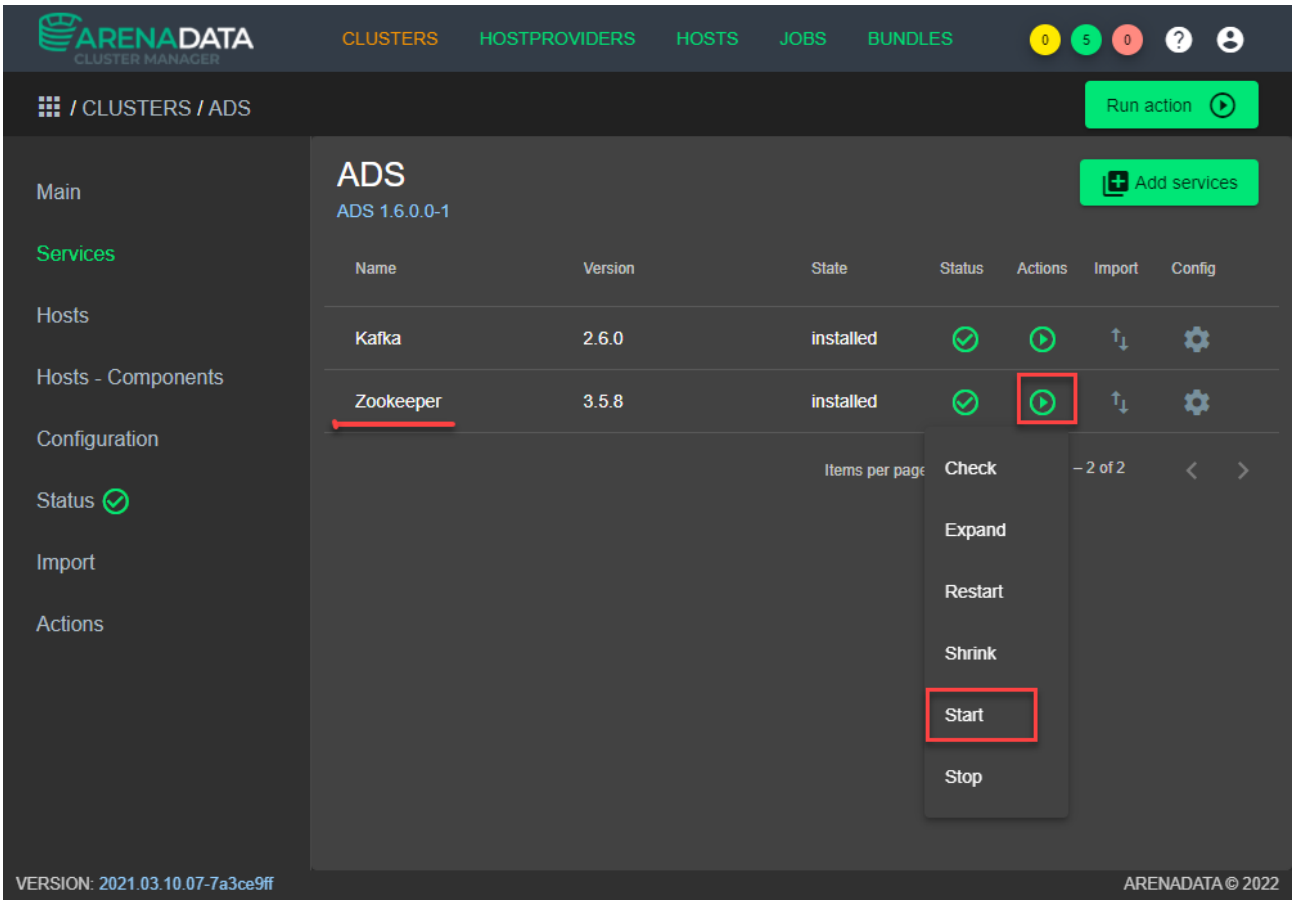

Рисунок - 3.1 Запуск сервиса Zookeeper через ADCM

3.22.2.2 Остановка

1. Авторизуйтесь в **Arenadata Cluster Manager (ADCM)**.
2. Перейдите в пункт кластер ADS.
3. В левом меню, выберите пункт **Services**.
4. В таблице со списком сервисов *ADS*, в строке *Zookeeper*, нажмите значок **Actions** и нажмите кнопку **Stop** (см. Рисунок - 3.1).

3.22.2.3 Перезапуск

Для перезапуска **Zookeeper** в **Arenadata Cluster Manager (ADCM)**, выполните следующие действия:

1. Авторизуйтесь в **Arenadata Cluster Manager (ADCM)**.
2. Перейдите в пункт кластер ADS.
3. В левом меню, выберите пункт **Services**.
4. В таблице со списком сервисов *ADS*, в строке *Zookeeper*, нажмите значок **Actions** и нажмите кнопку **Restart** (см. Рисунок - 3.1).

3.22.3 Запуск и остановка сервисов ADS через консоль

Для ручной остановки и запуска необходимо подключиться по SSH на сервер и с правами `sudo` использовать штатную функцию `systemctl`.

Внимание:
Сервис **Kafka** всегда необходимо запускать после сервиса **Zookeeper**.

Например:

sudo systemctl stop kafka sudo systemctl stop zookeeper sudo systemctl start zookeeper sudo systemctl start kafka

3.22.4 Запуск и остановка сервисов ADS через ADCM

Графический пользовательский интерфейс Arenadata Cluster Manager (ADCM) предоставляет возможность независимо выполнять операции запуска, остановки и перезапуска для сервисов *Kafka* и *Zookeeper*.

Запуск и остановка Arenadata Streaming (ADS) осуществляется через Arenadata Cluster Manager (ADCM) (см. Рисунок - 3.2).

3.22.4.1 Запуск

Для запуска Arenadata Cluster Manager (ADCM), выполните следующие действия:

1. Авторизуйтесь в Arenadata Cluster Manager (ADCM).
2. Перейдите в пункт кластер ADS.
3. Нажмите кнопку **Run actions** и выберите в контекстном меню пункт **Start** (см. Рисунок - 3.2).

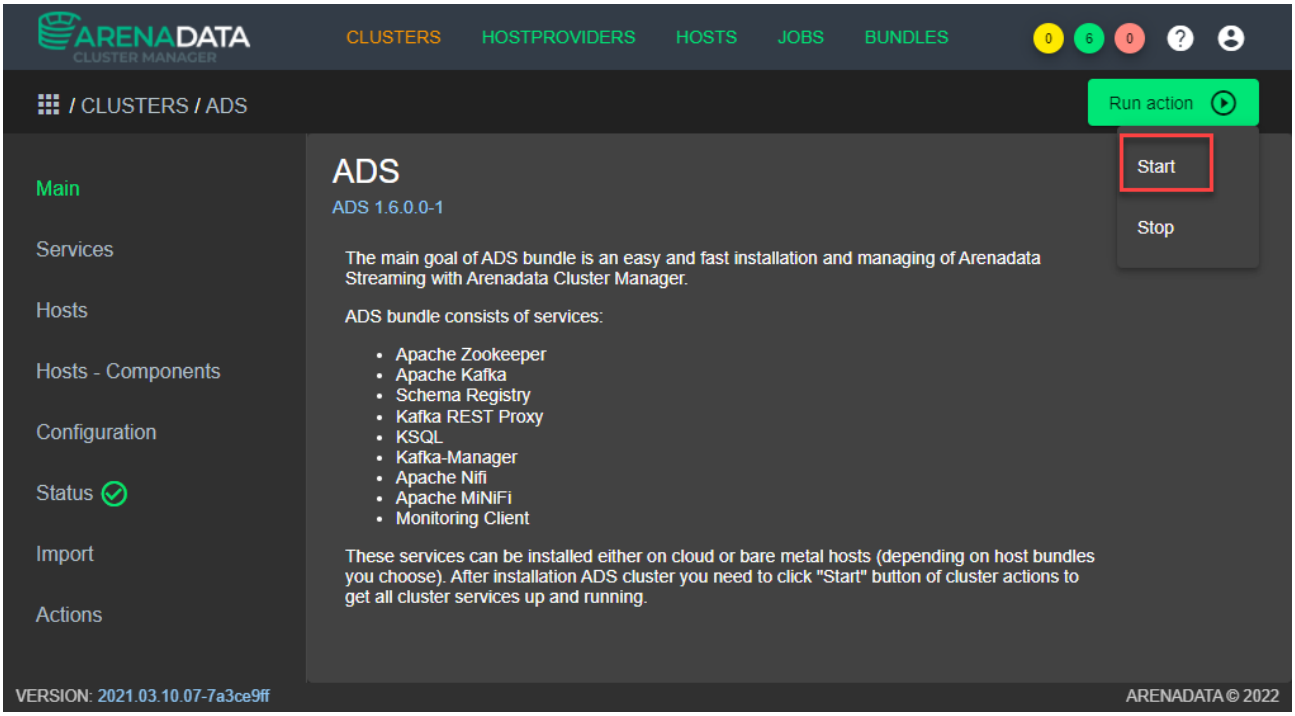

Рисунок - 3.2 Запуск Arenadata Streaming (ADS) в ADCM

3.22.4.2 Остановка

Внимание:
Остановка Arenadata Streaming (ADS) приведет к остановке сервисов *Zookeeper* и *Kafka*.

Для остановки Arenadata Cluster Manager (ADCM), выполните следующие действия:

1. Авторизуйтесь в Arenadata Cluster Manager (ADCM).
2. Перейдите в пункт кластер ADS.
3. Нажмите кнопку **Run actions** и выберите в контекстном меню пункт **Stop** (см. Рисунок - 3.3).

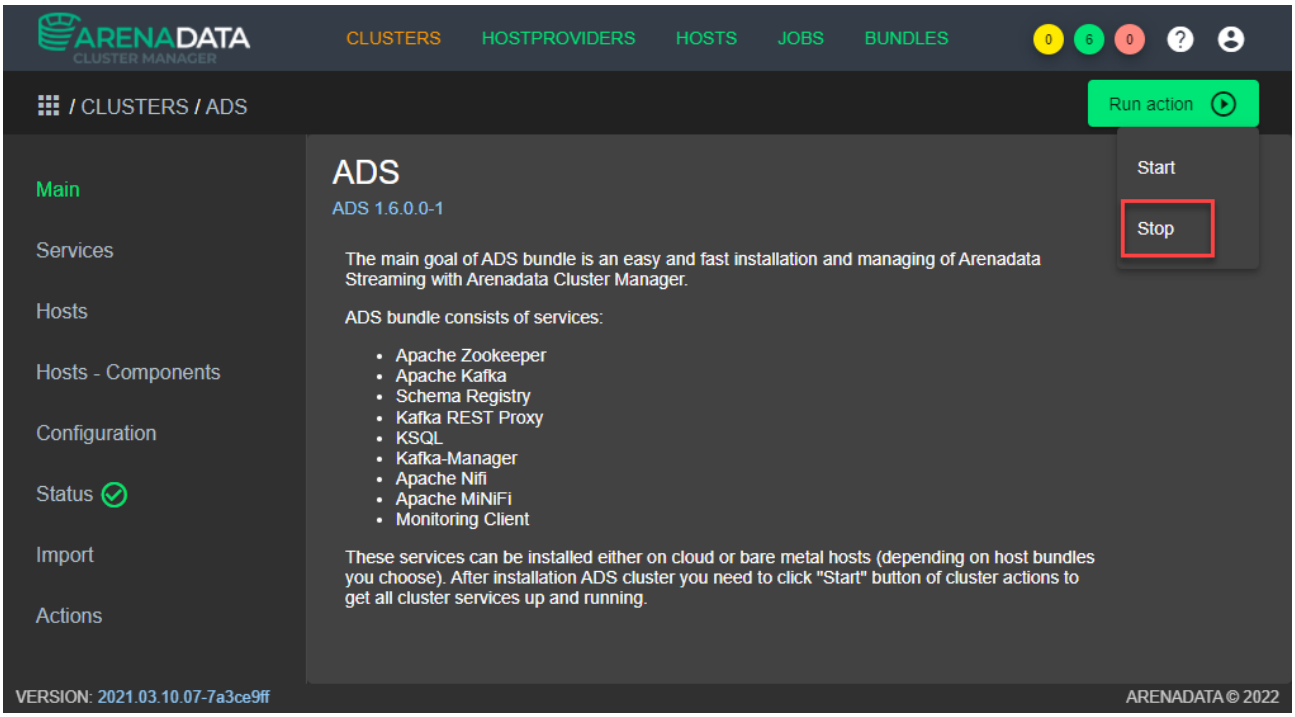

Рисунок - 3.3 Остановка Arenadata Streaming (ADS) в ADCM

4 БЕКАПИРОВАНИЕ ВИТРИНЫ ДАННЫХ НСУД

4.1 Состав резервной копии Компонента «Витрина данных»

Резервному копированию в Компоненте «Витрина данных» подлежат следующие компоненты:

- логическая модель Prostore;
- физические данные в СУБД;
- часть модулей слоя адаптеров, хранящих в Zookeeper данные, подлежащие бекапированию.

Таблица 4.7 Перечень модулей, подлежащих бекапированию

Имя модуля	**Zookeeper, из которого выполняется бекап**	**Путь хранения Zookeeper, подлежащий бекапированию**
CSV-Uploader	ADS	`/adapter/[env]/csv-uploader/config`
rest-uploader	ADS	`/adapter/[env]/rest-uploader/ids` `/adapter/[env]/rest-uploader/conditions`
smev3-adapter	настройка zookeeper.connect-string	`/adapter/[env]/smev3-adapter/[paramstorage.basePath]` `/adapter/[env]/smev3-adapter/[deltastorage.basePath]`

podd-adapter-replicator	ADS	`/adapter/[env]/podd-adapter-replicator/subscription`
		`/adapter/[env]/podd-adapter-replicator/replication`
		`/adapter/[env]/podd-adapter-replicator/delta`

counter-provider	ADS	`/adapter/[env]/counter-provider/counters/[counter-name]`

4.2 Описание и механизм работы утилиты Backup Manager

4.2.1 Описание утилиты Backup Manager

Утилита **Backup manager** разработана для:

- реализации механизма резервного копирования Витрины данных и восстановления из резервной копии;
- оркестрации процесса резервного копирования слоя адаптеров и утилиты **DTM-tools**, осуществляющей резервное копирование логической модели базы данных и физических данных СУБД, входящих в инсталляцию.

В конфигурации утилиты сохранены:

- путь к **DTM-tools** и параметры необходимые для работы утилиты **DTM-tools**:
  - маска топиков для бекапирования данных СУБД backend.backup.(datamart);
  - адрес Prostore (с переопределением через команду запуска);
- адрес Kafka и имена топиков для:
  - передачи команд модулям адаптера;
  - получения статусов модулей адаптера;
  - бекапирования данных бекенда Витрины;
  - бекапирования данных модулей адаптеров;

– путь к рабочей директории резервного копирования (доступный файловому коннектору) с переопределением через команду запуска;
– хранимый объем топика (значение по умолчанию 1Тб);
– время хранения топиков бекапа (значение по умолчанию 3 суток).

4.2.2 Механизм работы утилиты Backup manager

Backup Manager запускается в консольном режиме, принимает команду в качестве аргумента, обеспечивает выполнение 2 операций:

– **backup** - резервное копирование, вторым аргументом указывается целевой архив;
– **restore** - восстановление из резервной копии.

В процессе бекапирования или восстановления из бекапа утилита **Backup Manager** выводит в консоли сообщения об основных шагах и выполняемых операциях вида:

– отправлена команда остановки обработки запросов;
– отправлена команда бекапирования датамарта (datamart).

Также транслируется консольный вывод утилиты **DTM-tools**.

4.2.3 Создание резервной копии Компонент «Витрина данных» с использованием утилиты Backup Manager

Схематическое решение по созданию резервной копии Компонента «Витрина данных» с использованием утилиты **Backup Manager** представлено на рисунке ниже (см Рисунок - 4.1)

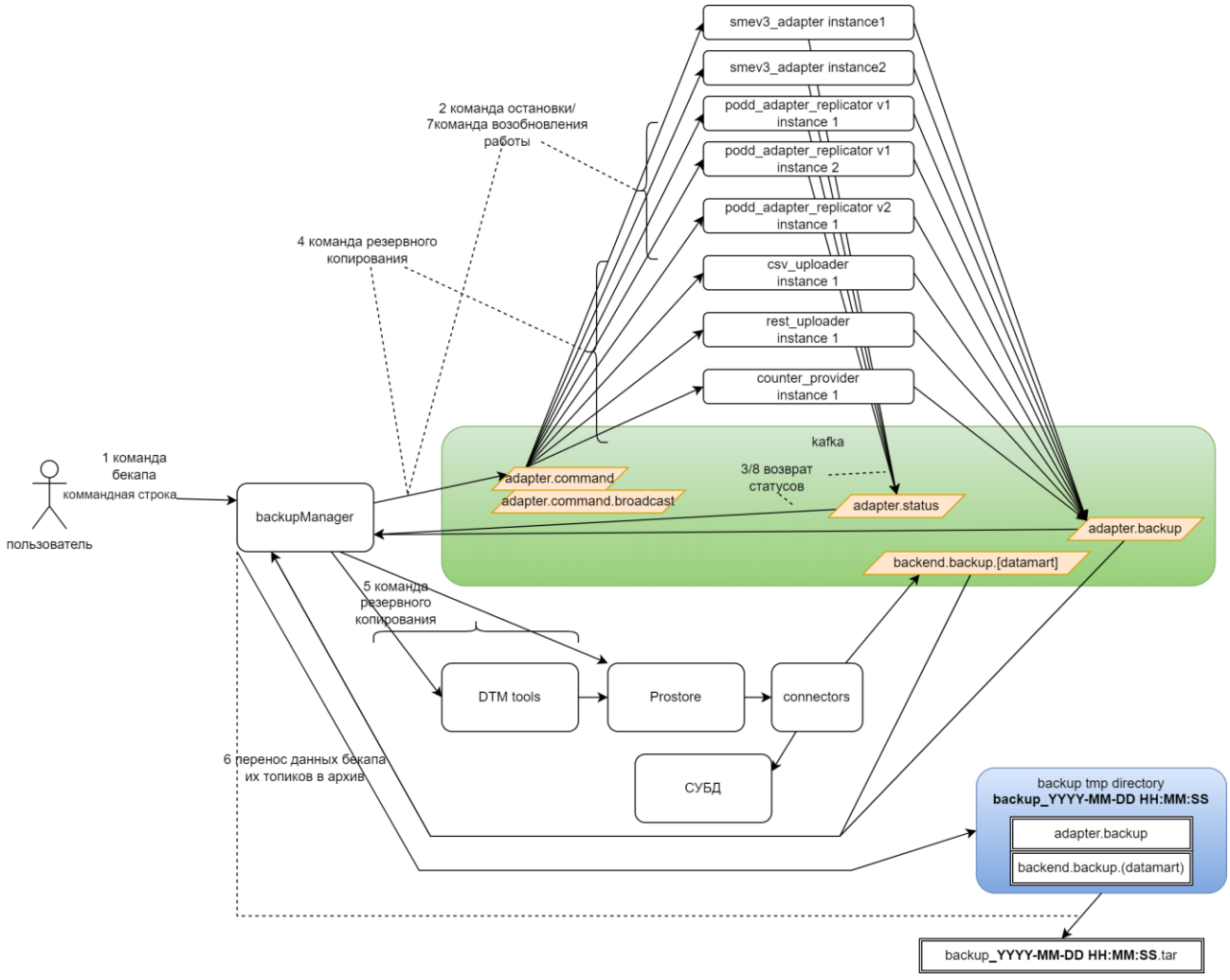

Рисунок - 4.1 Схема создания резервной копии Компонента «Витрины данных»

4.2.3.1 Механизм работы утилиты Backup manager при выполнении резервного копирования

Для запуска механизма резервного копирования, администратору системы нужно перейти в директорию backup-tools (утилиты **Backup Manager** и **DTM-Tools** должны быть расположены в одной директории) и выполнить в консоли команду `backup`:

```
java -jar backup-manager-1.10.0-SNAPSHOT.jar backup --dtmTools ./dtm-tools-1.12.0.jar --kafkaBrokers 10.81.7.90:12541 --prostore 10.81.7.90:12520
```

Процесс резервного копирования состоит из нескольких частей:

1. Подготовка к резервному копированию.
2. Остановка модулей, требующих консистентности с Prostore.
3. Ожидание остановки модулей, требующих консистентности с Prostore.
4. Резервное копирования слоя адаптеров Компонента «Витрина данных».
5. Резервное копирование логической модели Prostore и физических данных СУБД с использованием утилиты **DTM-tools**.
6. Формирование архива с резервной копией.
7. Запуск остановленных модулей слоя адаптеров.
8. Завершение работы.

4.2.4 Восстановление из резервной копии Компонента «Витрина данных» с использованием утилиты Backup Manager

Схематическое решение по восстановлению из резервной копии Компонента «Витрина данных» с использованием утилиты **Backup Manager** представлено на рисунке ниже (см Рисунок - 4.2)

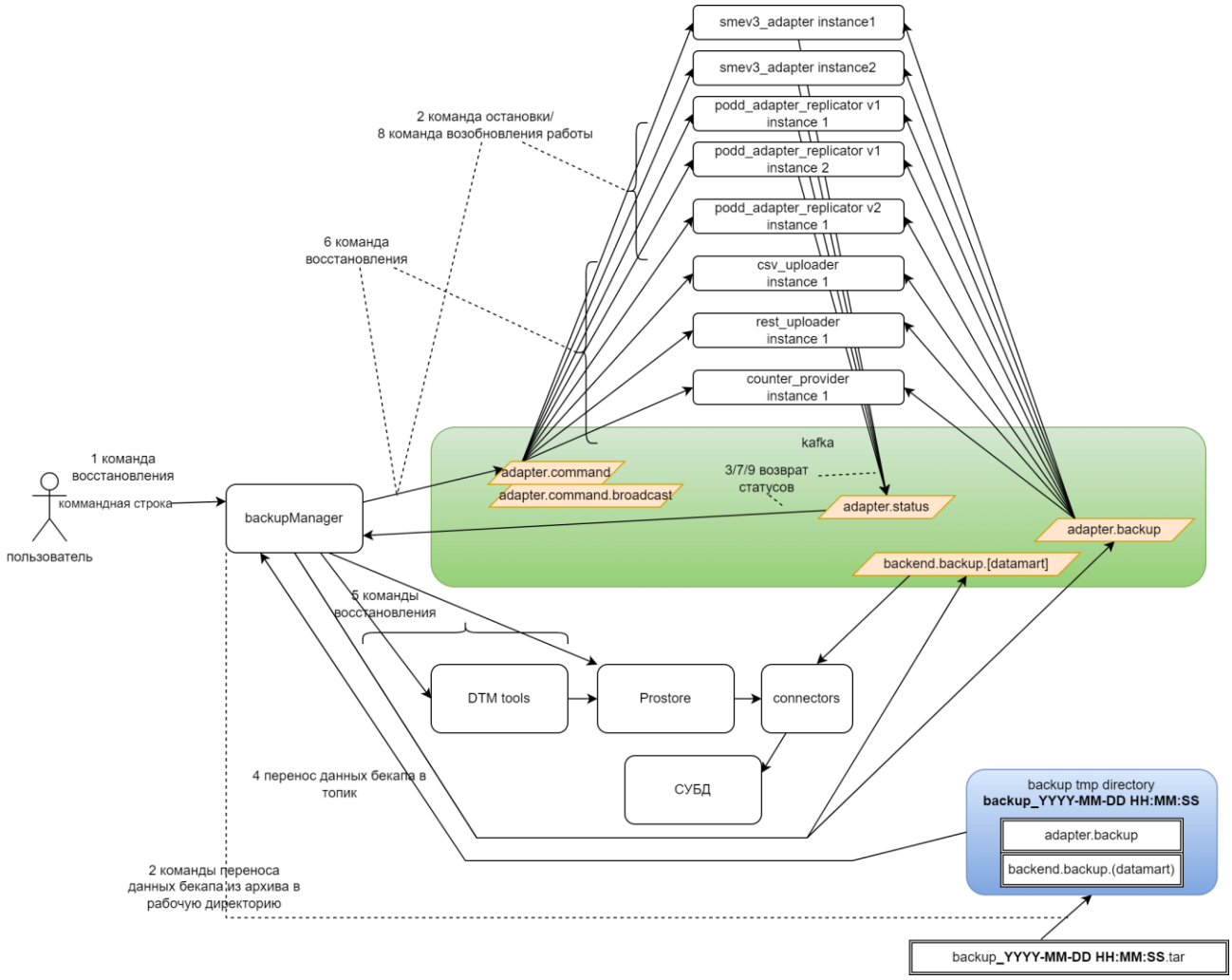

Рисунок - 4.2 Схема восстановления из резервной копии Компонента «Витрина данных»

Примечание:
В процессе восстановления данные могут содержаться частично, могут быть не консистентны, поэтому требуется предварительная остановка агента СМЭВ4 вручную перед запуском утилиты **Backup Manager**. После окончания процесса восстановления утилитой **Backup Manager**, работу агента СМЭВ4 следует возобновить. Модуль СМЭВ3 адаптер нужно отключить от сервера (отключить сетевое соединение). После восстановления сетевое подключение к СМЭВ3 следует возобновить.

Для запуска механизма восстановления из резервной копии, администратору системы нужно перейти в директорию backup-tools (утилиты **Backup Manager** и DTM-Tools должны быть расположены в одной директории) и выполнить в консоли команду `restore`:

```
java -jar backup-manager-1.10.0-SNAPSHOT.jar restore --dtmTools ./dtm-tools-1.12.0.jar
--kafkaBrokers 10.81.7.90:12541 --prostore 10.81.7.90:12520 --
backupArchive ./backup_YYYY-MM-DD HH:MM:SS.zip
```

Процесс восстановления из резервной копии состоит из нескольких частей:

1. Подготовка к восстановлению из резервной копии.
2. Остановка модулей слоя адаптеров, требующих консистентности с Prostore.
3. Ожидание остановки модулей, требующих консистентности с Prostore.
4. Перенос данных бекапа в топики резервного копирования.
5. Восстановление из резервной копии логической модели Prostore и физических данных СУБД с использованием утилиты **DTM-tools**.

6. Восстановление из резервной копии слоя адаптеров Компонента «Витрина данных» с остановкой модулей, требующих консистентности с Prostore.
7. Ожидание восстановления модулей слоя адаптеров.
8. Запуск остановленных модулей слоя адаптеров.
9. Завершение работы.

4.3 Реализация бекапирования в слое адаптеров Компонента «Витрина данных»

4.3.1 Работа модулей для обеспечения резервного копирования

Модули, подлежащие бекапированию, подписываются на топик `adapter.command` под одним groupId: `(имя модуля)_adapter_command` и создают продюсеров на топики:

- `adapter.status`;
- `adapter.backup`.

Все топики размещаются во внутренней кафке ADS.

Модули требующие консистентности с Prostore дополнительно подписываются на топик `adapter.command.broadcast` под уникальными groupId `(случайными (имя модуля)_(UUID))`.

Обработка команды не зависит от топика, через который она получена.

4.3.1.1 Сообщения в топиках команд

Таблица 4.8 Сообщения в топиках команд

Назначение команды	Топик	Ключ	Значение
Приостановка обработки запросов для модулей, которым требуется консистентность с Prostore	`adapter.command.broadcast`	backup.pause	null
Возобновление обработки запросов для модулей, которым требуется консистентность с Prostore	`adapter.command.broadcast`	backup.resume	null
Запрос персистированых данных из Zookeeper для резервной копии	`adapter.command`	backup.get	backupId
Применение данных резервной копии и запись персистированных данных в Zookeeper	`adapter.command`	backup.set	null

4.3.1.2 Статусы модулей

Статусы модулей возвращаются через компактный топик `adapter.status`

Формат сообщения статуса:

ключ в формате json (идентификатор вертикла и ключ шаблона присутствуют только для smev3-adapter):

```
{
    "group": "dev.nsud",
    "name": "smev3-adapter",
    "version": "4.0.13",
    "instance": "6f58378a-2205-42c9-80fc-c028ab12a3ba",
    "backupId": "2228378a-2205-42c9-80fc-c028ab12a222"
}
```

значение в формате json:

```
{
    "timestamp": "2023-02-17T12:10:45Z",
    "status": "started"
}
```

4.3.1.2.1 Значения статусов

Статус	Описание
started	работает
stopping	приостановка модуля для бекапирования
stopped	модуль приостановлен для бекапирования
restoring	модуль восстановлен из резервной копии
restored	модуль восстановлен из резервной копии
error_restoring	ошибка восстановления из резервной копии
error_stopping	ошибка приостановки модуля для бекапирования

4.4 Механизм приостановки модулей, требующих консистентности с Prostore

Обеспечение консистентности с Prostore реализовано для каждого экземпляра следующих модулей:

- smev3-adapter;
- podd-adapter-replicator.

Все экземпляры модулей, требующих консистентности с Prostore подписаны на топик `adapter.command.broadcast` с уникальной groupId консьюмера: (имя модуля)_(UUID)) groupId.

4.4.1 Приостановка модулей, требующих консистентности с Prostore

1. Каждый экземпляр модуля считывает команду `backup.pause` из топика `adapter.command.broadcast` и отправляет статус `stopping` в топик `adapter.status`.
2. В каждом экземпляре модуля выполняется процесс приостановки процессов, влияющих на консистентные с Prostore данные.
3. После остановки всех процессов, каждый экземпляр модуля отправляет статус `stopped` в топик `adapter.status`.

4.4.2 Восстановление модулей, требующих консистентности с Prostore

1. Каждый экземпляр модуля считывает команду backup.resume из топика `adapter.command.broadcast`.
2. В каждом экземпляре модуля выполняется повторный запуск процессов, влияющих на консистентные с Prostore данные.
3. После остановки всех процессов, каждый экземпляр модуля отправляет статус started в топик `adapter.status`.

4.5 Механизм резервного копирования и восстановления из резервной копии в модулях слоя адаптеров

Модули, данные которых необходимы для резервного копирования:

- CSV-Uploader;
- REST-Uploader;

- smev3-adapter;
- podd-adapter-replicator v1;
- counter-provider.

Резервное копирование выполняет только один из экземпляров каждого типа модулей, который успел считать сообщение из топика `adapter.command`.

4.5.1 Механизм резервного копирования модулей слоя адаптеров

При выполнении резервного копирования все модули, участвующие в бекапировании:

1. Подписаны на топик `adapter.command` с общей `qroupId` для каждого типа модулей.
2. Один из экземпляров модуля считывает сообщение`` backup.get`` из топика `adapter.command`.
3. Считывает метаданные, подлежащие бекапированию по пути в Zookeeper, в соответствии с путями хранения в сервисной БД.
4. Возвращает данные через топик `adapter.backup` в виде json.

В ключе сообщения формируются стандартные данные о версии и сборке:

```
{
    "group": "ru.itone.dtm.data.uploader"
    "name": "data-uploader",
    "version": "1.1.0-SNAPSHOT",
    "instance": "6f58378a-2205-42c9-80fc-c028ab12a3ba",
    "backupId": "2228378a-2205-42c9-80fc-c028ab12a222",
    "gitCommit": "a7c7770404ef61f62496983b783ef7b442989d74",
    "dateCommit": "2023-02-17T12:10:45Z",
    "branchCommit": "develop",
    "buildDate": "2023-02-17T12:11:36.627Z",
    "buildUnixTime": "1676635896"
}
```

В значении сообщения размещаются перситируемые данные, пример для модуля counter-provider.

```
{
"path":"/dev/counter-provider",
"value":"test_data",
"children":[
    {
        "path":"/dev/counter-provider/child2",
        "value":"",
        "children":[]
    },
    {
        "path":"/dev/counter-provider/child1",
        "value":"child1_data",
        "children":[]
    }
]
}
```

4.5.2 Механизм восстановления из резервной копии модулей слоя адаптеров

При выполнении резервного копирования все модули, участвующие в восстановлении из резервной копии:

1. Подписаны на топик `adapter.command` с общей qroupId для каждого типа модулей.

2. Один из экземпляров модуля считывает сообщение `backup.set` из топика `adapter.command`.
3. Отправляют статус `restoring` в топик `adapter.status`.
4. Удаляют данные по пути хранения метаданных в соответствии с путями хранения в сервисной БД.
5. Разбирают сообщения в топике `adapter.backup`: сначала фильтруют данные, относящиеся к модулю.
6. Записывают данные в сервисную БД.
7. Отправляют статус `restored` в топик `adapter.status`.

4.6 Поведение в случае ошибок при выполнении резервного копирования

При возникновении ошибок в процессе резервного копирования, утилита **Backup Manager** выполняет компенсирующие действия и завершает выполняемый процесс.

В случае, если ошибки возникли в процессе восстановления из резервной копии, стоит обратить внимание на тот факт, что в этом случае Компонент «Витрина данных» будет находится в не консистентном состоянии, требуется оперативный разбор ошибок и повторное восстановление из резервной копии.

4.6.1 Ошибки резервного копирования и восстановления из резервной копии

Ошибки, возможные в процессе резервного копирования/восстановления из резервной копии, и пути их устранения приведены ниже (см. Таблица 4.9)

Таблица 4.9 Ошибки резервного копирования и восстановления из резервной копии

Ошибка	Описание ошибки	Действия утилиты Backup Manager	Устранение ошибки
«Observed active backupManager process, file backup.lock exists»	Не удален файл `backup.lock`	– завершает процесс бекапирования/ восстановления с выводом ошибки «Observed active backupManager process, file backup.lock exists» – выводит финальный статус: `BACKUP/RESTORE is failed`	Может возникать при прерванной работе утилиты, требуется ручное удаление файла
«Error stopping (module) (instanse) (verticle) (key), see modul log for detail»	Ошибка приостановки одного из инстансов модулей, требующих консистентности с Prostore	– отправляет команду `backup.resume` на восстановление работы модулей в топик `adapter.command.broadcast` без ожидания статусов о восстановлении – завершает процесс бекапирования/ восстановления с выводом ошибки «error stopping (module) (instanse) (verticle) (key), see modul log for detail» – выводит финальный статус: `BACKUP/RESTORE is failed`	Требуется анализ логов модуля, в котором возникла ошибка, после ее устранения, повтор процесса бекапирования/восстановления
«timeout	Таймаут	– отправляет команду	Требуется анализ логов

Ошибка	Описание ошибки	Действия утилиты Backup Manager	Устранение ошибки
stopping (module) (instanse) (verticle) (key), see modul log for detail»	приостановки одного из инстансов модулей, требующих консистентности с Prostore	backup.resume на восстановление работы модулей в топик `adapter.command.broadcast` без ожидания статусов о восстановлении – завершает процесс бекапирования/ восстановления с выводом ошибки: «timeout stopping (module) (instanse) (verticle) (key), see modul log for detail» – выводит финальный статус: `BACKUP/RESTORE is failed`	модуля, в котором возникла ошибка, после ее устранения, повтор процесса бекапирования/восстановления
«timeout starting (module) (instanse), see modul log for detail»	Таймаут восстановления работоспособности одного из инстансов модулей, требующих консистентности с Prostore	– завершает процесс бекапирования/ восстановления завершается с выводом ошибки : «timeout starting (module) (instanse), see modul log for detail» – завершает процесс восстановления с выводом ошибки: «error restoring (module) (instanse), see modul log for detail» – выводит финальный статус: `RESTORE is failed`	Требуется анализ логов модуля, в котором возникла ошибка, после ее устранения, повтор процесса бекапирования/восстановления
«timeout restoring (module) (instanse), see modul log for detail»	Таймаут восстановления из резервной копии модуля слоя адаптеров	– отправляет команду `backup.resume` на восстановление работы модулей в топик `adapter.command.broadcast` без ожидания статусов о восстановлении – завершает процесс восстановления с выводом ошибки: «timeout restoring (module) (instanse), see modul log for detail» – выводит финальный статус: `RESTORE is failed`	Требуется анализ логов модуля, в котором возникла ошибка, после ее устранения, повтор процесса бекапирования/восстановления
Ошибки утилиты DTM-tools при создании резервной копии	Ошибки не формализованы	В случае не успеха процесса бекапирования утилиты DTM-tools: – отправляет команду `backup.resume` на восстановление работы модулей в топик `adapter.command.broadcast` без ожидания статусов о восстановлении – завершает процесс бекапирования с выводом	Требуется анализ логов модуля, в котором возникла ошибка, после ее устранения, повтор процесса бекапирования/восстановления

Ошибка	Описание ошибки	Действия утилиты Backup Manager	Устранение ошибки
		ошибки, переданной DTM-tools – выводит финальный статус: `BACKUP is failed`	
Ошибки утилиты DTM-tools при восстановлении из резервной копии	Ошибки не формализованы	В случае не успеха процесса бекапирования утилиты DTM-tools: – отправляет команду `backup.resume` на восстановление работы модулей в топик `adapter.command.broadcast` без ожидания статусов о восстановлении – завершает процесс восстановления из бекапа с выводом ошибки, переданной DTM-tools – выводит финальный статус: `RESTORE  is failed`	Требуется анализ логов модуля, в котором возникла ошибка, после ее устранения, повтор процесса бекапирования/восстановления

4.6.2 Поведение в случае остановки утилиты Backup Manager в процессе снятия/восстановления из резервной копии

В случае экстренного прекращения работы утилиты командой `kill` возможно как не консистентное состояние витрины (при восстановлении из резервной копии), так как и ее частичная неработоспособность, так как утилита **Backup Manager** в этом случае не успеет выполнить компенсирующие действия и восстановить работу остановленных модулей.

Рекомендуется в случае экстренного прекращения работы утилиты перезапустить поды модулей podd-adapter-replicator и smev3-adapter.

4.6.3 Ограничения

1. При выполнении DDL операций бэкап завершается ошибкой.
2. Для корректного выполнения функции бекапирования для каждого слоя адаптеров должен быть развернут свой экземпляр Kafka.

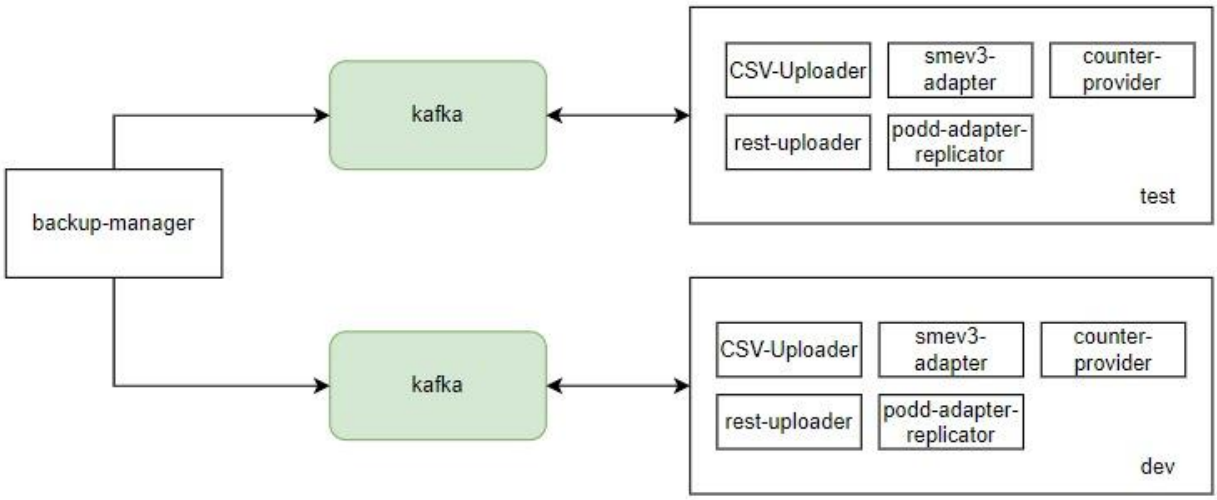

Рисунок - 4.3 Схема развертывания экземпляра Kafka

5 ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ

5.1 Дополнительные возможности конфигурации Стандарт

Примечание:

Инструкции данного раздела не выполняются в рамках первичной установки компонентов программы. Необходимость выполнения действий данного раздела определяется в процессе эксплуатации программы.

5.1.1 Установки опциональных приложений

Сервера сбора, хранения и индексирования логов устанавливаются независимо от наличия или отсутствия других приложений.

Конкретные средства логирования и мониторинга не входят в состав данного решения и выбираются в соответствии с требованиями конкретного ведомства.

Обязательно необходимо установить, как минимум один из серверов базы данных ADB (Greenplum), ADQM (Clickhouse) или ADG (Tarantool).

Обязательно нужно установить, как минимум одно программное обеспечение для работы со СМЭВ:

- СМЭВ3-адаптер;
- группа приложений состоящих из СМЭВ4-адаптера - Модуль исполнения запросов, Агента СМЭВ4, Диспетчер сообщений для СМЭВ4 «Kafka» (ADSP).

Агента СМЭВ4 и Диспетчер сообщений для СМЭВ4 «Kafka» (ADSP) не входят в состав данного решения и устанавливаются отдельно, согласно соответствующей документации.

5.1.2 Материлиазованные представления

Материализованное представление — это набор записей, который является результатом исполнения SELECT-запроса.

Материализованное представление позволяет предварительно вычислить результат запроса и сохранить его для будущего использования.

SELECT-запрос, на котором строится представление, может обращаться к данным одной или нескольких логических баз данных.

Материализованное представление строится на основе данных одной СУБД хранилища (далее — СУБД-источник), а его данные размещаются в других СУБД.

Это позволяет создавать инсталляции, где одна СУБД служит полноценным хранилищем исходных данных, а остальные СУБД отвечают за быструю выдачу данных по запросам чтения. В текущей версии системы доступно создание материализованных представлений в **ADQM**, **ADG** и **ADP** на основе данных **ADB**.

Материализованное представление помогает ускорить запросы к данным в следующих случаях:

- если представление содержит результаты сложного запроса, который на исходных данных выполняется дольше;
- если запросы к представлению возвращают значительно меньше данных, чем запросы к исходным данным;
- если запросы относятся к категории, которую СУБД хранилища, где размещены данные представления, выполняет более эффективно, чем СУБД-источник

(например, ADG быстрее всех из поддерживаемых СУБД обрабатывает чтение по ключу).

Материализованное представление дает доступ к актуальным и архивным данным. Чтение горячих данных из представления недоступно: это позволяет избежать чтения изменений, загруженных из СУБД-источника только частично.

Данные материализованного представления хранятся аналогично данным логических таблиц — в физических таблицах хранилища, которые автоматически создаются при создании представления (см. Рисунок - 5.1).

Рисунок - 5.1 Связи материализованного представления с физическими таблицами

Система поддерживает целостность данных материализованных представлений, размещенных в СУБД-приемнике, периодически синхронизируя их с СУБД-источником (см. Рисунок - 5.2).

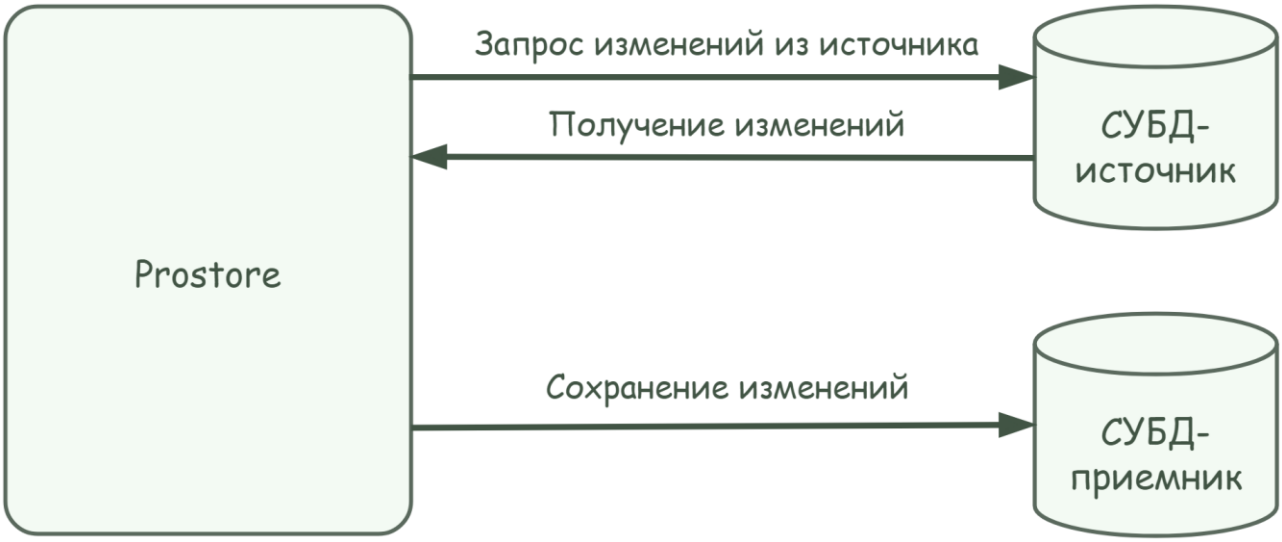

Рисунок - 5.2 Синхронизация материализованных представлений

Для материализованных представлений реализована возможность создания, чтения, записи, удаления из **ADB** в **Postgres**.

Более подробная информация об операциях над мат.представлениями изложена в документации Prostore. Загрузка и обновление данных недоступны для материализованных представлений.

Примечание:

Информацию о DDL-запросе, создавшем представление, можно получить с помощью запроса GET_ENTITY_DDL.

Примечание:

По умолчанию система ведет статистику обработки запросов к данным логических сущностей. Чтобы получить статистику, выполните запрос GET_ENTITY_STATISTICS..

При запросе или выгрузке данных из материализованного представления можно указать момент времени, по состоянию на который запрашиваются данные. Если момент времени не указан, система возвращает (выгружает) данные, актуальные на момент последней синхронизации представления, иначе — данные, актуальные на запрашиваемый момент времени.

При запросе или выгрузке данных на указанный момент времени может оказаться, что материализованное представление отстало от СУБД-источника и не содержит запрошенные данные. В этом случае система перенаправляет запрос к исходным таблицам СУБД-источника (см. раздел Маршрутизация запросов к материализованным представлениям). Перенаправленный запрос может выполняться дольше, однако это позволяет получить данные, полностью актуальные на указанный момент времени.

Синхронизация материализованных представлений

Система периодически проверяет, нужно ли синхронизировать материализованные представления окружения с СУБД-источником. Периодичность проверки настраивается в конфигурации системы с помощью параметра `MATERIALIZED_VIEWS_SYNC_PERIOD_MS`; по умолчанию проверка запускается раз в 5 секунд.

Примечание:

При необходимости синхронизацию материализованных представлений можно отключить, установив значение параметра `MATERIALIZED_VIEWS_SYNC_PERIOD_MS` равным 0.

Проверка материализованных представлений запускается по таймеру. Другие события (например, создание представления или загрузка данных) не запускают проверку представлений. При срабатывании таймера система проверяет, появились ли в СУБД-источнике дельты, закрытые после последней синхронизации и, если такие дельты появились, система синхронизирует материализованные представления с СУБД-источником.

Примечание:

Материализованное представление, основанное на таблицах из разных логических баз данных, синхронизируется при наличии новых дельт в основной логической базе данных — в той, которой принадлежит представление.

Количество одновременно синхронизируемых представлений задается в конфигурации системы с помощью параметра `MATERIALIZED_VIEWS_CONCURRENT`. По умолчанию одновременно синхронизируется максимум два представления, а остальные, если они есть, ожидают следующего цикла проверки.

Данные представления синхронизируются отдельно по каждой закрытой дельте — с полным сохранением изменений, выполненных в этих дельтах. В каждой дельте для материализованного представления рассчитывается и сохраняется результат запроса, указанного при создании этого представления. Таким образом, материализованное представление имеет такой же уровень историчности данных, как и исходные логические таблицы, на которых построено представление.

Если системе не удалось синхронизировать материализованное представление, она делает несколько повторных попыток. Максимальное количество таких попыток регулируется параметром конфигурации `MATERIALIZED_VIEWS_RETRY_COUNT`. По умолчанию система делает до 10 попыток. Если количество попыток исчерпано, но представление так и не удалось синхронизировать, система прекращает попытки синхронизировать это представление. В случае перезапуска системы счетчики попыток синхронизации обнуляются, и система снова пытается синхронизировать представления, которые остались несинхронизированными.

Примечание:

Статусы синхронизации материализованных представлений можно посмотреть с помощью запроса CHECK_MATERIALIZED_VIEW .

Пример синхронизации материализованного представления

Рассмотрим пример со следующими условиями:

- логическая БД marketing содержит логическую таблицу `sales` и материализованное представление `sales_by_stores`;
- логическая БД содержит две дельты:
  - дельта 0: в таблицу sales загружено две записи (с идентификаторами 100 и 101);
  - дельта 1: в таблицу sales загружено еще две записи (с идентификаторами 102 и 103);
- материализованное представление `sales_by_stores` содержит результат агрегации и группировки данных таблицы `sales` и построено на основе следующего запроса:

```
CREATE MATERIALIZED VIEW marketing.sales_by_stores (
      store_id INT NOT NULL,
      product_code VARCHAR(256) NOT NULL,
      product_units INT NOT NULL,
      PRIMARY KEY (store_id, product_code)
      )
      DISTRIBUTED BY (store_id)
      DATASOURCE_TYPE (adg)
      AS SELECT store_id, product_code, SUM(product_units) FROM marketing.sales
       WHERE product_code <> 'ABC0001'
       GROUP BY store_id, product_code
      DATASOURCE_TYPE = 'adb'
```

На рисунках ниже (см Рисунок - 5.3 и Рисунок - 5.4) показан порядок синхронизации материализованного представления `sales_by_stores`. В каждой дельте рассчитывается и сохраняется сумма по столбцу `product_units` таблицы sales с группировкой по столбцам `store_id` и `product_code`. При этом неважно, когда было создано материализованное представление: до дельты 0, после дельты 1 или в какой-то момент между этими дельтами.

Рисунок - 5.3 Состояние данных на момент дельты 0

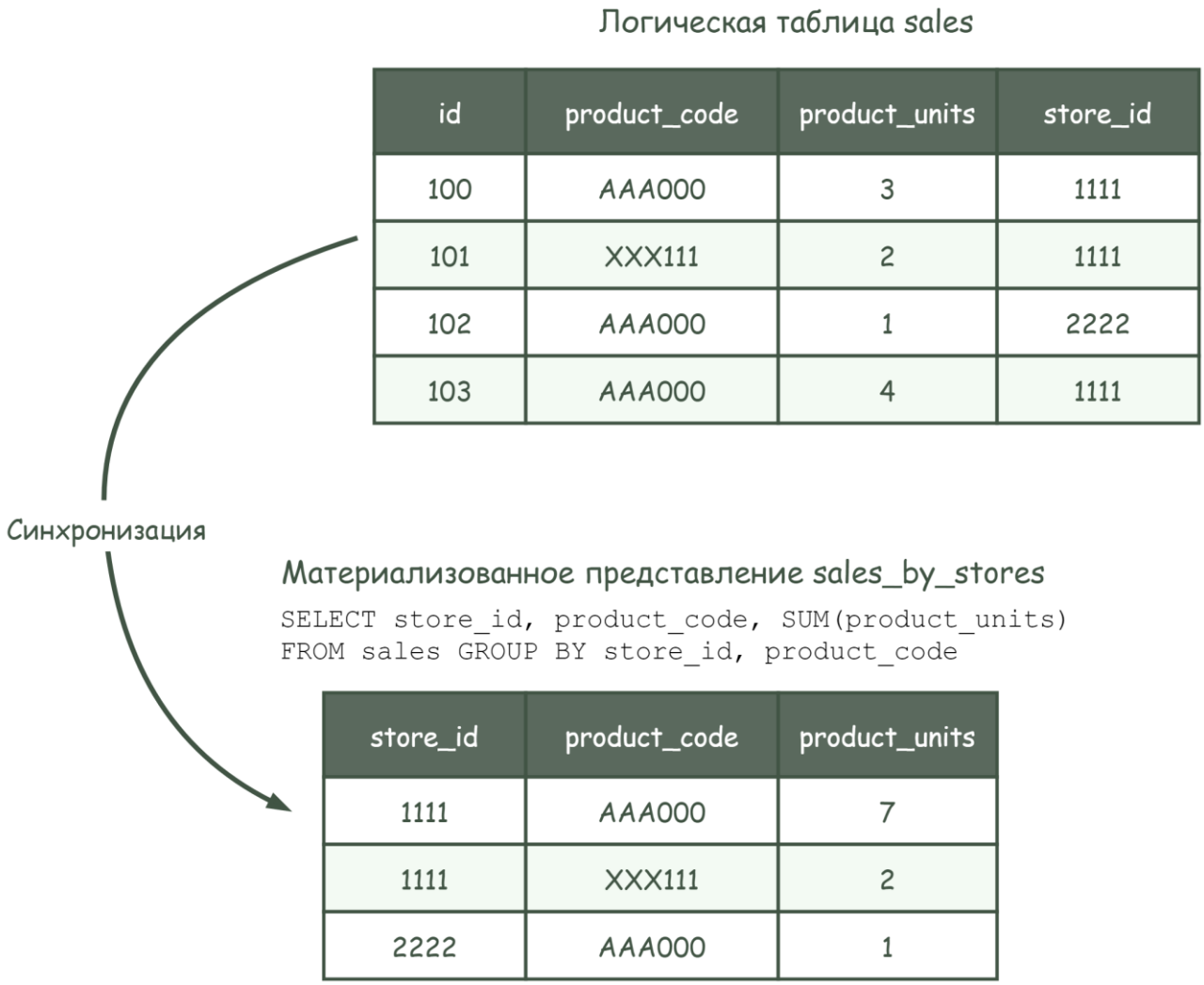

Рисунок - 5.4 Состояние данных на момент дельты 1

5.1.3 Маршрутизация запросов к материализованным представлениям

Запросы к данным материализованных представлений проходят все этапы маршрутизации, описанные выше, и затем — дополнительные этапы:

1. Если для материализованного представления не указано ключевое слово `FOR SYSTEM_TIME`, запрос направляется в СУБД, где размещены данные этого представления. Из представления выбираются данные, актуальные на момент его последней синхронизации.
2. Иначе, если ключевое слово `FOR SYSTEM_TIME` указано, система проверяет, есть ли в представлении данные за запрашиваемый момент времени:
   - Если в запросе есть ключевое слово `DATASOURCE_TYPE`, а данных за запрашиваемый момент времени в представлении нет, в ответе возвращается исключение.
   - Если в запросе нет ключевого слова `DATASOURCE_TYPE`:
     - Если данные есть в представлении, запрос направляется в СУБД, где размещены данные этого представления.
     - Иначе запрос направляется к исходным таблицам СУБД-источника, на которых построено представление.

Примечание:

В запросах к материализованным представлениям доступны не все выражения с ключевым словом `FOR SYSTEM_TIME` Подробнее см. в секции Доступность значений FOR SYSTEM_TIME раздела SELECT

5.1.4 Логирование

Лог-файлы компонентов могут быть найдены на соответствующих серверах, по относительным путям, описанным ниже (см. Таблица 5.1):

Таблица 5.1 Расположение лог-файлов на сервере

Наименование	**Относительный путь**
ClickHouse Server	`/var/log/clickhouse-server/clickhouse-server.log` `/var/log/clickhouse-server/clickhouse-server.err.log`
Greenplum Server	`/var/log/greenplum-server/greenplum-server.log` `/var/log/greenplum-server/greenplum-server.err.log`
Tarantool	`/var/log/tarantool-server/tarantool-server.log` `/var/log/tarantool-server/tarantool-server.err.log`
Apache Kafka	`/usr/lib/kafka/logs/*.log`
СМЭВ4-адаптер-Модуль исполнения запросов	`/opt/podd-migration/logs/application.log` `/opt/podd-adapter/logs/application.log`
СМЭВ3-адаптер	`/opt/smev3-adapter/logs/application.log`

ETL	`/opt/Airflow/logs`
	`/opt/spark/logs`
	`/opt/hadoop/logs`

REST-адаптер	`/opt/rest/logs`

5.1.5 Обновление

5.1.5.1 Менеджер кластера ADCM

Чтобы обновить **ADCM** вы должны сделать следующее:

1. Загрузить новый образ в докер:

```
docker pull arenadata/ADCM:latest
```

2. Остановить и удалить текущий контейнер:

```
docker stop ADCM
docker rm ADCM
```

3. Создать новый контейнер как указано в документации **ADCM**: https://docs.arenadata.io/adcm/user/install.html

5.1.5.2 Диспетчер сообщений ADS

Обновление кластера **ADS** доступно с версии 1.4.11

ADCM предоставляет возможность обновления существующего кластера **ADS**.

Процесс обновления состоит из двух последовательных шагов:

- Обновление бандла;
- Обновление кластера.

В текущей версии доступно обновление кластеров как версий 1.3.X, так и 1.4.X

5.1.5.2.1 Обновление бандла

Для обновления бандла необходимо:

1. Загрузить бандл **ADS** новой версии. После его загрузки на вкладке **Clusters** в строке кластера с более старой версией бандла в колонке **Upgrade** появляется пиктограмма, указывающая на возможность обновления (см. Рисунок - 5.10).

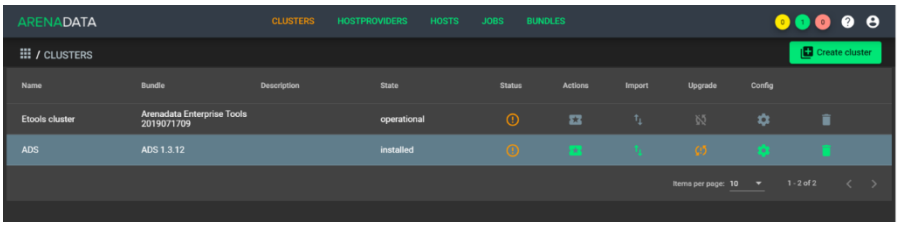

Рисунок - 5.10 Доступно обновление бандла

2. Нажать значок в колонке **Upgrade** и выбрать доступную требуемую версию из списка (см. Рисунок - 5.11).

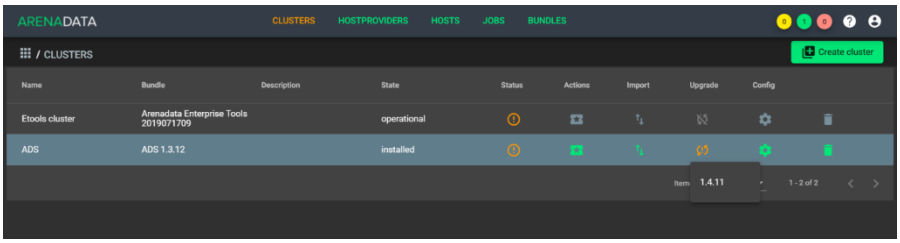

Рисунок - 5.11 Доступные обновления

3. В открывшемся диалоговом окне подтвердить действие, после чего кластер меняет состояние на `upgrade from 1.3.X` или `upgrade from 1.4.X` в зависимости от установленной версии бандла (см. Рисунок - 5.12).

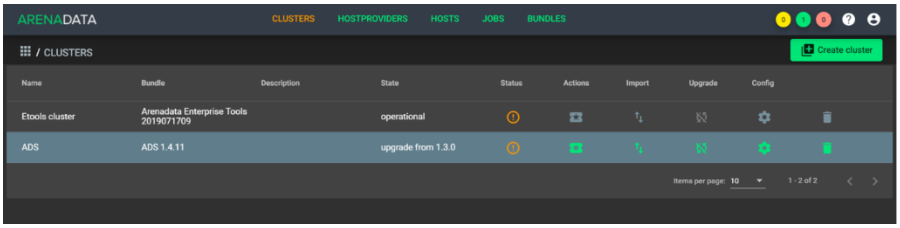

Рисунок - 5.12 Изменение состояния кластера после обновления

> Примечание:
>
> Если заданные по умолчанию настройки сервисов *Zookeeper*, *Kafka* изменены, то их необходимо скопировать и сохранить прежде, чем приступить к обновлению конфигураций сервисов.

В частности, это касается файлов `nifi.properties`, `zoo.cfg` и `server.properties` сервисов `Nifi`, `Zookeeper` и `Kafka` соответственно.

5.1.5.2.2 Обновление кластера

После завершения операции **Upgrade Configs** в кластере становится доступным действие **Upgrade**. Данная операция применяет новые настройки, полученные на предыдущем шаге, и обновляет пакеты всех сервисов до указанных версий.

1. В поле **Actions** для обновляемого кластера нажать на значок и выбрать действие **Upgrade** (см. Рисунок - 5.13).

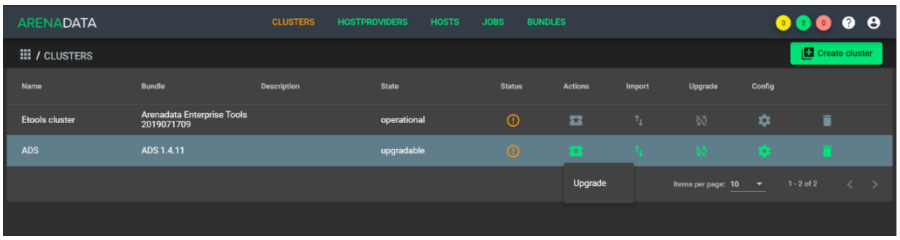

Рисунок - 5.13 Обновление пакетов сервисов

2. Подтвердить действие в открывшемся диалоговом окне нажатием кнопки **Run**.

После успешного завершения операции **Upgrade** кластер меняет свое состояние на `installed`.

Если заданные по умолчанию настройки сервисов были изменены перед обновлением, то после операции **Upgrade Configs** необходимо выполнить действия для соответствующих сервисов:

Перейти к настройкам сервиса *Zookeeper*, проверить раздел **zoo.cfg** и при необходимости внести сохраненные ранее изменения;

Перейти к настройкам сервиса Kafka, проверить разделы **Main** и **server.properties** и при необходимости внести сохраненные ранее изменения;

5.1.6 Миграция из Bare metal варианта установки в Kubernetes

В процессе миграции необходимо отделить модули, которым предстоит переехать в **Kubernetes** от тех, которые остаются в **Bare Metal** режиме инсталляции.

Миграции подлежат модули адаптеров Витрины данных и модули Prostore. **Kafka**, **Zookeeper** и **СУБД** остаются вне **Kubernetes**.

Для мигрирующего модуля оформляется K8S deployment, конфигурация `application.yml` и `logback.xml` размещаются в K8S `configmap`. При смене версии модуля необходимо актуализировать конфигурацию `application.yml` в соответствии с новой версией документации.

Альтернативно, вместо использования `application.yml` конфигурировать приложение можно через переменные окружения K8S контейнера.

Сервис исполнения запросов корректно может работать только в рамках одного пода.

Для модулей, имеющих HTTP-интерфейс, дополнительно формируется K8S service, обеспечивающий маршрутизацию к экземплярам модулей.

На диаграмме (см. Рисунок - 5.14) представлена миграция модуля исполнения запросов и Prostore.

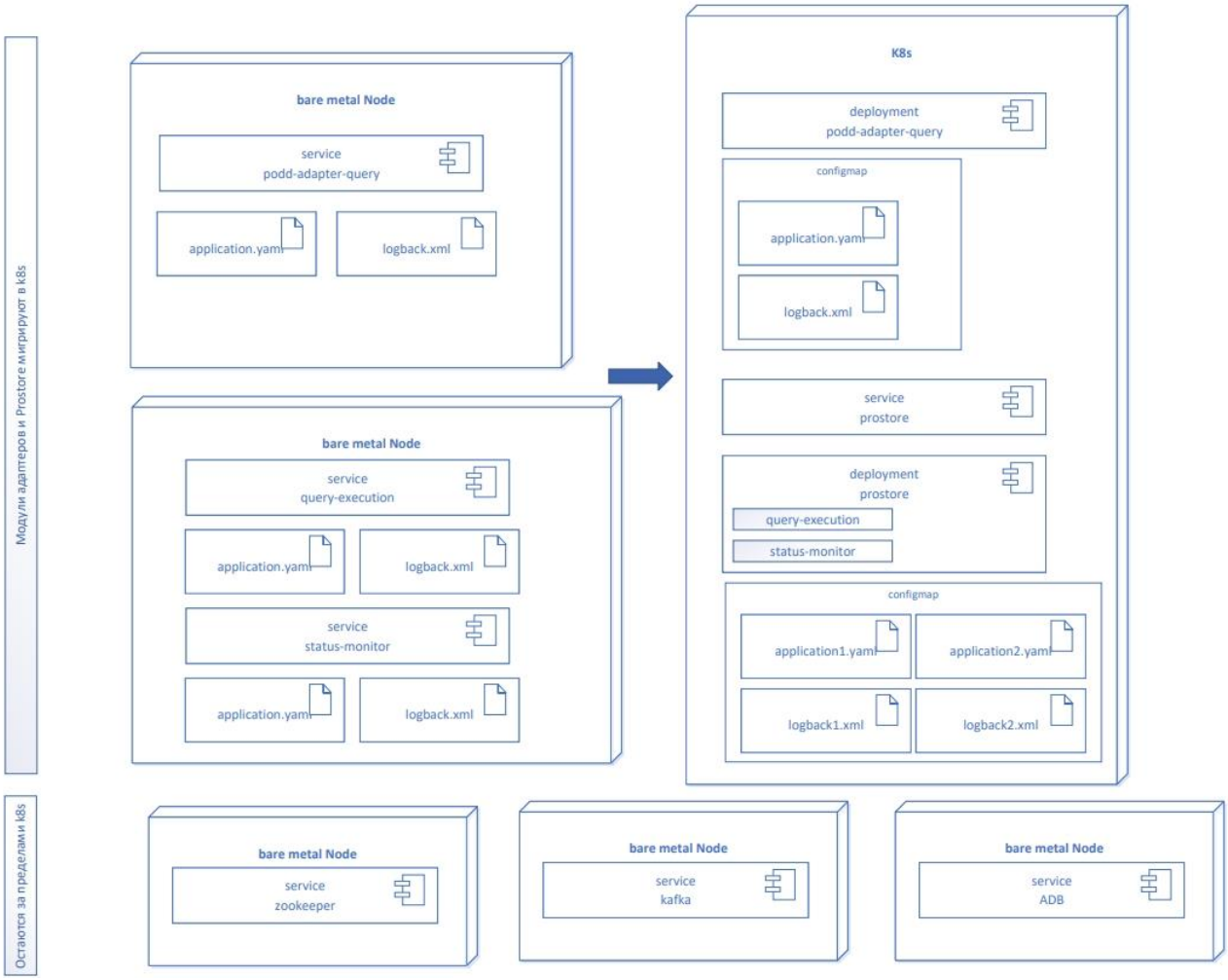

Рисунок - 5.14 Миграция в Kubernetes

Для миграции модуля в его корневой директории необходимо создать манифест файлы с инструкциями:

- `deployment`;
- `service`;
- `configmap`.

Примеры создания манифест файлов приведены ниже.

Создать объекты из манифест файлов в Kubernetes можно при помощи утилиты kubectl:

```
kubectl apply -f <FILE_NAME>
```

5.1.6.1 Примеры инструкций по развертыванию СМЭВ4-адаптера — Модуля исполнения запросов в Kubernetes

Пример создания файла `deployment`

```
apiVersion: apps/v1
kind: Deployment
metadata:
labels:
app.kubernetes.io/instance: podd-adapter-query
app.kubernetes.io/name: podd-adapter-query
name: podd-adapter-query
spec:
progressDeadlineSeconds: 600
replicas: 2
```

```
revisionHistoryLimit: 10
selector:
matchLabels:
     app.kubernetes.io/instance: podd-adapter-query
     app.kubernetes.io/name: podd-adapter-query
strategy:
rollingUpdate:
     maxSurge: 25%
     maxUnavailable: 25%
type: RollingUpdate
template:
metadata:
     annotations:
     prometheus.io/port: "9837"
     prometheus.io/scrape: "true"
     creationTimestamp: null
     labels:
     app.kubernetes.io/instance: podd-adapter-query
     app.kubernetes.io/name: podd-adapter-query
spec:
     containers:
     # Основной контейнер приложения
     # Настройки приложения через переменные среды
     - env:
     - name: AGENT_TOPIC_PREFIX
          value: demo_view.
     - name: DTMDB_COUNT
          value: "5"
     - name: DTMDB_DRIVER
          value: ru.datamart.prostore.jdbc.Driver
     - name: DTMDB_FETCH_SIZE
          value: "1000"
     - name: DTMDB_HOST
          value: prostore
     - name: DTMDB_MAX_POOL_SIZE
          value: "5"
     - name: DTMDB_PORT
          value: "9090"
     - name: DTMDB_SUBPROTOCOL
          value: prostore
     - name: ENVIRONMENT_NAME
          value: k8s
     - name: JAVA_OPTS
          value: -Xmx2g
     - name: JDBC_VERSION
          value: 5.8.0
     - name: K8S_SERVICE_NAME
          value: podd-adapter-query
     - name: KAFKA_BOOTSTRAP_SERVERS
          value: demo-dtm-kz01.ru-central1.internal:9092
     - name: LLR_ROWS_LIMIT
          value: "1000"
     - name: LOGBACK_PARAM
          value: --logging.config=/app/fluent-bit/logback.xml
     - name: PFS_HOST
          value: pf.k8s.ru
     - name: PFS_PORT
          value: "80"
     - name: PF_REQUEST_LOG_ENABLED
          value: "true"
     - name: PF_RESPONSE_LOG_ENABLED
          value: "true"
```

```
    - name: QUERY_REQUEST_LOG_ENABLED
      value: "true"
    - name: QUERY_RESPONSE_LOG_ENABLED
      value: "true"
    - name: TZ
      value: Europe/Moscow
    - name: VERTICLE_QUERY_REQUEST_INSTANCES
      value: "1"
    - name: VERTX_DTMPOOL
      value: "10"
    - name: VERTX_POOL_EVENTLOOPPOOL
      value: "10"
    - name: VERTX_POOL_QUERYPOOL
      value: "20"
    - name: VERTX_POOL_WORKERPOOL
      value: "10"
    - name: ZOOKEEPER_DS_ADDRESS
      value: demo-dtm-kz01.ru-central1.internal:2181
    - name: ZOOKEEPER_HOSTS
      value: demo-dtm-kz01.ru-central1.internal
    - name: ZOOKEEPER_PORT
      value: "2181"
  image: cr.yandex/crpfi51tpl7q2b98nn66/podd-adapter-query:5.1.10-develop-43
  imagePullPolicy: IfNotPresent
  livenessProbe:
    failureThreshold: 3
    httpGet:
    path: /version
    port: http
    scheme: HTTP
    initialDelaySeconds: 5
    periodSeconds: 10
    successThreshold: 1
    timeoutSeconds: 1
  name: podd-adapter-query
  ports:
  - containerPort: 8083
    name: http
    protocol: TCP
  - containerPort: 9837
    name: metrics
    protocol: TCP
  readinessProbe:
    failureThreshold: 3
    httpGet:
    path: /version
    port: http
    scheme: HTTP
    initialDelaySeconds: 5
    periodSeconds: 10
    successThreshold: 1
    timeoutSeconds: 1
  resources:
    limits:
    cpu: "2"
    memory: 3Gi
    requests:
    cpu: "1"
    memory: 1Gi
  securityContext: {}
  terminationMessagePath: /dev/termination-log
  terminationMessagePolicy: File
```

```
    volumeMounts:
    # Директория хранения логов приложения
    - mountPath: /fluent-bit/logs/
        name: fluent-bit-logs
    # Директория хранения logback файла
    - mountPath: /app/fluent-bit/
        name: fluent-bit-logback
    # Контейнер для сбора и передачи логов
    - env:
    - name: DEPLOYMENTUNIT
        value: podd-adapter-query
    image: fluent/fluent-bit:1.9.6
    imagePullPolicy: IfNotPresent
    name: fluent-bit
    resources: {}
    securityContext: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    # Директория хранения логов приложения
    - mountPath: /fluent-bit/logs/
        name: fluent-bit-logs
    # Настройки fluentbit
    - mountPath: /fluent-bit/etc/
        name: fluent-bit-config
    dnsPolicy: ClusterFirst
    imagePullSecrets:
    - name: ycr
    restartPolicy: Always
    schedulerName: default-scheduler
    securityContext: {}
    serviceAccount: default
    serviceAccountName: default
    terminationGracePeriodSeconds: 30
    volumes:
    - configMap:
        defaultMode: 420
        items:
        - key: parsers.conf
        path: parsers.conf
        - key: fluent-bit.conf
        path: fluent-bit.conf
        - key: scripts.lua
        path: scripts.lua
        name: fluent-bit-config-json-demo
    name: fluent-bit-config
    - configMap:
        defaultMode: 420
        items:
        - key: logback.xml
        path: logback.xml
        name: fluent-bit-logback-json-demo
    name: fluent-bit-logback
    - emptyDir: {}
    name: fluent-bit-logs
```

Пример создания файла `service`

```
apiVersion: v1
kind: Service
metadata:
labels:
app.kubernetes.io/instance: podd-adapter-query
app.kubernetes.io/name: podd-adapter-query
name: podd-adapter-query
spec:
ports:
- name: http
port: 8083
protocol: TCP
targetPort: http
selector:
app.kubernetes.io/instance: podd-adapter-query
app.kubernetes.io/name: podd-adapter-query
sessionAffinity: None
type: ClusterIP
```

Пример создания файла `configmap`

```
# В STDOUT выводит в JSON формате с полями подходящими для ГОСТЕХ
# В FILE_FLUENT выводит JSON формате с полями для внутреннего пользования стенда разработки и тестирования
apiVersion: v1
kind: ConfigMap
metadata:
name: fluent-bit-logback-json-demo
namespace: default
data:
logback.xml: |
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <property name="serviceName" value="${K8S_SERVICE_NAME:-}"/>
    <property name="instanceID" value="${HOSTNAME:-}"/>
    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
    <encoder class="net.logstash.logback.encoder.LogstashEncoder">
        <includeContext>false</includeContext>
        <includeTags>true</includeTags>
        <includeMdc>true</includeMdc>
        <mdcKeyFieldName>requestId=traceId</mdcKeyFieldName>
        <fieldNames>
        <logger>className</logger>
        <timestamp>dateTime</timestamp>
        <level>logLevel</level>
        <stackTrace>stackTrace</stackTrace>
        <thread>threadName</thread>
        <version>[ignore]</version>
        <levelValue>[ignore]</levelValue>
        </fieldNames>
        <customFields>{"instanceID": "${instanceID}", "serviceName":
"${serviceName}"}</customFields>
    </encoder>
    </appender>
    <appender name="FILE_FLUENT"
class="ch.qos.logback.core.rolling.RollingFileAppender">
    <file>/fluent-bit/logs/log.log</file>
    <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
        <fileNamePattern>/fluent-bit/logs/log.%d{yyyy-MM-dd}.log</fileNamePattern>
        <maxHistory>1</maxHistory>
        <totalSizeCap>1GB</totalSizeCap>
    </rollingPolicy>
```

```
    <append>false</append>
    <encoder class="net.logstash.logback.encoder.LogstashEncoder">
        <includeContext>false</includeContext>
        <includeTags>true</includeTags>
        <includeMdc>true</includeMdc>
        <fieldNames>
        <version>[ignore]</version>
        <levelValue>[ignore]</levelValue>
        </fieldNames>
    </encoder>
    </appender>
    <root level="info" additivity="false">
    <appender-ref ref="STDOUT"/>
    <appender-ref ref="FILE_FLUENT"/>
    </root>
</configuration>
```

Пример создания файла `configmap` для Fluentbit

```
apiVersion: v1
kind: ConfigMap
metadata:
name: fluent-bit-config-json-demo
data:
fluent-bit.conf: |
[SERVICE]
    Flush        1
    Log_Level    info
    Daemon       off
    Parsers_File /fluent-bit/etc/parsers.conf
[INPUT]
    Name             tail
    Path             /fluent-bit/logs/log.log
    Tag              services
    Buffer_Chunk_Size 400k
    Buffer_Max_Size   6MB
    Mem_Buf_Limit     6MB
    Parser           docker
    Refresh_Interval  20
[FILTER]
    Name   record_modifier
    Match  *
    Record hostname "${HOSTNAME}"
    Record serviceName "${DEPLOYMENTUNIT}"
[OUTPUT]
    Name  forward
    Match *
    host  demo-dtm-vector01.ru-central1.internal
    port  24228
parsers.conf: |
[PARSER]
    Name        docker
    Format      json
    Key_Name    log
    Time_Key    @timestamp
scripts.lua: ""
```

5.1.6.2 Примеры инструкций по развертыванию Prostore в Kubernetes

Пример создания файла `deployment`

```
apiVersion: apps/v1
kind: Deployment
```

```yaml
metadata:
  name: prostore
  namespace: dtm-dev
  uid: 46d6e239-427e-4dad-a988-4ce44a53b75e
  resourceVersion: '630322324'
  generation: 39
  creationTimestamp: '2023-03-03T07:36:03Z'
  labels:
    app.kubernetes.io/instance: prostore
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/name: prostore
    app.kubernetes.io/version: 6.7.0
    helm.sh/chart: prostore-0.2.0
    k8slens-edit-resource-version: v1
  annotations:
    deployment.kubernetes.io/revision: '35'
    helm.sh/template: 1.0.1
    meta.helm.sh/release-name: prostore
    meta.helm.sh/release-namespace: dtm-dev
  selfLink: /apis/apps/v1/namespaces/dtm-dev/deployments/prostore
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/instance: prostore
      app.kubernetes.io/name: prostore
  template:
    metadata:
      creationTimestamp: null
      labels:
        app.kubernetes.io/instance: prostore
        app.kubernetes.io/name: prostore
      annotations:
        checksum/config:
2d5e69a4edfcbaf92ee27d05855c797f1a825c16c08d78b82db62da024cc7b1d
        helm.sh/template: 1.0.1
        kubectl.kubernetes.io/restartedAt: '2023-11-24T09:25:44Z'
        rollout: QMnGFyw4ilxm
    spec:
      volumes:
        - name: logs-q
          emptyDir: {}
        - name: logs-s
          emptyDir: {}
        - name: logback-q
          configMap:
            name: prostore.config
            items:
              - key: logback-q.xml
                path: logback.xml
            defaultMode: 420
        - name: logback-s
          configMap:
            name: prostore.config
            items:
              - key: logback-s.xml
                path: logback.xml
            defaultMode: 420
        - name: fluent-bit-config-q
          configMap:
            name: prostore.config
            items:
```

```
            - key: parsers.conf
              path: parsers.conf
            - key: fluent-bit-q.conf
              path: fluent-bit.conf
          defaultMode: 420
        - name: fluent-bit-config-s
          configMap:
            name: prostore.config
            items:
              - key: parsers.conf
                path: parsers.conf
              - key: fluent-bit-s.conf
                path: fluent-bit.conf
            defaultMode: 420
      containers:
        - name: prostore
          image: registry.gosuslugi.local/dtm-dev/query-execution:6.8.1
          command:
            - java
            - '-XX:MaxRAMPercentage=80.0'
            - '-jar'
            -  dtm-query-execution-core.jar
          args:
            - '--logging.config=logback.xml'
          ports:
            - name: http-q
              containerPort: 9090
              protocol: TCP
            - name: metrics-q
              containerPort: 8080
              protocol: TCP
          env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: metadata.name
            - name: POD_NAMESPACE
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: metadata.namespace
            - name: POD_IP
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: status.podIP
            - name: NODE_NAME
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: spec.nodeName
            - name: ADB_HOST
              value: 10.81.0.99
            - name: ADB_MPPW_DEFAULT_MESSAGE_LIMIT
              value: '1000'
            - name: ADB_MPPW_FDW_TIMEOUT_MS
              value: '2000'
            - name: ADB_MPPW_USE_ADVANCED_CONNECTOR
              value: 'true'
            - name: ADB_NAME
            - name: ADB_PASS
```

```
            value: dtm
          - name: ADB_USERNAME
            value: dtm
          - name: ADP_HOST
            value: postgres
          - name: ADP_PASS
            value: dtm
          - name: ADP_PORT
            value: '5432'
          - name: ADP_USERNAME
            value: dtm
          - name: ADP_MAX_POOL_SIZE
            value: '4'
          - name: KAFKA_JET_POLL_DURATION_MS
            value: '1000'
          - name: KAFKA_JET_POLL_BUFFER_SIZE
            value: '1000'
          - name: KAFKA_JET_DB_BUFFER_SIZE
            value: '3000'
          - name: ADP_EXECUTORS_COUNT
            value: '4'
          - name: ADP_REST_START_LOAD_URL
            value: http://kafka-postgres-writer:8096/newdata/start
          - name: ADP_REST_STOP_LOAD_URL
            value: http://kafka-postgres-writer:8096/newdata/stop
          - name: ADP_MPPW_CONNECTOR_VERSION_URL
            value: http://kafka-postgres-writer:8096/versions
          - name: ADP_MPPR_QUERY_URL
            value: http://kafka-postgres-reader:8094/query
          - name: ADP_MPPR_CONNECTOR_VERSION_URL
            value: http://kafka-postgres-reader:8094/versions
          - name: CORE_PLUGINS_ACTIVE
            value: ADP
          - name: DTM_NAME
            value: dev
          - name: EDML_CHANGE_OFFSET_TIMEOUT_MS
            value: '180000'
          - name: EDML_DATASOURCE
            value: ADP
          - name: EDML_DEFAULT_CHUNK_SIZE
            value: '500'
          - name: EDML_FIRST_OFFSET_TIMEOUT_MS
            value: '180000'
          - name: KAFKA_BOOTSTRAP_SERVERS
            value: kafka-0.kafka-headless:9092
          - name: KAFKA_JET_WRITERS
            value: http://kafka-jet-writer:8080
          - name: KAFKA_STATUS_EVENT_ENABLED
            value: 'true'
          - name: KAFKA_STATUS_EVENT_TOPIC
            value: status.event
          - name: KAFKA_STATUS_EVENT_WRITE_OPERATIONS_ENABLED
            value: 'true'
          - name: >-
              LOGGING_LEVEL_RU_DATAMART_PROSTORE_QUERY_EXECUTION_CORE_BASE_SERVICE
            value: warn
          - name: TZ
            value: Europe/Moscow
          - name: ZOOKEEPER_DS_ADDRESS
            value: zookeeper-0.zookeeper-headless:2181
          - name: ZOOKEEPER_KAFKA_ADDRESS
            value: zookeeper-0.zookeeper-headless:2181
```

```
        resources:
          limits:
            cpu: '1'
            memory: 4Gi
          requests:
            cpu: 125m
          memory: 128Mi
        volumeMounts:
          - name: logs-q
            mountPath: /app/logs
          - name: logback-q
            mountPath: /app/logback.xml
            subPath: logback.xml
        livenessProbe:
          httpGet:
            path: /actuator/health
            port: metrics-q
            scheme: HTTP
          initialDelaySeconds: 20
          timeoutSeconds: 5
          periodSeconds: 10
          successThreshold: 1
          failureThreshold: 3
        readinessProbe:
          httpGet:
            path: /actuator/health
            port: metrics-q
            scheme: HTTP
          initialDelaySeconds: 20
          timeoutSeconds: 5
          periodSeconds: 10
          successThreshold: 1
          failureThreshold: 3
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
        imagePullPolicy: Always
      - name: fluent-bit-q
        image: registry.gosuslugi.local/proxy-docker.io/fluent/fluent-bit:1.9.6
        env:
          - name: POD_NAME
            valueFrom:
              fieldRef:
                apiVersion: v1
                fieldPath: metadata.name
          - name: POD_NAMESPACE
            valueFrom:
              fieldRef:
                apiVersion: v1
                fieldPath: metadata.namespace
          - name: POD_IP
            valueFrom:
              fieldRef:
                apiVersion: v1
                fieldPath: status.podIP
          - name: NODE_NAME
            valueFrom:
              fieldRef:
                apiVersion: v1
                fieldPath: spec.nodeName
        resources:
          limits:
            cpu: 100m
```

```
            memory: 256Mi
          requests:
            cpu: 100m
            memory: 256Mi
        volumeMounts:
          - name: logs-q
            mountPath: /app/logs
          - name: fluent-bit-config-q
            mountPath: /fluent-bit/etc/
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
        imagePullPolicy: IfNotPresent
    restartPolicy: Always
    terminationGracePeriodSeconds: 30
    dnsPolicy: ClusterFirst
    securityContext: {}
    imagePullSecrets:
      - name: registry.gosuslugi.local
    schedulerName: default-scheduler
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 25%
      maxSurge: 25%
  revisionHistoryLimit: 10
  progressDeadlineSeconds: 600
```

Пример создания файла `service`

```
apiVersion: v1
kind: Service
metadata:
name: prostore
spec:
ports:
- name: jdbc
port: 9090
protocol: TCP
targetPort: jdbc
selector:
app.kubernetes.io/instance: prostore
app.kubernetes.io/name: prostore
sessionAffinity: None
type: ClusterIP
```

Пример создания файла `configmap`

```
# В STDOUT выводит в простом "читаемом" формате
# В FILE_FLUENT выводит в logfmt формате с полями для внутреннего пользования стенда разработки и тестирования
apiVersion: v1
kind: ConfigMap
metadata:
name: fluent-bit-logback
data:
logback.xml: |
<configuration>
    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
    <layout class="ch.qos.logback.classic.PatternLayout">
        <pattern>
        <Pattern>
        %d{yyyy-MM-dd HH:mm:ss.SSS} %-5level %logger{36} - %msg%n
        </Pattern>
```

```
          </pattern>
     </layout>
     </appender>
     <appender name="FILE_FLUENT"
class="ch.qos.logback.core.rolling.RollingFileAppender">
     <file>/fluent-bit/logs/log.log</file>
     <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
          <fileNamePattern>/fluent-bit/logs/log.%d{yyyy-MM-dd}.log</fileNamePattern>
          <maxHistory>1</maxHistory>
          <totalSizeCap>1GB</totalSizeCap>
     </rollingPolicy>
     <append>false</append>
     <layout class="ch.qos.logback.classic.PatternLayout">
          <pattern>
          <Pattern>
          @timestamp="%d{yyyy-MM-dd'T'HH:mm:ss.SSSXXX, UTC}" level=%level
threadName="%thread" logger="%logger"
message="%replace(%replace(%m){'¥n','¥¥n'}){'¥"','¥''}"
exception="%replace(%replace(%ex){'¥"','¥''}){'¥n','¥¥n'}%nopex" ¥n
          </Pattern>
          </pattern>
     </layout>
     </appender>
     <root level="debug" additivity="false">
     <appender-ref ref="STDOUT"/>
     <appender-ref ref="FILE_FLUENT"/>
     </root>
</configuration>
```

Пример создания файла `configmap` для Fluentbit

```
apiVersion: v1
kind: ConfigMap
metadata:
name: fluent-bit-config-demo
data:
fluent-bit.conf: |
[SERVICE]
     Flush       1
     Log_Level    info
     Daemon       off
     Parsers_File /fluent-bit/etc/parsers.conf
[INPUT]
     Name             tail
     Path             /fluent-bit/logs/log.log
     Tag              services
     Buffer_Chunk_Size 400k
     Buffer_Max_Size   6MB
     Mem_Buf_Limit     6MB
     Parser           logfmt
     Refresh_Interval  20
[FILTER]
     Name   record_modifier
     Match  *
     Record hostname "${HOSTNAME}"
     Record serviceName "${DEPLOYMENTUNIT}"
[OUTPUT]
     Name  forward
     Match *
     host  demo-dtm-vector01.ru-central1.internal
     port  24228
parsers.conf: |
```

```
[PARSER]
    Name        logfmt
    Format      logfmt
scripts.lua: ""
```

5.2 Дополнительные возможности конфигурации Лайт

Необходимость выполнения действий данного раздела определяется в процессе эксплуатации программы.

5.2.1 Логирование

Сбор лог-файлов программы, с записями о событиях производится с помощью *Graylog*, через утилиту полнотекстового поиска и аналитики **Elasticsearch**, которая позволяет в режиме реального времени хранить, искать и анализировать большие объемы данных.

При запуске **Graylog** автоматически конфигурирует **Elasticsearch**.

Для передачи сообщений в **Graylog** используется **Filebeat**.

Просмотр записей лог-файлов доступен через web-интерфейс *Graylog* (см. Рисунок - 5.15) по адресу http://0.0.0.0:9010/ (авторизация: admin/somepasswordpepper).

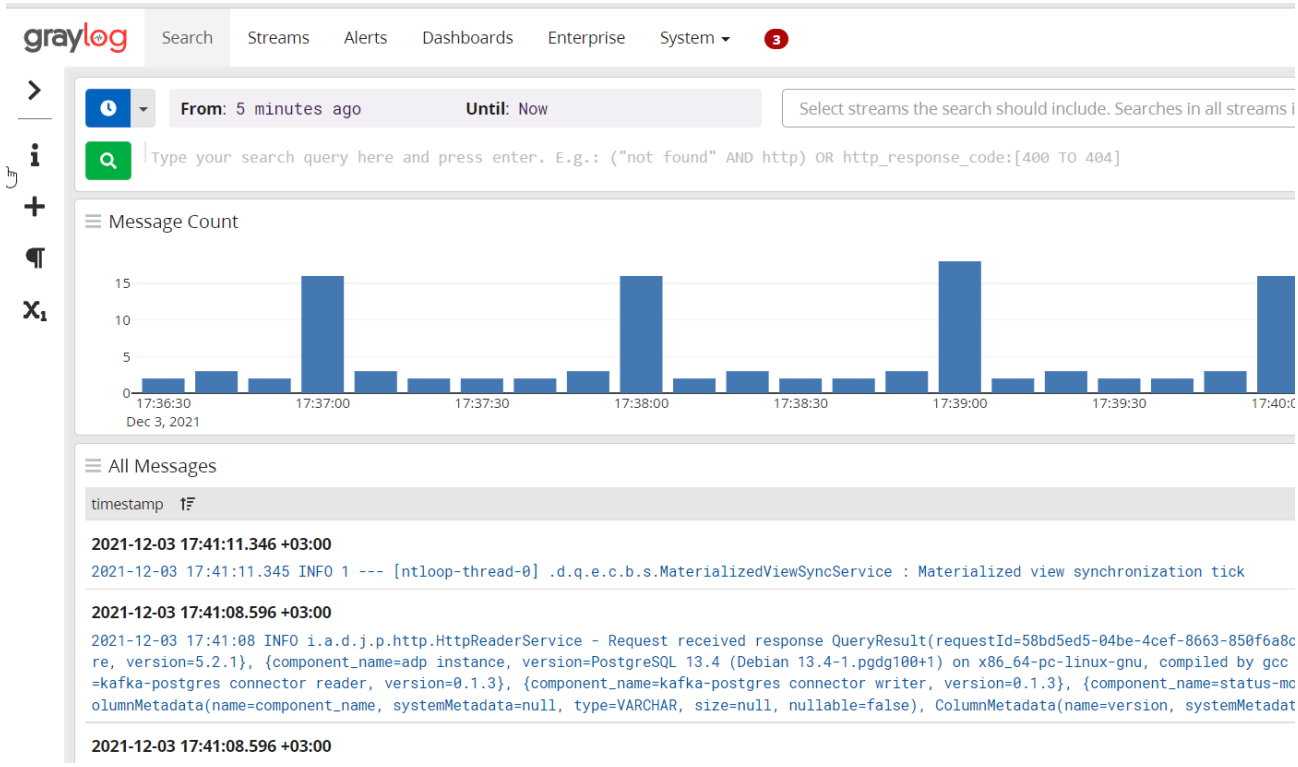

Рисунок - 5.15 Просмотр записей лог-файлов в Graylog

Каждая запись в таблице содержит следующую информацию:

1. Уровень логирования;
2. Дата и время события в формате `yyyy-mm-dd hh:mm:ss`;
3. Имя узла, на котором произошло событие.

5.2.2 Проверка версии компонентов

Версии используемых компонентов программы можно проверить с помощью запроса `CHECK_VERSIONS`.

6 СООБЩЕНИЯ АДМИНИСТРАТОРУ

6.1 Сообщения в ходе установки и настройки программы

Сообщения в ходе выполнения настройки конфигурации Стандарт

Выполнение настройки программы представляет собой процесс ручного или автоматизированного формирования конфигурационных файлов программы, в частности указания сетевых адресов и идентификаторов компонентов для взаимосвязи между ними, задания путей на дисковых пространствах для обработки полезных и служебных данных, а также метаданных.

Внесенные в конфигурацию изменения влияют на результаты выполнения новой проверки программы.

Поток сообщений о ходе выполнения настройки является частным случаем потока сообщений при выполнении проверки программы.

Компоненты программы в ходе выполнения настройки формируют сообщения и выводят их в стандартный порт вывода, перенаправленный в соответствующие лог-файлы, размещенные по указанным адресам.

Сообщения в ходе установки конфигурации Лайт с помощью Ansible

Выполнение установки и настройки программы представляет собой процесс автоматизированного формирования конфигурационных файлов программы с помощью Ansible, в частности указания сетевых адресов и идентификаторов компонентов для взаимосвязи между ними, задания путей на дисковых пространствах для обработки полезных и служебных данных, а также метаданных.

Описание типичных ошибок при работе **Ansible** можно просмотреть на официальном сайте разработчика приложения.

Внесенные изменения в дистрибутив приложения и конфигурационные файлы влияют на результаты установки и работы программы.

Компоненты программы в ходе выполнения настройки формируют сообщения и выводят их в стандартный порт вывода, перенаправленный в соответствующие лог-файлы. Просмотреть лог-файлы можно с помощью приложения Grafana.

6.2 Сообщения при эксплуатации программы

В ходе эксплуатации компоненты программы формируют сообщения и выводят их в стандартный порт вывода, перенаправленный в соответствующие лог-файлы. Генерация сообщений администратору в ходе эксплуатации программы подчиняются следующей блок-схеме (см. Рисунок - 6.1).

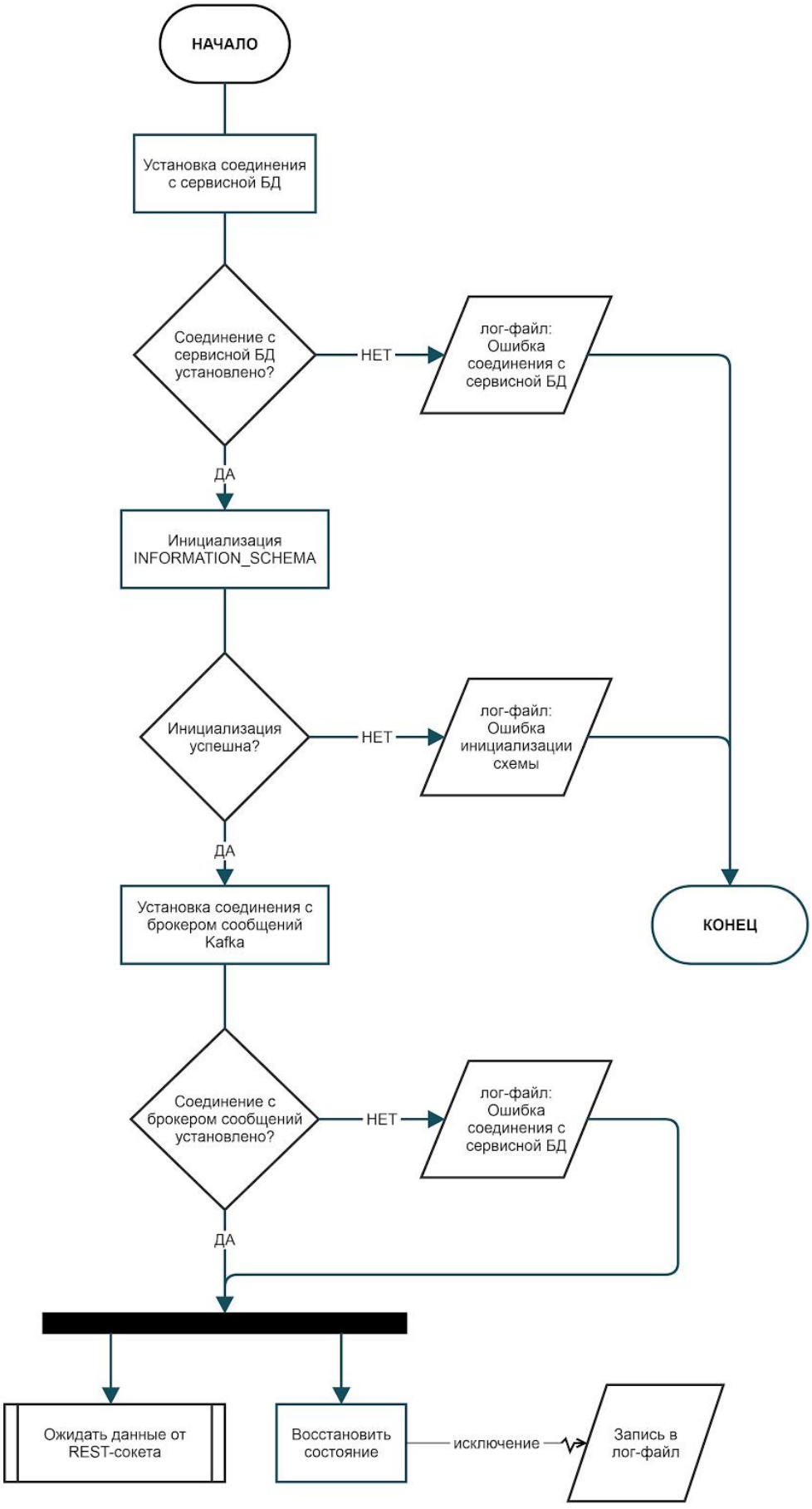

Рисунок - 6.1 Блок-схема журналирования сообщений в лог-файлы при запуске программы

Дополнительно, система может формировать следующие сообщения:

Таблица 6.1 Пример сообщений

Сообщение	**Описание**
DATAMART-17473	Запрос не прошел валидацию. Если в запросе тип данных параметра не поддерживается и/или формат значения недопустимый и/или набор параметров (или их значения, или их сочетание) некорректные.
DATAMART-17001	Внутренняя ошибка Витрины, возникает в процессе генерации файлов (в случае успешного считывания параметров).

7 МЕТРИКИ В МОДУЛЯХ КОМПОНЕНТА «ВИТРИНА ДАННЫХ»

Для обеспечения унифицированного мониторинга, а так же, для обеспечения совместимости с платформой Гостех формируются метрики всеми модулями Компонента «Витрина данных».

Для каждого функционального блока модулей Компонента «Витрина данных» реализованы три метрики запросов:

- число запросов в секунду для каждого функционального блока `{worker}_req_count_total`;
- время обработки запроса для каждого функционального блока `{worker}_req_time_seconds_sum`/ `{worker}_req_time_seconds_count`;
- число ошибок в единицу времени для каждого функционального блока `{worker}_req_time_err_total`;

где `{worker}` - название функционального блока.

Для СМЭВ4-адаптера - Модуля исполнения запросов (podd-adapter-query) реализованы метрики для функций:

- query: LLR/MPPR запросы;
- report: запросы к сервису формирования документов;
- metadata new data: запрос создания скрипта генерации таблицы;
- metadata:запрос метаданных;
- statistics: запрос статистики;
- cancel query: отмены запросов.

Для СМЭВ4-адаптера - Модуля MPPR (podd-adapter-mppr) реализованы метрики для функций:

- mppr delta: передача данных новой дельты для репликации;
- mppr query: отправка результата выполнения запроса.

Для СМЭВ4-адаптера - Модуля MPPW (podd-adapter-mppw) реализованы метрики для функций:

- mppw delta in: обработка команды на загрузку данных по подписке;.
- mppw tp: обработка запроса с табличными параметрами;
- mppw upload rq: обработка запроса загрузки большого объема данных.

Для СМЭВ4-адаптера - Модуля дефрагментации чанков табличных параметров (podd-avro-defragmentator) реализованы метрики для функций:

- assemble service verticle: обработка чанка модулем;
- kafka input verticle: получение сообщения из топика query.tp.bin;
- kafka output verticle: отправка сообщения в топика query.tp;
- message store verticle: отправка чанка в хранилище.

Для Модуля загрузки CSV-файлов (csv-uploader) реализованы метрики для функций:

- delete from table: удаление данных (1 файл);
- get data by id: получение данных по идентификатору;
- get table data: получение данных таблицы;
- upload rest-uploader: добавление/изменение данных (несколько файлов) через rest-uploader;
- upsert multipart: добавление/изменение данных (несколько файлов) через csv-

uploader;

- upsert to table: обновление данных таблицы;
- delete multipart: удаление данных (несколько файлов).

Для Модуля исполнения асинхронных заданий (data-uploader) реализованы метрики для функций:

- handle mppw answer: обработка ответов от модуля MPPW;
- send mppw tasks: отправка заданий на загрузку чанков по MPPW;
- upload files: загрузка файлов с данными;
- prepare chunks: создание чанков на основе файлов с данными.

Для Модуля асинхронной загрузки данных из сторонних источников (rest-uploader) реализованы метрики для функций:

- upload data: загрузка данных;
- delete data: удаление данных.

Для Blob-Адаптера реализованы метрики для функций:

- blob rq: запрос для выгрузки данных по ссылке через Kafka;
- blob web: запрос для выгрузки данных по ссылке через REST (POST): по _/download.

Для Сервиса формирования документов (printable-form-service) реализованы метрики для функции:

- report: обработка запроса на получение документа.

Для СМЭВ3-адаптера реализованы метрики для функции:

- request process: обработка запросов из очереди СМЭВ.

Для СМЭВ4-адаптера - Модуля импорта табличных параметров (podd-adapter-import-tp) реализованы метрики для функций:

- cancel rq: отмена выполнения запроса;
- mppw rs: обработка результата запроса с табличными параметрами;
- tp delete tmp: удаление временных таблиц после выполнения запроса;
- tp upload delta: обработка запроса на пересечение данных дельт;
- tp upload delta in: обработка запроса на получение чанков распределенных подписок;
- tp upload query: обработка запроса с табличными параметрами.

Для СМЭВ4-адаптера - Модуля группировки табличных параметров (podd-adapter-group-tp) реализованы метрики для функции:

- delta in rq: группировка поступающих пакетов каждого табличного параметра в отдельные топики.

Для СМЭВ4-адаптера - Модуля подписок (podd-adapter-replicator) реализованы метрики для функций:

- delta apply: обработка результата применения дельт;
- delta request: обработка запроса на загрузку дельты;
- replication cancel: обработка запроса на отмену подписки;
- replication register: обработка запроса на регистрацию подписки;
- replication register in: обработка запроса на создание таблиц для подписки;
- status event: отправка статуса событий;
- subscription consumer cancel: отмена подписки для конкретного потребителя.

Для СМЭВ4-адаптера - Модуля группировки чанков репликации (podd-adapter-group-repl) реализованы метрики для функции:

– delta in rq: обработка запроса с фрагментом дельты для репликации.

Для Сервиса генерации уникального номера (counter-provider) реализованы метрики для функции:

– get counter: обработка запроса на получение номера.

Таблица 7.1 Метрики в модулях Компонента «Витрина данных»

Модуль	**Сервис**	**Функции**	**Метрика**	**Насыщение ресурсов**
СМЭВ4-адаптер - Модуль исполнения запросов	podd-adapter-query	cancel query; metadata; metadata new data; query; report; statistics.	`req_count_total` - число запросов по функциям; `req_time_seconds_sum` - время обработки запроса для каждого функционального блока; `req_time_seconds_sum`/ `req_count_total` – среднее время обработки.	1. `process_cpu_usage` - использование процессора; 2. `jvm_memory_used_bytes` - использование памяти: `jvm_memory_used_bytes{area="heap",id="PS Survivor Space",}`; `jvm_memory_used_bytes{area="heap",id="PS Old Gen",}`; `jvm_memory_used_bytes{area="heap",id="PS Eden Space",}`; `jvm_memory_used_bytes{area="nonheap",id="Metaspace",}`; `jvm_memory_used_bytes{area="nonheap",id="Code Cache",}`; `jvm_memory_used_bytes{area="nonheap",id="Compressed Class Space",}`; `jvm_buffer_memory_used_bytes{id="direct",}`; `jvm_buffer_memory_used_bytes{id="mapped",}`.
СМЭВ4-адаптер - Модуль MPPR	podd-adapter-mppr	mppr delta; mppr qiery.		
СМЭВ4-адаптер - Модуль MPPW	podd-adapter-mppw	mppw delta in; mppw tp; mppw upload rq.		
СМЭВ4-адаптер - Модуль дефрагментации чанков табличных параметров	podd-avro-defragmentator	assemble service verticle; kafka input verticle; kafka output verticle; message store verticle.		
Модуль загрузки CSV-файлов	csv-uploader	delete from table; delete multipart; get data by id; get table data; upload rest-uploader; upsert multipart; upsert to table; delete multipart.		
Модуль исполнения асинхронных заданий	data-uploader	handle mppw answer; send mppw tasks; upload files; prepare chunks.		
Модуль асинхронной загрузки данных из сторонних источников	rest-uploader	delete data; upload data.		
Blob-Адаптер	blob-adapter	blob rq; blob web.		

Сервис формирования документов	printable-form-service	report.		
СМЭВ3-адаптер	smev3-adapter	request process.		
Модуль импорта данных табличных параметров	podd-adapter-import-tp	cancel rq; mppw rs; tp delete tmp; tp upload delta; tp upload delta in; tp upload query;		
Модуль группировки данных табличных параметров	podd-adapter-group-tp	delta in rq.		
Модуль подписок	podd-adapter-replicator	delta apply; delta request; replication cancel; replication register; replication register in; status event; subscription consumer cancel;		
Модуль группировки чанков репликации	podd-adapter-group-repl	delta in rq.		
Сервис генерации уникального номера	counter-provider	get counter.		

ПРИЛОЖЕНИЕ 1. ОПИСАНИЕ СПЕЦИФИКАЦИИ

1 Спецификация Модуля исполнения запросов

1.1 Запрос данных из Витрины

Данная спецификация описывает возможность запроса данных к Витрине, получения успешного ответа на запрос или ошибки, в случае невозможности выполнения запроса, с описанием причины ошибки.

1.1.1 query.rq

`query.rq` - Топик sql запросов на исполнение

Структура сообщения

```yaml
datamartExecuteQueryRequestMessage:
  description: Исполнение sql запроса на витрине
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  bindings:
    kafka:
      key:
        type: string
        format: uuid
        description: Уникальный идентификатор подзапроса
  headers:
    type: object
    properties:
      MESSAGE_TYPE:
        description: Тип сообщения
        type: string
        const: DatamartExecuteQueryRequest:0.1
      REQUEST_ID:
        description: Идентификатор запроса
        type: string
      QUERY_DEADLINE:
        description: Время в миллисекундах от эпохи, до которого запрос должен быть выполнен
        type: string
        format: int64
      AGENT_CONSUMER_ID:
        description: Мнемоника потребителя (мнемоника агента)
        type: string
      QUERY_MNEMONIC:
        description: '<Полная мнемоника РЗ>.<версия РЗ>'
        type: string
  payload:
    $ref: '#/components/schemas/datamartExecuteQueryRequest'
  examples:
    - name: simple
      summary: Простой запрос на исполнение без параметров
      headers:
        MESSAGE_TYPE: DatamartExecuteQueryRequest:0.1
      payload:
        requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
        subRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
        replyTo: agent-fias
        datamartMnemonic: fias
        sql: select * from v1_addrobj
        parameters: [ ]
        namedParams: [ ]
```

```
      tableParams: [ ]
      isForEstimation: false
      rowCountThreshold: -1
      customerId: aaa
      customerOgrn: ""
      queryMnemonic: fias.selectAllAddrobj.1.0
  - name: estimation
    summary: Запрос на оценку
    headers:
      MESSAGE_TYPE: DatamartExecuteQueryRequest:0.1
    payload:
      requestId: 403eada5-05f6-480c-bca9-03328091efeb
      subRequestId: 451000b8-dff2-4a1b-ab1b-42500a70d232
      replyTo: agent-fias
      datamartMnemonic: fias
      sql: select * from v1_addrobj
      parameters: [ ]
      namedParams: [ ]
      tableParams: [ ]
      isForEstimation: true
      rowCountThreshold: 1000
      customerId:
        string: agent-fias
      customerOgrn:
        string: "1053600591197"
      queryMnemonic:
        string: fias.selectAllAddrobj.1.0
  - name: complex
    summary: Запрос с параметрами
    headers:
      MESSAGE_TYPE: DatamartExecuteQueryRequest:0.1
    payload:
      requestId: 68758a92-0027-4258-bf17-aa3d24f85094
      subRequestId: 96e6eb99-7ff1-4efa-abae-ef1c5744b723
      replyTo: agent-fias
      datamartMnemonic: fias
      sql: select * from v1_addrobj where oktmo = ? and name = @tbl.fullname
      parameters:
        - type: STRING
          value:
            string: asdasdasd
        - type: LONG
          value: null
      namedParams: [ ]
      tableParams: [ ]
      isForEstimation: false
      rowCountThreshold: -1
      customerId:
        string: agent-fias
      customerOgrn:
        string: "1053600591197"
      queryMnemonic:
        string: fias.selectAddrobjWithParams.1.0
  - name: complex_named
    summary: Запрос с именованными параметрами
    headers:
      MESSAGE_TYPE: DatamartExecuteQueryRequest:0.1
    payload:
      requestId: 12358a92-0027-4258-bf17-aa3d24f85094
      subRequestId: 56e6eb99-7ff1-4efa-abae-ef1c5744b723
      replyTo: agent-fias
      datamartMnemonic: fias
```

```
      sql: select * from @tbl.fullname el LEFT JOIN v1_addrobj where oktmo = @p1 and
kod = @p2
      parameters: [ ]
      namedParams:
        - name: p1
          type: STRING
          value:
            string: asdasdasd
        - name: p2
          type: LONG
          value: null
      tableParams: [ ]
      isForEstimation: false
      rowCountThreshold: -1
      customerId:
        string: agent-fias
      customerOgrn:
        string: "1053600591197"
      queryMnemonic:
        string: fias.selectAddrobjWithParams.1.0
  - name: deadline
    summary: Простой запрос на исполнение без параметров
    headers:
      MESSAGE_TYPE: DatamartExecuteQueryRequest:0.1
      QUERY_DEADLINE: 1629289006904
    payload:
      requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
      subRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
      replyTo: agent-fias
      datamartMnemonic: fias
      sql: select * from v1_addrobj
      parameters: [ ]
      namedParams: [ ]
      tableParams: [ ]
      isForEstimation: false
      rowCountThreshold: -1
      customerId: agent-fias
      customerOgrn: "1053600591197"
      queryMnemonic: fias.selectAllWithDeadline.1.0
```

Avro-схема сообщения

```
datamartExecuteQueryRequest:
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  type: record
  name: QueryRequest
  namespace: datamart.query
  fields:
    - name: requestId
      description: Уникальный идентификатор запроса
      type:
        type: string
        logicalType: uuid
    - name: subRequestId
      description: Уникальный идентификатор подзапроса
      type:
        type: string
        logicalType: uuid
    - name: replyTo
      description: Служебная информация маршрутизации сообщения. Ответ, формируемый
витриной, обязан содержать переданное значение без каких либо искажений
      type: string
```

```
    - name: datamartMnemonic
      description: Мнемоника витрины, к которой выполняется запрос
      type: string
    - name: sql
      description: SQL запрос на исполнение, либо имя хранимой процедуры для
регламентированных запросов
      type: string
    - name: parameters
      description: Параметры к SQL запросу
      default: [ ]
      type:
        type: array
        items:
          type: record
          name: QueryParameter
          description: Описание параметра
          fields:
            - name: type
              type: string
              description: Тип параметра
              enum:
                - BIG_DECIMAL
                - BINARY
                - BOOLEAN
                - DATE
                - DOUBLE
                - FLOAT
                - INTEGER
                - LONG
                - SHORT
                - STRING
                - TIME
                - TIMESTAMP
            - name: value
              description: Значение параметра
              type:
                - string
                - 'null'
    - name: namedParams
      description: Именованные параметры запроса
      default: [ ]
      type:
        type: array
        items:
          type: record
          name: NamedParam
          description: Описание именованного параметра
          fields:
            - name: name
              description: Имя (мнемоника) параметра
              type: string
            - name: type
              type: string
              description: Тип параметра
              enum:
                - BIG_DECIMAL
                - BINARY
                - BOOLEAN
                - DATE
                - DOUBLE
                - FLOAT
                - INTEGER
```

```
              - LONG
              - SHORT
              - STRING
              - TIME
              - TIMESTAMP
          - name: value
            description: Значение параметра
            type:
              - string
              - 'null'
    - name: tableParams
      description: Табличные параметры запроса
      default: [ ]
      type:
        type: array
        items:
          type: record
          name: TableParam
          fields:
            - name: id
              description: Уникальный идентификатор
              type:
                type: string
                logicalType: uuid
            - name: name
              description: Имя параметра
              type: string
            - name: columns
              description: Описание колонок таблицы
              type:
                type: array
                items:
                  type: record
                  description: Описание колонки
                  name: TableParamColumnInfo
                  fields:
                    - name: name
                      type: string
                      description: Имя колонки
                    - name: type
                      type: string
                      description: Тип атрибута
                      enum:
                        - BIG_DECIMAL
                        - BINARY
                        - BOOLEAN
                        - DATE
                        - DOUBLE
                        - FLOAT
                        - INTEGER
                        - LONG
                        - SHORT
                        - STRING
                        - TIME
                        - TIMESTAMP
    - name: isForEstimation
      description: Признак необходимости вернуть статистику по запросу в качестве
результата. В случае, если оценка по результату исполнения sql запроса не превышает
rowCountThreshold записей, должен сразу отдаваться результат без отправки оценки в ядро
      type: boolean
      default: false
    - name: rowCountThreshold
```

```
      description: Максимальное оценочное количество строк результата, при превышении которого возвращается статистика по запросу. Если оценка по запросу не превышет данный параметр, витрина сразу возвращает ответ с результатом. Заполняется в случае isForEstimation = true
      type: long
      default: -1
    - name: customerId
      description: Мнемоника ИС Потребителя
      type:
        - 'null'
        - string
      default: null
    - name: customerOgrn
      description: ОГРН ИС Потребителя
      type:
        - 'null'
        - string
      default: null
    - name: queryMnemonic
      description: 'Мнемоника РЗ, сформированная по правилу: <мнемоника витрины>.<мнемоника РЗ>.<версия РЗ> Если запрос распределенный, то формируется по правилу: podd.<мнемоника РЗ>.<версия РЗ>'
      type:
        - 'null'
        - string
      default: null
```

1.1.2 query.rs

`query.rs` - Топик с чанками данных исполнения запросов

Структура сообщения

```
datamartExecuteQueryResultChunkMessage:
  description: Чанк с данными по исполнению запроса
  contentType: 'application/octet-stream'
  bindings:
    kafka:
      key:
        $ref: '#/components/schemas/datamartExecuteQueryResultChunk'
  headers:
    type: object
    properties:
      MESSAGE_TYPE:
        description: Тип сообщения
        type: string
        const: DatamartExecuteQueryResultChunk:0.1
  payload:
    description: Бинарные данные чанка
  examples:
    - name: base64
      headers:
        MESSAGE_TYPE: DatamartExecuteQueryResultChunk:0.1
      payload:
        value: JEEJNodyLO7p1pgsRHG9pEiXeYGvHW4YCl4FgrgBmu5C92iVX1PV2GZdcqsb66bx8sk=
```

Avro-схема сообщения

```
datamartExecuteQueryResultChunk:
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  type: record
  name: QueryResultChunk
  namespace: datamart.query
```

```
  fields:
    - name: requestId
      description: Уникальный идентификатор запроса
      type:
        type: string
        logicalType: uuid
    - name: subRequestId
      description: Уникальный идентификатор подзапроса
      type:
        type: string
        logicalType: uuid
    - name: replyTo
      description: Служебная информация маршрутизации сообщения. Заполняется
соответсвующим значением из запроса
      type: string
    - name: chunkNumber
      description: Номер порции по порядку
      type: int
      minimum: 1
    - name: isLastChunk
      description: Признак последнего сообщения
      type: boolean
    - name: streamNumber
      description: Номер стрима дланных
      minimum: 1
      type:
        - int
        - "null"
    - name: streamTotal
      description: Общее количество стримов
      minimum: 1
      type:
        - int
        - "null"
    - name: isFragmented
      description: Признак присутствия в чанке неполных строк (строк, которые были
разбиты на несколько чанков)
      type: boolean
    - name: uncompressedSize
      description: Признак присутствия в чанке неполных строк (строк, которые были
разбиты на несколько чанков)
      type: int
      minimum: 0
  examples:
    - requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
      subRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
      replyTo: agent-fias
      chunkNumber: 1
      isLastChunk: true
      streamNumber:
        int: 1
      streamTotal:
        int: 1
      isFragmented: false
      uncompressedSize: 10
```

1.1.3 query.err

`query.err` - Топик с ошибками исполнения sql запросов на витрине

Структура сообщения

```
datamartExecuteQueryErrorMessage:
```

```yaml
  description: Ошибка исполнения запроса
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  bindings:
    kafka:
      key:
        type: string
        format: uuid
        description: Уникальный идентификатор подзапроса
  headers:
    type: object
    properties:
      MESSAGE_TYPE:
        description: Тип сообщения
        type: string
        const: DatamartExecuteQueryError:0.1
  payload:
    $ref: '#/components/schemas/datamartExecuteQueryError'
  examples:
    - name: error
      summary: Сообщение с ошибкой исполенния запроса на витрине без header
      payload:
        requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
        subRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
        replyTo: agent-fias
        errorCode: DATAMART-001
        message: Непредвиденная ошибка
    - name: errorWithHeader
      summary: Сообщение с ошибкой исполенния запроса на витрине
      headers:
        MESSAGE_TYPE: DatamartExecuteQueryError:0.1
      payload:
        requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
        subRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
        replyTo: agent-fias
        errorCode: DATAMART-001
        message: Непредвиденная ошибка
```

Avro-схема сообщения

```yaml
datamartExecuteQueryError:
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  type: record
  name: QueryError
  namespace: datamart.query
  fields:
    - name: requestId
      description: Уникальный идентификатор запроса
      type:
        type: string
        logicalType: uuid
    - name: subRequestId
      description: Уникальный идентификатор подзапроса
      type:
        type: string
        logicalType: uuid
    - name: replyTo
      description: Служебная информация маршрутизации сообщения. Заполняется
соответсвующим значением из запроса
      type: string
    - name: errorCode
      description: Код возникшей ошибки
      type: string
```

```
    - name: message
      description: Сообщение с ошибкой исполнения
      type: string
  examples:
    - requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
      subRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
      replyTo: agent-fias
      errorCode: DATAMART-001
      message: Непредвиденная ошибка
```

1.1.4 query.estimation.rs

Топик `QUERY.ESTIMATION.RS` предоставляет возможность произвести предварительную оценку объема получаемых данных при выполнении запроса к Витрине данных, а также, ограничить выгрузку данных в случае, если количество получаемых данных превысит заданное количество строк (параметр `rowCountThreshold`). В этом случае, ответом на запрос будет предварительная оценка объема.

Например, если вам нужна информация из какой-либо таблицы контактов, то, возможно, следует предварительно узнать, какой объем данных вы можете получить на такой запрос т.к ответ может содержать несколько гигабайт информации и выполнение запроса может занять много времени. Вы сможете установить ограничение на получение данных, например, не более 10 контактов из таблицы. В этом случае, если ответом на запрос будет 5 контактов, то Витрина предоставит ответ полностью. Если ответом будет 1000 контактов, то в качестве ответа будет сформирована предварительная оценка такого ответа, а именно, что данный ответ будет содержать 1000 строк и содержать информацию, например, на 15000 байт. Используя топик `query.estimation.rs` можно прогнозировать объем получаемых данных, в соответствии с которыми оптимизировать запросы к Витрине.

В случае использования топика `query.estimation.rs` запрашивается не конечный ответ на запрос, а приблизительная оценка объема (байт) и количество строк в ответе.

> Примечание:
>
> Данное требование не распространяется на механизм подписок Потребителей данных ПОДД.

Алгоритм работы query.estimation.rs

1. Витрина получает запрос query.rq с признаком `isForEstimation` оценивает объем результата по этому запросу (в байтах и количестве строк).
2. Витрина сравнивает результаты оценки объема запроса со значением предельного числа строк в параметре `rowCountThreshold` (топик query.rq).
3. Если значение в оценке меньше, чем предельное значение в `rowCountThreshold`, то Витрина возвращает результат запроса в качестве ответа в топик query.rs.
4. Если значение оценки превышает предельное значение, возвращает предварительную оценку объема в качестве ответа.

Структура сообщения

```
datamartQueryEstimationMessage:
  description: Оценка по запросу
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  bindings:
    kafka:
      key:
        type: string
        format: uuid
        description: Уникальный идентификатор подзапроса
```

```
  payload:
    $ref: '#/components/schemas/datamartQueryEstimation'
  examples:
    - name: estimation
      summary: Сообщение с оценкой по исполнению запроса
      payload:
        requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
        subRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
        estimatedRowCount: 100
        estimatedSize: 1000
        estimatedTime: 50
```

Avro-схема сообщения

```
datamartQueryEstimation:
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  type: record
  name: Estimation
  namespace: datamart.query
  fields:
    - name: requestId
      description: Уникальный идентификатор запроса
      type:
        type: string
        logicalType: uuid
    - name: subRequestId
      description: Уникальный идентификатор подзапроса
      type:
        type: string
        logicalType: uuid
    - name: estimatedRowCount
      description: Оценка количества строк результата выполнения запроса
      type: long
    - name: estimatedSize
      description: Оценка объема результата выполнения запроса, в байтах
      type: long
    - name: estimatedTime
      description: Оценка времени выполнения запроса в миллисекундах
      type: long
  examples:
    - requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
      subRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
      estimatedRowCount: 100
      estimatedSize: 1000
      estimatedTime: 50
```

1.2 Отмена запроса данных

Данная спецификация описывает возможность отмены ранее отправленного запроса к Витрине, получения ответа об успешной отмене запроса или ошибки, с описанием возможной причины.

1.2.1 cancel.rq

`cancel.rq` - Топик с сообщениями об отмене исполнения запроса

Структура сообщения

```
datamartQueryCancellationRequestMessage:
  description: Запрос на отмену исполнения
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  bindings:
    kafka:
```

```
      key:
        type: string
        format: uuid
        description: Уникальный идентификатор подзапроса
  headers:
    type: object
    properties:
      REQUEST_ID:
        description: Идентификатор запроса
        type: string
      AGENT_CONSUMER_ID:
        description: Идентификатор агента потребителя
        type: string
  payload:
    $ref: '#/components/schemas/datamartQueryCancellationRequest'
  examples:
    - name: request
      summary: Пример запроса на отмену
      headers:
        REQUEST_ID: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
        AGENT_CONSUMER_ID: agent-fias
      payload:
        requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
```

Avro-схема сообщения

```
datamartQueryCancellationRequest:
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  type: record
  name: AgentQueryCancellationRequest
  namespace: ru.rtlabs.common.query.cancel
  fields:
    - name: requestId
      description: Уникальный идентификатор запроса
      type:
        type: string
        logicalType: uuid
```

1.2.2 cancel.rs

`cancel.rs` - Топик с ответами на отмену запроса

Структура сообщения

```
datamartCancelQuerySuccessMessage:
  description: Ответ об успешной отмене запроса
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  bindings:
    kafka:
      key:
        type: string
        format: uuid
        description: Уникальный идентификатор подзапроса
  headers:
    type: object
    properties:
      REQUEST_ID:
        description: Идентификатор запроса
        type: string
      AGENT_CONSUMER_ID:
        description: Идентификатор агента потребителя
        type: string
  payload:
```

```
    $ref: '#/components/schemas/datamartCancelQuerySuccess'
  examples:
    - name: success
      summary: Пример запроса на отмену
      headers:
        REQUEST_ID: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
        AGENT_CONSUMER_ID: agent-fias
      payload:
        requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
        isSuccess: true
```

Avro-схема сообщения

```
datamartCancelQuerySuccess:
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  type: record
  name: DatamartCancelQuerySuccess
  namespace: datamart.query
  fields:
    - name: requestId
      description: Уникальный идентификатор запроса
      type:
        type: string
        logicalType: uuid
    - name: isSuccess
      description: Признак успешного выполнения операции
      type: boolean
```

1.2.3 cancel.err

`cancel.err` - Топик с ошибками по отмене запроса

Структура сообщения

```
datamartCancelQueryErrorMessage:
  description: Ответ об успешной отмене запроса
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  bindings:
    kafka:
      key:
        type: string
        format: uuid
        description: Уникальный идентификатор подзапроса
  headers:
    type: object
    properties:
      REQUEST_ID:
        description: Идентификатор запроса
        type: string
      AGENT_CONSUMER_ID:
        description: Идентификатор агента потребителя
        type: string
  payload:
    $ref: '#/components/schemas/datamartCancelQueryError'
  examples:
    - name: success
      summary: Пример запроса на отмену
      headers:
        REQUEST_ID: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
        AGENT_CONSUMER_ID: agent-fias
      payload:
        requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
        errorCode: DATAMART-001
```

```
      message: Непредвиденная ошибка
```

Avro-схема сообщения

```
datamartCancelQueryError:
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  type: record
  name: DatamartCancelQueryError
  namespace: datamart.query.cancel
  fields:
    - name: requestId
      description: Уникальный идентификатор запроса
      type:
        type: string
        logicalType: uuid
    - name: errorCode
      description: Код ошибки выполнения
      type: string
    - name: message
      description: Сообщение об ошибке
      type: string
```

1.3 Запрос оценки выполнения запроса на Витрине

Данная спецификация описывает возможность получения оценки выполнения запросов на Витрине.

1.3.1 query.rq

`query.rq` - Топик sql запросов на исполнение

Структура сообщения

```
datamartExecuteQueryRequestMessage:
  description: Исполнение sql запроса на витрине
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  bindings:
    kafka:
      key:
        type: string
        format: uuid
        description: Уникальный идентификатор подзапроса
  headers:
    type: object
    properties:
      MESSAGE_TYPE:
        description: Тип сообщения
        type: string
        const: DatamartExecuteQueryRequest:0.1
      REQUEST_ID:
        description: Идентификатор запроса
        type: string
      QUERY_DEADLINE:
        description: Время в миллисекундах от эпохи, до которого запрос должен быть
выполнен
        type: string
        format: int64
      AGENT_CONSUMER_ID:
        description: Мнемоника потребителя (мнемоника агента)
        type: string
      QUERY_MNEMONIC:
        description: '<Полная мнемоника РЗ>.<версия РЗ>'
        type: string
```

```yaml
payload:
  $ref: '#/components/schemas/datamartExecuteQueryRequest'
examples:
  - name: simple
    summary: Простой запрос на исполнение без параметров
    headers:
      MESSAGE_TYPE: DatamartExecuteQueryRequest:0.1
    payload:
      requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
      subRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
      replyTo: agent-fias
      datamartMnemonic: fias
      sql: select * from v1_addrobj
      parameters: [ ]
      namedParams: [ ]
      tableParams: [ ]
      isForEstimation: false
      rowCountThreshold: -1
      customerId: aaa
      customerOgrn: ""
      queryMnemonic: fias.selectAllAddrobj.1.0
  - name: estimation
    summary: Запрос на оценку
    headers:
      MESSAGE_TYPE: DatamartExecuteQueryRequest:0.1
    payload:
      requestId: 403eada5-05f6-480c-bca9-03328091efeb
      subRequestId: 451000b8-dff2-4a1b-ab1b-42500a70d232
      replyTo: agent-fias
      datamartMnemonic: fias
      sql: select * from v1_addrobj
      parameters: [ ]
      namedParams: [ ]
      tableParams: [ ]
      isForEstimation: true
      rowCountThreshold: 1000
      customerId:
        string: agent-fias
      customerOgrn:
        string: "1053600591197"
      queryMnemonic:
        string: fias.selectAllAddrobj.1.0
  - name: complex
    summary: Запрос с параметрами
    headers:
      MESSAGE_TYPE: DatamartExecuteQueryRequest:0.1
    payload:
      requestId: 68758a92-0027-4258-bf17-aa3d24f85094
      subRequestId: 96e6eb99-7ff1-4efa-abae-ef1c5744b723
      replyTo: agent-fias
      datamartMnemonic: fias
      sql: select * from v1_addrobj where oktmo = ? and name = @tbl.fullname
      parameters:
        - type: STRING
          value:
            string: asdasdasd
        - type: LONG
          value: null
      namedParams: [ ]
      tableParams: [ ]
      isForEstimation: false
      rowCountThreshold: -1
```

```
        customerId:
          string: agent-fias
        customerOgrn:
          string: "1053600591197"
        queryMnemonic:
          string: fias.selectAddrobjWithParams.1.0
    - name: complex_named
      summary: Запрос с именованными параметрами
      headers:
        MESSAGE_TYPE: DatamartExecuteQueryRequest:0.1
      payload:
        requestId: 12358a92-0027-4258-bf17-aa3d24f85094
        subRequestId: 56e6eb99-7ff1-4efa-abae-ef1c5744b723
        replyTo: agent-fias
        datamartMnemonic: fias
        sql: select * from @tbl.fullname el LEFT JOIN v1_addrobj where oktmo = @p1 and
kod = @p2
        parameters: [ ]
        namedParams:
          - name: p1
            type: STRING
            value:
              string: asdasdasd
          - name: p2
            type: LONG
            value: null
        tableParams: [ ]
        isForEstimation: false
        rowCountThreshold: -1
        customerId:
          string: agent-fias
        customerOgrn:
          string: "1053600591197"
        queryMnemonic:
          string: fias.selectAddrobjWithParams.1.0
    - name: deadline
      summary: Простой запрос на исполнение без параметров
      headers:
        MESSAGE_TYPE: DatamartExecuteQueryRequest:0.1
        QUERY_DEADLINE: 1629289006904
      payload:
        requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
        subRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
        replyTo: agent-fias
        datamartMnemonic: fias
        sql: select * from v1_addrobj
        parameters: [ ]
        namedParams: [ ]
        tableParams: [ ]
        isForEstimation: false
        rowCountThreshold: -1
        customerId: agent-fias
        customerOgrn: "1053600591197"
        queryMnemonic: fias.selectAllWithDeadline.1.0
```

Avro-схема сообщения

```
datamartExecuteQueryRequest:
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  type: record
  name: QueryRequest
  namespace: datamart.query
```

```
  fields:
    - name: requestId
      description: Уникальный идентификатор запроса
      type:
        type: string
        logicalType: uuid
    - name: subRequestId
      description: Уникальный идентификатор подзапроса
      type:
        type: string
        logicalType: uuid
    - name: replyTo
      description: Служебная информация маршрутизации сообщения. Ответ, формируемый
витриной, обязан содержать переданное значение без каких либо искажений
      type: string
    - name: datamartMnemonic
      description: Мнемоника витрины, к которой выполняется запрос
      type: string
    - name: sql
      description: SQL запрос на исполнение, либо имя хранимой процедуры для
регламентированных запросов
      type: string
    - name: parameters
      description: Параметры к SQL запросу
      default: [ ]
      type:
        type: array
        items:
          type: record
          name: QueryParameter
          description: Описание параметра
          fields:
            - name: type
              type: string
              description: Тип параметра
              enum:
                - BIG_DECIMAL
                - BINARY
                - BOOLEAN
                - DATE
                - DOUBLE
                - FLOAT
                - INTEGER
                - LONG
                - SHORT
                - STRING
                - TIME
                - TIMESTAMP
            - name: value
              description: Значение параметра
              type:
                - string
                - 'null'
    - name: namedParams
      description: Именованные параметры запроса
      default: [ ]
      type:
        type: array
        items:
          type: record
          name: NamedParam
          description: Описание именованного параметра
```

```
        fields:
          - name: name
            description: Имя (мнемоника) параметра
            type: string
          - name: type
            type: string
            description: Тип параметра
            enum:
              - BIG_DECIMAL
              - BINARY
              - BOOLEAN
              - DATE
              - DOUBLE
              - FLOAT
              - INTEGER
              - LONG
              - SHORT
              - STRING
              - TIME
              - TIMESTAMP
          - name: value
            description: Значение параметра
            type:
              - string
              - 'null'
    - name: tableParams
      description: Табличные параметры запроса
      default: [ ]
      type:
        type: array
        items:
          type: record
          name: TableParam
          fields:
            - name: id
              description: Уникальный идентификатор
              type:
                type: string
                logicalType: uuid
            - name: name
              description: Имя параметра
              type: string
            - name: columns
              description: Описание колонок таблицы
              type:
                type: array
                items:
                  type: record
                  description: Описание колонки
                  name: TableParamColumnInfo
                  fields:
                    - name: name
                      type: string
                      description: Имя колонки
                    - name: type
                      type: string
                      description: Тип атрибута
                      enum:
                        - BIG_DECIMAL
                        - BINARY
                        - BOOLEAN
                        - DATE
```

```
                    - DOUBLE
                    - FLOAT
                    - INTEGER
                    - LONG
                    - SHORT
                    - STRING
                    - TIME
                    - TIMESTAMP
    - name: isForEstimation
      description: Признак необходимости вернуть статистику по запросу в качестве
результата. В случае, если оценка по результату исполнения sql запроса не превышает
rowCountThreshold записей, должен сразу отдаваться результат без отправки оценки в ядро
      type: boolean
      default: false
    - name: rowCountThreshold
      description: Максимальное оценочное количество строк результата, при превышении
которого возвращается статистика по запросу. Если оценка по запросу не превышет данный
параметр, витрина сразу возвращает ответ с результатом. Заполняется в случае
isForEstimation = true
      type: long
      default: -1
    - name: customerId
      description: Мнемоника ИС Потребителя
      type:
        - 'null'
        - string
      default: null
    - name: customerOgrn
      description: ОГРН ИС Потребителя
      type:
        - 'null'
        - string
      default: null
    - name: queryMnemonic
      description: 'Мнемоника РЗ, сформированная по правилу: <мнемоника
витрины>.<мнемоника РЗ>.<версия РЗ> Если запрос распределенный, то формируется по
правилу: podd.<мнемоника РЗ>.<версия РЗ>'
      type:
        - 'null'
        - string
      default: null
```

1.3.2 query.estimation.rs

Топик `QUERY.ESTIMATION.RS` предоставляет возможность произвести предварительную оценку объема получаемых данных при выполнении запроса к Витрине данных, а также, ограничить выгрузку данных в случае, если количество получаемых данных превысит заданное количество строк (параметр `rowCountThreshold`). В этом случае, ответом на запрос будет предварительная оценка объема.

Например, если вам нужна информация из какой-либо таблицы контактов, то, возможно, следует предварительно узнать, какой объем данных вы можете получить на такой запрос т.к ответ может содержать несколько гигабайт информации и выполнение запроса может занять много времени. Вы сможете установить ограничение на получение данных, например, не более 10 контактов из таблицы. В этом случае, если ответом на запрос будет 5 контактов, то Витрина предоставит ответ полностью. Если ответом будет 1000 контактов, то в качестве ответа будет сформирована предварительная оценка такого ответа, а именно, что данный ответ будет содержать 1000 строк и содержать информацию, например, на 15000 байт. Используя топик

query.estimation.rs можно прогнозировать объем получаемых данных, в соответствии с которыми оптимизировать запросы к Витрине.

В случае использования топика query.estimation.rs запрашивается не конечный ответ на запрос, а приблизительная оценка объема (байт) и количество строк в ответе.

> Примечание:
>
> Данное требование не распространяется на механизм подписок Потребителей данных ПОДД.

Алгоритм работы query.estimation.rs

1. Витрина получает запрос query.rq с признаком isForEstimation оценивает объем результата по этому запросу (в байтах и количестве строк).
2. Витрина сравнивает результаты оценки объема запроса со значением предельного числа строк в параметре rowCountThreshold (топик query.rq).
3. Если значение в оценке меньше, чем предельное значение в rowCountThreshold, то Витрина возвращает результат запроса в качестве ответа в топик query.rs.
4. Если значение оценки превышает предельное значение, возвращает предварительную оценку объема в качестве ответа.

Структура сообщения

```yaml
datamartQueryEstimationMessage:
  description: Оценка по запросу
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  bindings:
    kafka:
      key:
        type: string
        format: uuid
        description: Уникальный идентификатор подзапроса
  payload:
    $ref: '#/components/schemas/datamartQueryEstimation'
  examples:
    - name: estimation
      summary: Сообщение с оценкой по исполнению запроса
      payload:
        requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
        subRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
        estimatedRowCount: 100
        estimatedSize: 1000
        estimatedTime: 50
```

Avro-схема сообщения

```yaml
datamartQueryEstimation:
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  type: record
  name: Estimation
  namespace: datamart.query
  fields:
    - name: requestId
      description: Уникальный идентификатор запроса
      type:
        type: string
        logicalType: uuid
    - name: subRequestId
      description: Уникальный идентификатор подзапроса
      type:
        type: string
        logicalType: uuid
```

```
    - name: estimatedRowCount
      description: Оценка количества строк результата выполнения запроса
      type: long
    - name: estimatedSize
      description: Оценка объема результата выполнения запроса, в байтах
      type: long
    - name: estimatedTime
      description: Оценка времени выполнения запроса в миллисекундах
      type: long
  examples:
    - requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
      subRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
      estimatedRowCount: 100
      estimatedSize: 1000
      estimatedTime: 50
```

1.3.3 Запрос статистики

Данная спецификация описывает возможность запроса статистики Витрины.

1.3.4 statistics.rq

`statistics.rq` - Топик запросов статистики витрины

Структура сообщения

```
datamartStatisticRequestMessage:
  description: Запрос статистики витрины
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  bindings:
    kafka:
      key:
        $ref: '#/components/schemas/datamartStatisticRequestKey'
  headers:
    type: object
    properties:
      REQUEST_ID:
        description: Идентификатор запроса
        type: string
  payload:
    $ref: '#/components/schemas/datamartStatisticRequest'
  examples:
    - name: simple
      headers:
        REQUEST_ID: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
      payload:
        protocol: read.statistic.protocol.v.1
        requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
        datamart:
          mnemonic: fias
          version:
            major: 1
            minor: 0
```

Avro-схема сообщения

```
datamartStatisticRequest:
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  type: record
  name: DatamartStatisticRequest
  namespace: ru.rtlabs.common.statistic
  fields:
    - name: protocol
      description: Версия протокола. Указывается константа read.statistic.protocol.v.1
```

```
      type: string
    - name: requestId
      description: Уникальный идентификатор запроса
      type:
        type: string
        logicalType: uuid
    - name: datamart
      description: Витрина
      type:
        type: record
        name: DatamartInfo
        fields:
          - name: mnemonic
            description: Мнемоника витрины
            type: string
          - name: version
            description: Версия
            type:
              type: record
              name: SemanticVersion
              namespace: ru.rtlabs.common.model.metadata
              fields:
                - name: major
                  type: int
                  minimum: 1
                - name: minor
                  type: int
                  minimum: 0
```

1.3.5 statistics.rs

`statistics.rs` - Топик со статистикой витрины

Структура сообщения

```
datamartStatisticResponseMessage:
  description: Статистика витрины
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  bindings:
    kafka:
      key:
        type: string
        format: uuid
        description: Уникальный идентификатор запроса
  headers:
    type: object
    properties:
      REQUEST_ID:
        description: Идентификатор запроса
        type: string
  payload:
    $ref: '#/components/schemas/datamartStatisticResponse'
  examples:
    - name: simple
      headers:
        REQUEST_ID: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
      payload:
        protocol: read.statistic.protocol.v.1
        requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
        datamart:
          mnemonic: fias
          version:
            major: 1
```

```
          minor: 0
        tables:
          - mnemonic: addrobj
            columns:
              - mnemonic: oktmo
                notGreater10: 10.0
                inRange11And100: 50.0
                inRange101And1000: 30.0
                moreThan1000: 10.0
```

Avro-схема сообщения

```
datamartStatisticResponse:
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  type: record
  name: DatamartStatisticResponse
  namespace: ru.rtlabs.common.datamart.profile
  fields:
    - name: protocol
      description: Версия протокола. Указывается константа read.statistic.protocol.v.1
      type: string
    - name: requestId
      description: Уникальный идентификатор запроса
      type:
        type: string
        logicalType: uuid
    - name: datamart
      description: Статистика по витрине
      type:
        type: record
        name: DatamartStatistic
        fields:
          - name: mnemonic
            description: Мнемоника витрины
            type: string
          - name: version
            description: Версия
            type:
              type: record
              name: SemanticVersion
              namespace: ru.rtlabs.common.model.metadata
              fields:
                - name: major
                  type: int
                  minimum: 1
                - name: minor
                  type: int
                  minimum: 0
          - name: tables
            type:
              type: array
              items:
                type: record
                name: TableStatistic
                fields:
                  - name: mnemonic
                    description: Мнемоника витрины
                    type: string
                  - name: columns
                    description: Колонки
                    type:
                      type: array
```

```
                    items:
                      type: record
                      name: ColumnStatistic
                      description: Статистика по колонке
                      fields:
                        - name: mnemonic
                          type: string
                        - name: notGreater10
                          type: double
                        - name: inRange11And100
                          type: double
                        - name: inRange101And1000
                          type: double
                        - name: moreThan1000
                          type: double
```

1.3.6 statistics.err

`statistics.err` - Топик с ошибками получения статистики витрины

Структура сообщения

```
datamartStatisticErrorMessage:
  description: Неуспешный результат обработки запроса на получение статистики
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  bindings:
    kafka:
      key:
        type: string
        format: uuid
        description: Уникальный идентификатор запроса
  headers:
    type: object
    properties:
      REQUEST_ID:
        description: Идентификатор запроса
        type: string
  payload:
    $ref: '#/components/schemas/datamartStatisticError'
  examples:
    - name: simple
      headers:
        REQUEST_ID: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
      payload:
        protocol: read.statistic.protocol.v.1
        requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
        errorCode: DATAMART-001
        message: Непредвиденная ошибка
```

Avro-схема сообщения

```
datamartStatisticError:
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  type: record
  name: DatamartStatisticError
  namespace: ru.rtlabs.common.statistic
  fields:
    - name: protocol
      type: string
      description: Версия протокола. Указывается константа read.statistic.protocol.v.1
      conts: read.statistic.protocol.v.1
    - name: requestId
      description: Уникальный идентификатор запроса
```

```
      type:
        type: string
        logicalType: uuid
    - name: errorCode
      description: Код ошибки
      type: string
    - name: message
      description: Сообщение об ошибке
      type: string
```

1.4 Запрос данных по регламентированным запросам

Данная спецификация описывает возможность запроса данных по регламентированным запросам

1.4.1 procedure.query.rq

`procedure.query.rq` - Топик регламентированных запросов на исполнение

Структура сообщения

```
examples:
  - name: simple
    summary: Простой запрос на исполнение без параметров
    headers:
      MESSAGE_TYPE: DatamartExecuteQueryRequest:0.1
    payload:
      requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
      subRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
      replyTo: agent-fias
      datamartMnemonic: fias
      sql: select * from v1_addrobj
      parameters: [ ]
      namedParams: [ ]
      tableParams: [ ]
      isForEstimation: false
      rowCountThreshold: -1
      customerId: aaa
      customerOgrn: ""
      queryMnemonic: fias.selectAllAddrobj.1.0
  - name: estimation
    summary: Запрос на оценку
    headers:
      MESSAGE_TYPE: DatamartExecuteQueryRequest:0.1
    payload:
      requestId: 403eada5-05f6-480c-bca9-03328091efeb
      subRequestId: 451000b8-dff2-4a1b-ab1b-42500a70d232
      replyTo: agent-fias
      datamartMnemonic: fias
      sql: select * from v1_addrobj
      parameters: [ ]
      namedParams: [ ]
      tableParams: [ ]
      isForEstimation: true
      rowCountThreshold: 1000
      customerId:
        string: agent-fias
      customerOgrn:
        string: "1053600591197"
      queryMnemonic:
        string: fias.selectAllAddrobj.1.0
  - name: complex
    summary: Запрос с параметрами и табличными параметрами
```

```
    headers:
      MESSAGE_TYPE: DatamartExecuteQueryRequest:0.1
    payload:
      requestId: 68758a92-0027-4258-bf17-aa3d24f85094
      subRequestId: 96e6eb99-7ff1-4efa-abae-ef1c5744b723
      replyTo: agent-fias
      datamartMnemonic: fias
      sql: select * from v1_addrobj where oktmo = ? and name = @tbl.fullname
      parameters:
        - type: STRING
          value:
            string: asdasdasd
        - type: LONG
          value: null
      namedParams: [ ]
      tableParams: [ ]
      isForEstimation: false
      rowCountThreshold: -1
      customerId:
        string: agent-fias
      customerOgrn:
        string: "1053600591197"
      queryMnemonic:
        string: fias.selectAddrobjWithParams.1.0
  - name: complex_named
    summary: Запрос с именованными параметрами
    headers:
      MESSAGE_TYPE: DatamartExecuteQueryRequest:0.1
    payload:
      requestId: 12358a92-0027-4258-bf17-aa3d24f85094
      subRequestId: 56e6eb99-7ff1-4efa-abae-ef1c5744b723
      replyTo: agent-fias
      datamartMnemonic: fias
      sql: select * from @tbl.fullname el LEFT JOIN v1_addrobj where oktmo = @p1 and kod
= @p2
      parameters: [ ]
      namedParams:
        - name: p1
          type: STRING
          value:
            string: asdasdasd
        - name: p2
          type: LONG
          value: null
      tableParams: [ ]
      isForEstimation: false
      rowCountThreshold: -1
      customerId:
        string: agent-fias
      customerOgrn:
        string: "1053600591197"
      queryMnemonic:
        string: fias.selectAddrobjWithParams.1.0
  - name: deadline
    summary: Простой запрос на исполнение без параметров
    headers:
      MESSAGE_TYPE: DatamartExecuteQueryRequest:0.1
      QUERY_DEADLINE: 1629289006904
    payload:
      requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
      subRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
      replyTo: agent-fias
```

```
    datamartMnemonic: fias
    sql: select * from v1_addrobj
    parameters: [ ]
    namedParams: [ ]
    tableParams: [ ]
    isForEstimation: false
    rowCountThreshold: -1
    customerId: agent-fias
    customerOgrn: "1053600591197"
    queryMnemonic: fias.selectAllWithDeadline.1.0
```

Avro-схема сообщения

```
datamartExecuteQueryRequest:
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  type: record
  name: QueryRequest
  namespace: datamart.query
  fields:
    - name: requestId
      description: Уникальный идентификатор запроса
      type:
        type: string
        logicalType: uuid
    - name: subRequestId
      description: Уникальный идентификатор подзапроса
      type:
        type: string
        logicalType: uuid
    - name: replyTo
      description: Служебная информация маршрутизации сообщения. Ответ, формируемый витриной, обязан содержать переданное значение без каких либо искажений
      type: string
    - name: datamartMnemonic
      description: Мнемоника витрины, к которой выполняется запрос
      type: string
    - name: sql
      description: SQL запрос на исполнение, либо имя хранимой процедуры для регламентированных запросов
      type: string
    - name: parameters
      description: Параметры к SQL запросу
      default: [ ]
      type:
        type: array
        items:
          type: record
          name: QueryParameter
          description: Описание параметра
          fields:
            - name: type
              type: string
              description: Тип параметра
              enum:
                - BIG_DECIMAL
                - BINARY
                - BOOLEAN
                - DATE
                - DOUBLE
                - FLOAT
                - INTEGER
                - LONG
```

```
            - SHORT
            - STRING
            - TIME
            - TIMESTAMP
        - name: value
          description: Значение параметра
          type:
            - string
            - 'null'
  - name: namedParams
    description: Именованные параметры запроса
    default: [ ]
    type:
      type: array
      items:
        type: record
        name: NamedParam
        description: Описание именованного параметра
        fields:
          - name: name
            description: Имя (мнемоника) параметра
            type: string
          - name: type
            type: string
            description: Тип параметра
            enum:
              - BIG_DECIMAL
              - BINARY
              - BOOLEAN
              - DATE
              - DOUBLE
              - FLOAT
              - INTEGER
              - LONG
              - SHORT
              - STRING
              - TIME
              - TIMESTAMP
          - name: value
            description: Значение параметра
            type:
              - string
              - 'null'
  - name: tableParams
    description: ¥use only Datamart¥ Табличные параметры запроса
    default: [ ]
    type:
      type: array
      items:
        type: record
        name: TableParam
        fields:
          - name: id
            description: Уникальный идентификатор
            type:
              type: string
              logicalType: uuid
          - name: name
            description: Имя параметра
            type: string
          - name: columns
            description: Описание колонок таблицы
```

```
            type:
              type: array
              items:
                type: record
                description: Описание колонки
                name: TableParamColumnInfo
                fields:
                  - name: name
                    type: string
                    description: Имя колонки
                  - name: type
                    type: string
                    description: Тип атрибута
                    enum:
                      - BIG_DECIMAL
                      - BINARY
                      - BOOLEAN
                      - DATE
                      - DOUBLE
                      - FLOAT
                      - INTEGER
                      - LONG
                      - SHORT
                      - STRING
                      - TIME
                      - TIMESTAMP
    - name: isForEstimation
      description: Признак необходимости вернуть статистику по запросу в качестве результата. В случае, если оценка по результату исполнения sql запроса не превышает rowCountThreshold записей, должен сразу отдаваться результат без отправки оценки в ядро
      type: boolean
      default: false
    - name: rowCountThreshold
      description: Максимальное оценочное количество строк результата, при превышении которого возвращается статистика по запросу. Если оценка по запросу не превышет данный параметр, витрина сразу возвращает ответ с результатом. Заполняется в случае isForEstimation = true
      type: long
      default: -1
    - name: customerId
      description: Мнемоника ИС Потребителя
      type:
        - 'null'
        - string
      default: null
    - name: customerOgrn
      description: ОГРН ИС Потребителя
      type:
        - 'null'
        - string
      default: null
    - name: queryMnemonic
      description: 'Мнемоника РЗ, сформированная по правилу: <мнемоника витрины>.<мнемоника РЗ>.<версия РЗ> Если запрос распределенный, то формируется по правилу: podd.<мнемоника РЗ>.<версия РЗ>'
      type:
        - 'null'
        - string
      default: null
```

1.4.2 procedure.query.rs

`procedure.query.rs` - Топик с чанками данных исполнения запросов

Структура сообщения

```
datamartExecuteQueryResultChunk:
schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
type: record
name: QueryResultChunk
namespace: datamart.query
fields:
    - name: requestId
    description: Уникальный идентификатор запроса
    type:
        type: string
        logicalType: uuid
    - name: subRequestId
    description: Уникальный идентификатор подзапроса
    type:
        type: string
        logicalType: uuid
    - name: replyTo
    description: Служебная информация маршрутизации сообщения. Заполняется
соответсвующим значением из запроса
    type: string
    - name: chunkNumber
    description: Номер порции по порядку
    type: int
    minimum: 1
    - name: isLastChunk
    description: Признак последнего сообщения
    type: boolean
    - name: streamNumber
    description: Номер стрима данных
    minimum: 1
    type:
        - int
        - "null"
    - name: streamTotal
    description: Общее количество стримов
    minimum: 1
    type:
        - int
        - "null"
    - name: isFragmented
    description: Признак присутствия в чанке неполных строк (строк, которые были
разбиты на несколько чанков)
    type: boolean
    - name: uncompressedSize
    description: Оригинальный размер чанка в байтах
    type: int
    minimum: 0
```

Пример key query.rs

```
examples:
- requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
    subRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
    replyTo: agent-fias
    chunkNumber: 1
    isLastChunk: true
    streamNumber:
    int: 1
    streamTotal:
    int: 1
    isFragmented: false
    uncompressedSize: 10
```

1.4.3 procedure.query.err

`procedure.query.err` - Топик с ошибками исполнения sql запросов на витрине

Структура сообщения

```
datamartExecuteQueryError:
schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
type: record
name: QueryError
namespace: datamart.query
fields:
    - name: requestId
    description: Уникальный идентификатор запроса
    type:
        type: string
        logicalType: uuid
    - name: subRequestId
    description: Уникальный идентификатор подзапроса
    type:
        type: string
        logicalType: uuid
    - name: replyTo
    description: Служебная информация маршрутизации сообщения. Заполняется
соответсвующим значением из запроса
    type: string
    - name: errorCode
    description: Код возникшей ошибки
    type: string
    - name: message
    description: Сообщение с ошибкой исполнения
    type: string
```

Пример query.err

```
examples:
- name: error
    summary: Сообщение с ошибкой исполнения запроса на витрине без header
    payload:
    requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
    subRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
    replyTo: agent-fias
    errorCode: DATAMART-001
    message: Непредвиденная ошибка
- name: errorWithHeader
    summary: Сообщение с ошибкой исполнения запроса на витрине
    headers:
    MESSAGE_TYPE: DatamartExecuteQueryError:0.1
    payload:
    requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
    subRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
    replyTo: agent-fias
    errorCode: DATAMART-001
    message: Непредвиденная ошибка
```

1.5 Запрос метаданных

Данная спецификация описывает возможность запроса метаданных Витрины

1.5.1 metadata.rq

Передача Агентом ПОДД запроса метаданных в Витрину.

Формат сообщения

Header	Не используется
Key	текст, содержит requestId, не используется.
Value	Сериализация: в json (см. схему ниже.)

Схема

```
{
"$schema": "http://json-schema.org/draft-04/schema#",
"type": "object",
"properties": {
        "requestId": {
        "type": "string"
        },
        "datamartMnemonic": {
        "type": "string"
        }
},
"required": [
        "requestId",
        "datamartMnemonic"
]
}
```

, где:

- requestId - UUID запроса;
- datamartMnemonic - мнемоника Витрины данных, к которой адресован запрос.

1.5.2 metadata.rs

Передача Агенту ПОДД ответа на запрос метаданных витрины из Витрины.

Формат сообщения

Header	Не используется
Key	текст, содержит requestId
Value	Сериализация: в json (см. схему ниже.)

Схема

```
{
"$schema": "http://json-schema.org/draft-04/schema#",
"type": "object",
"properties": {
        "requestId": {
        "type": "string"
        },
        "metadata": {
        "type": "array",
        "items": [
                {
                "type": "object",
                "properties": {
                        "datamart": {
                        "type": "object",
                        "properties": {
                                "id": {
                                "type": "string"
                                },
                                "mnemonic": {
                                "type": "string"
                                },
                                "datamartClassess": {
                                "type": "array",
                                "items": [
                                        {
                                        "type": "object",
                                        "properties": {
                                                "id": {
                                                "type": "string"
                                                },
                                                "mnemonic": {
                                                "type": "string"
                                                },
                                                "label": {
                                                "type": "string"
                                                },
                                                "classAttributes": {
                                                "type": "array",
                                                "items": [
                                                        {
                                                        "type": "object",
                                                        "properties": {
                                                                "id": {
                                                                "type": "string"
                                                                },
                                                                "mnemonic": {
                                                                "type": "string"
                                                                },
                                                                "type": {
                                                                "type": "object",
                                                                "properties": {
                                                                        "id": {
                                                                        "type": "string"
                                                                        },
```

```
                                        "value": {
                                        "type": "string"
                                        }
                                },
                                "required": [
                                        "id",
                                        "value"
                                ]
                                }
                        },
                        "required": [
                                "id",
                                "mnemonic",
                                "type"
                        ]
                        }
                ]
                },
                "primaryKey": {
                "type": "array",
                "items": [
                        {
                        "type": "object",
                        "properties": {
                                "id": {
                                "type": "string"
                                },
                                "mnemonic": {
                                "type": "string"
                                },
                                "type": {
                                "type": "object",
                                "properties": {
                                        "id": {
                                        "type": "string"
                                        },
                                        "value": {
                                        "type": "string"
                                        }
                                },
                                "required": [
                                        "id",
                                        "value"
                                ]
                                }
                        },
                        "required": [
                                "id",
                                "mnemonic",
                                "type"
                        ]
                        }
                ]
                }
        },
        "required": [
                "id",
                "mnemonic",
                "label",
                "classAttributes",
                "primaryKey"
        ]
```

```
                                        }
                                ]
                                }
                        },
                        "required": [
                                "id",
                                "mnemonic",
                                "datamartClassess"
                        ]
                        }
                },
                "required": [
                        "datamart"
                ]
                }
        ]
        }
},
"required": [
        "requestId",
        "metadata"
]
}
```

, где:

- `requestId` - UUID запроса;
- `metadata` – описание структуры данных;
- `datamart` – описание витрины;
- `id` – UUID витрины (не используется);
- `mnemonic` – имя витрины;
- `datamartClassess` – список таблиц витрины;
- `id` – UUID таблицы (не используется);
- `mnemonic` – имя таблицы;
- `label`– не используется;
- `classAttributes` – список полей таблицы;
- `id` – UUID поля (не используется);
- `mnemonic` – имя поля;
- `type` – тип данных поля;
- `id` – UUID типа данных (не используется);
- `value` – название типа данных;
- `primaryKey` – список полей, составляющих PK таблицы;
- `id` – UUID поля (не используется);
- `mnemonic` – имя поля;
- `type` – тип данных поля;
- `id` – UUID типа данных (не используется);
- `value` – название типа данных.

1.5.3 metadata.err

Получение Агентом ПОДД ошибки при обработке запроса метаданных от Витрины.

Формат сообщения

Header	Не используется
Key	текст, содержит requestId

Value	Сериализация: в json (см. схему ниже.)

Схема

```
{
"$schema": "http://json-schema.org/draft-04/schema#",
"type": "object",
"properties": {
        "requestId": {
        "type": "string"
        },
        "errorCode": {
        "type": "string"
        },
        "msg": {
        "type": "string"
        }
},
"required": [
        "requestId",
        "errorCode",
        "msg"
]
}
```

, где:

- `requestId` - UUID запроса;
- `errorCode` – содержит константу `INTERNAL`;
- `msg` – описание ошибки.

2 Спецификация модуля «BLOB-адаптер»

2.1 Запрос на считывание BLOB

Настоящая спецификация определяет формат обмена электронными сообщениями через BLOB-адаптер. Описывает возможность запроса на считывание BLOB-объект по полученной ссылке, получения успешного ответа на чтение содержимого BLOB или ошибки, в случае невозможности считывания, с описанием причины ошибки.

Топик	Назначение
blob.rq	Запросы на считывание BLOB'а по ссылке.
blob.rs	Содержимое BLOB'а (ответ на запрос).
blob.err	Сообщения об ошибке считывания.

Для строковых параметров используется кодировка UTF-8.

2.2.1 blob.rq

`blob.rq` - Топик запросов на получение бинарных данных по полученной ранее ссылке.

Структура сообщения

```
datamartBlobRequestMessage:
  description: Запрос бинарных данных по ссылке
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  bindings:
    kafka:
      key:
        type: string
        format: uuid
```

```
      description: Уникальный идентификатор подзапроса
  headers:
    type: object
    properties:
      REQUEST_ID:
        description: Идентификатор запроса
        type: string
      AGENT_CONSUMER_ID:
        description: Идентификатор агента потребителя
        type: string
      MESSAGE_TYPE:
        description: Тип сообщения
        type: string
        const: DatamartBlobRequest:0.1
  payload:
    $ref: '#/components/schemas/datamartBlobRequest'
  examples:
    - name: getBlobDataRequest
      summary: Запрос бинарных данных по ссылке
      headers:
        REQUEST_ID: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
        AGENT_CONSUMER_ID: agent-fias
        MESSAGE_TYPE: DatamartBlobRequest:0.1
      payload:
        requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
        queryRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
        reference:
          subRequestId: 4cbb11d6-47de-4928-953f-47dfa6c6b310
          path: reference
```

Avro-схема сообщения

```
datamartBlobRequest:
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  type: record
  name: BlobRequest
  namespace: datamart.blob
  fields:
    - name: requestId
      description: Уникальный идентификатор запроса
      type:
        type: string
        logicalType: uuid
    - name: queryRequestId
      description: Идентификатор исходного запроса, в рамках которого была получена
ссылка
      type:
        type: string
        logicalType: uuid
    - name: reference
      description: Ссылка на данные
      type:
        type: record
        name: BinaryReference
        namespace: query.result
        fields:
          - name: subRequestId
            description: Идентификатор подзапроса
            type:
              type: string
              logicalType: uuid
          - name: path
```

```
          description: Ссылка
          type: string
```

2.2.2 blob.rs

`blob.rs` - Топик с бинарными данным блобов

Структура сообщения

```
datamartBlobChunkMessage:
  description: Чанки бинарных данных
  contentType: 'application/octet-stream'
  bindings:
    kafka:
      key:
        $ref: '#/components/schemas/datamartBlobChunkInfo'
  headers:
    type: object
    properties:
      AGENT_CONSUMER_ID:
        description: Идентификатор агента потребителя
        type: string
      MESSAGE_TYPE:
        description: Тип сообщения
        type: string
        const: DatamartBlobChunkInfo:0.1
  payload:
    description: Бинарные данные
  examples:
    - name: base64
      headers:
        MESSAGE_TYPE: DatamartBlobChunkInfo:0.1
      payload:
        value: JEEJNodyLO7p1pgsRHG9pEiXeYGvHW4YCl4FgrgBmu5C92iVX1PV2GZdcqsb66bx8sk=
```

Avro-схема сообщения

```
datamartBlobChunkInfo:
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  type: record
  name: BlobChunk
  namespace: datamart.blob
  fields:
    - name: requestId
      description: Уникальный идентификатор запроса
      type:
        type: string
        logicalType: uuid
    - name: queryRequestId
      description: Идентификатор исходного запроса, в рамках которого была получена
ссылка
      type:
        type: string
        logicalType: uuid
    - name: chunkNum
      description: Номер чанка
      type: int
      minimum: 1
    - name: isLast
      description: Признак последнего чанка
      type: boolean
  examples:
    - requestId: 3546e40b-47fe-41b6-9c06-a2e915eb4181
```

```
      queryRequestId: a8e9f47b-38cd-4db6-a245-0fbd6e78c195
      chunkNum: 1
      isLast: true
```

2.2.3 blob.err

`blob.err` - Топик с ошибками получения бинарных данных по ссылке.

Структура сообщения

```
datamartBlobErrorResponseMessage:
  description: Ошибка получения бинарных данных по ссылке
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  bindings:
    kafka:
      key:
        type: string
        format: uuid
        description: Уникальный идентификатор подзапроса
  headers:
    type: object
    properties:
      AGENT_CONSUMER_ID:
        description: Идентификатор агента потребителя
        type: string
      MESSAGE_TYPE:
        description: Тип сообщения
        type: string
        const: DatamartBlobErrorResponse:0.1
  payload:
    $ref: '#/components/schemas/datamartBlobErrorResponse'
  examples:
    - name: blobError
      summary: Пример ошибки получения бинарных данных по ссылке
      headers:
        AGENT_CONSUMER_ID: agent-fias
        MESSAGE_TYPE: DatamartBlobErrorResponse:0.1
      payload:
        requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
        queryRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
        errorCode: DATAMART-001
        errorMessage: Непредвиденная ошибка обработки
```

Avro-схема сообщения

```
datamartBlobErrorResponse:
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  type: record
  name: BlobError
  namespace: datamart.blob
  fields:
    - name: requestId
      description: Уникальный идентификатор запроса
      type:
        type: string
        logicalType: uuid
    - name: queryRequestId
      description: Идентификатор исходного запроса, в рамках которого была получена
ссылка
      type:
        type: string
        logicalType: uuid
    - name: errorCode
```

```
      description: Код ошибки
      type: string
    - name: errorMessage
      description: Сообщение с ошибкой
      type: string
```

3 Спецификация модуля «Сервис Формирования документов»

3.1 Запрос формирования документов

Данная спецификация описывает возможность запроса на генерацию формирования файлов, получения успешного ответа (сгенерированных файлов и их метаданных) или ошибки, в случае невозможности сгенерировать файлы, с описанием причины ошибки.

Топик	Назначение
Report.rq	Запросы на генерацию файлов.
Report.rs	Содержимое сгенерированных файлов (ответ на запрос).
Report.err	Сообщения об ошибке генерации.

3.1.1 report.rq

> Внимание:
>
> Название топика может быть изменено на этапе внедрения!

Запрос на генерацию файлов. Одно сообщение - один запрос. Один запрос - один набор параметров (в наборе м.б. много параметров, в параметрах м.б. указано «сгенерируй много форматов файлов на основании одной и той же выборки данных»).

Структура сообщения

```
datamartExecuteQueryRequestMessage:
description: Исполнение sql запроса на витрине
schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
bindings:
    kafka:
    key:
        type: string
        format: uuid
        description: Уникальный идентификатор подзапроса
headers:
    type: object
    properties:
    MESSAGE_TYPE:
        description: Тип сообщения
        type: string
        const: DatamartExecuteQueryRequest:0.1
    REQUEST_ID:
        description: Идентификатор запроса
        type: string
    QUERY_DEADLINE:
        description: Время в миллисекундах от эпохи, до которого запрос должен быть
выполнен
        type: string
        format: int64
    AGENT_CONSUMER_ID:
        description: Мнемоника потребителя (мнемоника агента)
        type: string
    QUERY_MNEMONIC:
        description: '<Полная мнемоника РЗ>.<версия РЗ>'
```

```
        type: string
payload:
    $ref: '#/components/schemas/datamartExecuteQueryRequest'
examples:
    - name: report
    summary: Запрос к Сервису формирования документов
    headers:
        MESSAGE_TYPE: DatamartExecuteQueryRequest:0.1
    payload:
        requestId: 68758a92-0027-4258-bf17-aa3d24f85094
        subRequestId: 96e6eb99-7ff1-4efa-abae-ef1c5744b723
        replyTo: agent-fias
        datamartMnemonic: fias
        sql: v1_printable_form_address
        parameters:
        - type: STRING
            value:
            string:
MIIB9wYJKoZIhvcNAQcCoIIB6DCCAeQCAQExADALBgkqhkiG9w0BBwGgggHMMIIByDCCAXOgAwIBAgIEV/dqTjA
MBggqhQMHAQEDAgUAMDUxCzAJBgNVBAYTAlJVMQswCQYDVQQKEwJSVDEZMBcGA1UEAwwQYmxhc3RvZmZfY2FfdG
VzdDAeFw0yMjAzMDQwNzQ5MzBaFw0zMjAzMDEwNzQ5MzBaMC0xETAPBgNVBAMMCGl0b25lZGV2MQswCQYDVQQKD
AJSVDELMAk
        - type: STRING
            value:
            string: xml
        namedParams: [ ]
        tableParams: [ ]
        isForEstimation: false
        rowCountThreshold: -1
        customerId:
        string: agent-fias
        customerOgrn:
        string: "1053600591197"
        queryMnemonic:
        string: fias.selectAddrobjWithParams.1.0
```

Avro-схема сообщения

```
datamartExecuteQueryRequest:
schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
type: record
name: QueryRequest
namespace: datamart.query
fields:
    - name: requestId
    description: Уникальный идентификатор запроса
    type:
        type: string
        logicalType: uuid
    - name: subRequestId
    description: Уникальный идентификатор подзапроса
    type:
        type: string
        logicalType: uuid
    - name: replyTo
    description: Служебная информация маршрутизации сообщения. Ответ, формируемый витриной, обязан содержать переданное значение без каких либо искажений
    type: string
    - name: datamartMnemonic
    description: Мнемоника витрины, к которой выполняется запрос
    type: string
    - name: sql
```

```
    description: тип отчета
    type: string
    - name: parameters
    description: Параметры к SQL запросу (при запросе к сервису формирования
документов, первым параметром ВСЕГДА идет сертификат, а вторым формат файла (xml или
pdf))
    default: [ ]
    type:
        type: array
        items:
        type: record
        name: QueryParameter
        description: Описание параметра
        fields:
            - name: type
            type: string
            description: Тип параметра
            enum:
                - BIG_DECIMAL
                - BINARY
                - BOOLEAN
                - DATE
                - DOUBLE
                - FLOAT
                - INTEGER
                - LONG
                - SHORT
                - STRING
                - TIME
                - TIMESTAMP
            - name: value
            description: Значение параметра
            type:
                - string
                - 'null'
    - name: namedParams
    description: Именованные параметры запроса
    default: [ ]
    type:
        type: array
        items:
        type: record
        name: NamedParam
        description: Описание именованного параметра
        fields:
            - name: name
            description: Имя (мнемоника) параметра
            type: string
            - name: type
            type: string
            description: Тип параметра
            enum:
                - BIG_DECIMAL
                - BINARY
                - BOOLEAN
                - DATE
                - DOUBLE
                - FLOAT
                - INTEGER
                - LONG
                - SHORT
                - STRING
```

```
                - TIME
                - TIMESTAMP
            - name: value
            description: Значение параметра
            type:
                - string
                - 'null'
    - name: tableParams
    description: Табличные параметры запроса
    default: [ ]
    type:
        type: array
        items:
        type: record
        name: TableParam
        fields:
            - name: id
            description: Уникальный идентификатор
            type:
                type: string
                logicalType: uuid
            - name: name
            description: Имя параметра
            type: string
            - name: columns
            description: Описание колонок таблицы
            type:
                type: array
                items:
                type: record
                description: Описание колонки
                name: TableParamColumnInfo
                fields:
                    - name: name
                    type: string
                    description: Имя колонки
                    - name: type
                    type: string
                    description: Тип атрибута
                    enum:
                        - BIG_DECIMAL
                        - BINARY
                        - BOOLEAN
                        - DATE
                        - DOUBLE
                        - FLOAT
                        - INTEGER
                        - LONG
                        - SHORT
                        - STRING
                        - TIME
                        - TIMESTAMP
    - name: isForEstimation
    description: Признак необходимости вернуть статистику по запросу в качестве
результата. В случае, если оценка по результату исполнения sql запроса не превышает
rowCountThreshold записей, должен сразу отдаваться результат без отправки оценки в ядро
    type: boolean
    default: false
    - name: rowCountThreshold
    description: Максимальное оценочное количество строк результата, при превышении
которого возвращается статистика по запросу. Если оценка по запросу не превышет данный
параметр, витрина сразу возвращает ответ с результатом. Заполняется в случае
```

```
isForEstimation = true
    type: long
    default: -1
    - name: customerId
    description: Мнемоника ИС Потребителя
    type:
        - 'null'
        - string
    default: null
    - name: customerOgrn
    description: ОГРН ИС Потребителя
    type:
        - 'null'
        - string
    default: null
    - name: queryMnemonic
    description: 'Мнемоника РЗ, сформированная по правилу: <мнемоника
витрины>.<мнемоника РЗ>.<версия РЗ> Если запрос распределенный, то формируется по
правилу: podd.<мнемоника РЗ>.<версия РЗ>'
    type:
        - 'null'
        - string
    default: null
```

3.1.2 report.rs

> Внимание:
>
> Название топика может быть изменено на этапе внедрения!

Позитивный ответ на запрос - содержимое сгенерированных файлов и метаданные для них, передается только в случае успешного выполнения генерации. Один запрос - один ответ. Один ответ - несколько сообщений. Одно сообщение - один chunk.

Структура сообщения

```
datamartExecuteQueryResultChunkMessage:
  description: Чанк с данными по исполнению запроса
  contentType: 'application/octet-stream'
  bindings:
    kafka:
      key:
        $ref: '#/components/schemas/datamartExecuteQueryResultChunk'
  headers:
    type: object
    properties:
      MESSAGE_TYPE:
        description: Тип сообщения
        type: string
        const: DatamartExecuteQueryResultChunk:0.1
  payload:
    description: Бинарные данные чанка
  examples:
    - name: base64
      headers:
        MESSAGE_TYPE: DatamartExecuteQueryResultChunk:0.1
      payload:
        value: JEEJNodyLO7p1pgsRHG9pEiXeYGvHW4YCl4FgrgBmu5C92iVX1PV2GZdcqsb66bx8sk=
```

Avro-схема Key сообщения

```
datamartExecuteQueryResultChunk:
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
```

```yaml
  type: record
  name: QueryResultChunk
  namespace: datamart.query
  fields:
    - name: requestId
      description: Уникальный идентификатор запроса
      type:
        type: string
        logicalType: uuid
    - name: subRequestId
      description: Уникальный идентификатор подзапроса
      type:
        type: string
        logicalType: uuid
    - name: replyTo
      description: Служебная информация маршрутизации сообщения. Заполняется
соответсвующим значением из запроса
      type: string
    - name: chunkNumber
      description: Номер порции по порядку
      type: int
      minimum: 1
    - name: isLastChunk
      description: Признак последнего сообщения
      type: boolean
    - name: streamNumber
      description: Номер стрима данных
      minimum: 1
      type:
        - int
        - "null"
    - name: streamTotal
      description: Общее количество стримов
      minimum: 1
      type:
        - int
        - "null"
    - name: isFragmented
      description: Признак присутствия в чанке неполных строк (строк, которые были
разбиты на несколько чанков)
      type: boolean
    - name: uncompressedSize
      description: Оригинальный размер чанка в байтах
      type: int
      minimum: 0
  examples:
    - requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
      subRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
      replyTo: agent-fias
      chunkNumber: 1
      isLastChunk: true
      streamNumber:
        int: 1
      streamTotal:
        int: 1
      isFragmented: false
      uncompressedSize: 10
```

Avro-схема Value (для report.rs она универсальная)

```
{
  "type": "record",
```

```
  "name": "QueryResultRow",
  "namespace": "datamart.query",
  "fields": [
    {
      "name": "DocType",
      "type": "string"
    },
    {
      "name": "FileName",
      "type": "string"
    },
    {
      "name": "Content",
      "type": "bytes"
    },
    {
      "name": "Meta",
      "type": ["string", "null"]
    }
  ]
}
```

3.1.3 report.err

> Внимание:
>
> Название топика может быть изменено на этапе внедрения!

Негативный ответ на запрос - описание причины ошибки, передается только в случае невозможности выполнения запроса (если для одного из форматов в запросе не настроена генерация, то возвращаются настроенные форматы и это не считается ошибкой). Один запрос - один ответ (об ошибке). Один ответ - одно сообщение.

Структура сообщения

```
datamartExecuteQueryErrorMessage:
  description: Ошибка исполнения запроса
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  bindings:
    kafka:
      key:
        type: string
        format: uuid
        description: Уникальный идентификатор подзапроса
  headers:
    type: object
    properties:
      MESSAGE_TYPE:
        description: Тип сообщения
        type: string
        const: DatamartExecuteQueryError:0.1
  payload:
    $ref: '#/components/schemas/datamartExecuteQueryError'
  examples:
    - name: error
      summary: Сообщение с ошибкой исполнения запроса на витрине без header
      payload:
        requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
        subRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
        replyTo: agent-fias
        errorCode: DATAMART-001
        message: Непредвиденная ошибка
```

```
    - name: errorWithHeader
      summary: Сообщение с ошибкой исполнения запроса на витрине
      headers:
        MESSAGE_TYPE: DatamartExecuteQueryError:0.1
      payload:
        requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
        subRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
        replyTo: agent-fias
        errorCode: DATAMART-001
        message: Непредвиденная ошибка
```

Avro-схема сообщения

```
datamartExecuteQueryError:
  schemaFormat: 'application/vnd.apache.avro;version=1.9.0'
  type: record
  name: QueryError
  namespace: datamart.query
  fields:
    - name: requestId
      description: Уникальный идентификатор запроса
      type:
        type: string
        logicalType: uuid
    - name: subRequestId
      description: Уникальный идентификатор подзапроса
      type:
        type: string
        logicalType: uuid
    - name: replyTo
      description: Служебная информация маршрутизации сообщения. Заполняется
соответсвующим значением из запроса
      type: string
    - name: errorCode
      description: Код возникшей ошибки
      type: string
    - name: message
      description: Сообщение с ошибкой исполнения
      type: string
  examples:
    - requestId: 2e8c8ab2-44db-4dcb-8ae5-2365121b4e14
      subRequestId: 608c5a5a-01d4-4439-8220-dda41a8519fe
      replyTo: agent-fias
      errorCode: DATAMART-001
      message: Непредвиденная ошибка
```

ПРИЛОЖЕНИЕ 2. ПОДДЕРЖКА ФУНКЦИЙ SQL

1 SQL-синтаксис

Регламентированные SQL-запросы, регистрируемые в СМЭВ4 должны соответствовать следующему синтаксису.

Таблица 1 SQL-синтаксис

№	Описание	Пример запроса
	Числовые типы данных	
1	Типы данных INTEGER и SMALLINT	`SELECT CAST(1 AS INT)`
2	Типы данных REAL, DOUBLE PRECISION и FLOAT	`SELECT CAST(1 AS FLOAT)`
3	Типы данных DECIMAL и NUMERIC	`SELECT CAST(1 AS NUMERIC)`
4	Арифметические операторы	`SELECT 10+1, 9-2, 8*3, 7/2`
5	Числовые сравнения	`SELECT 1 WHERE 1 < 2`
6	Неявные преобразования между числовыми типами данных	`SELECT int_column FROM t WHERE int_column = 1.00`
Символьные типы данных		
7	Тип данных CHARACTER Длина по умолчанию 30	`SELECT CAST('1111111111111111111111111111111111111111' AS CHAR)`
8	Тип данных CHARACTER VARYING Длина по умолчанию 30	`SELECT CAST('1111111111111111111111111111111111111111' AS VARCHAR)`
9	Символьные строки	`SELECT ''`
10	Функция CHARACTER_LENGTH убирает завершающие пробелы из значений CHARACTER перед подсчётом символов	`SELECT character_length(char_column) FROM t`
11	Функция OCTET_LENGTH	`SELECT octet_length(char_column) FROM t`
12	Функция SUBSTRING	`SELECT substring(char_column FROM 1 FOR 1) FROM t`
13	Конкатенация символьных строк	`SELECT 'a' || 'b' FROM t`
14	Функции UPPER и LOWER	`SELECT upper('a'),lower('B') FROM t` `SELECT int_column FROM t WHERE int_column > (SELECT DISTINCT (int_column) FROM t)`
15	Функция TRIM	`SELECT trim('a ') FROM t`
16	Неявные преобразования между типами символьных строк	`SELECT char_column FROM t WHERE char_column > varchar_column`
17	Функция POSITION	`SELECT position('A' IN char_column) FROM t`
18	Сравнения символов	`SELECT char_column FROM t WHERE char_column > 'a'`
Идентификаторы		
19	Идентификаторы с разделителями	`SELECT 1 AS «t47»`
20	Идентификаторы в нижнем регистре	`SELECT 1 AS t48`
21	Завершающее подчёркивание	`SELECT 1 AS t49_`
Базовое определение запросов		
22	SELECT DISTINCT	`SELECT DISTINCT int_column FROM t`
23	Предложение GROUP BY	`SELECT DISTINCT int_column FROM t GROUP BY int_column`
24	GROUP BY может содержать колонки не из <списка выборки>	`SELECT DISTINCT char_column FROM t GROUP BY lower(char_column)`
25	Элементы списка выборки могут переименовываться	`SELECT int_column AS K FROM t ORDER BY K`
26	Предложение HAVING	`SELECT count(*) FROM t HAVING count(*) > 0`
27	Корреляционные имена в предложении FROM	`SELECT K.column FROM t AS K`
28	Переименование колонок в	`SELECT column FROM t AS x(q, c)`

№	Описание	Пример запроса
	предложении FROM	
Базовые предикаты и условия поиска		
29	Предикат сравнения	`SELECT column FROM t WHERE 0 = 0`
30	Предикат BETWEEN	`SELECT column FROM t WHERE ' ' BETWEEN '' AND ''`
31	Предикат IN со списком значений	`SELECT column FROM t WHERE char_column IN ('a', upper('a'))`
32	Предикат LIKE	`SELECT column FROM t WHERE char_column LIKE '_'`
33	Предложение ESCAPE в предикате LIKE	`SELECT column FROM t WHERE 'abc' LIKE 'abcX_' ESCAPE 'X'`
34	Предикат NULL	`SELECT column FROM t WHERE char_column IS NOT NULL`
35	Предикаты количественного сравнения	`SELECT column FROM t WHERE char_column = ANY (SELECT char_column FROM t)`
36	Предикат EXISTS	`SELECT column FROM t WHERE NOT EXISTS (SELECT char_column FROM t)`
37	Подзапросы в предикате сравнения	`SELECT column FROM t WHERE int_column > (SELECT max (int_column) FROM t)`
38	Подзапросы в предикате IN	`SELECT column FROM t WHERE char_column IN (SELECT char_column FROM t)`
39	Подзапросы в предикате количественного сравнения	`SELECT column FROM t WHERE char_column >= ALL(SELECT char_column FROM t)`
40	Коррелирующие подзапросы	`SELECT column FROM t WHERE int_column = (SELECT int_column FROM t2 WHERE t2.char_ column = t.char_column)`
41	Условие поиска	`SELECT column FROM t WHERE 0 <> 0 OR 'a' < 'b' AND int_column IS NOT NULL`
Простые выражения с запросами		
42	Табличный оператор UNION DISTINCT	`SELECT column FROM t UNION DISTINCT SELECT column1 FROM t`
43	Табличный оператор UNION ALL	`SELECT column FROM t UNION ALL SELECT column1 FROM t`
44	Табличный оператор EXCEPT DISTINCT	`SELECT column FROM t EXCEPT DISTINCT SELECT column1 FROM t`
45	Колонки, объединяемые табличными операторами, могут иметь разные типы данных	`SELECT char_column FROM t UNION SELECT 5`
46	Табличные операторы в подзапросах	`SELECT column FROM t WHERE 'a' IN (SELECT char_column FROM t UNION SELECT char_column FROM t)`
Функции множеств		
47	AVG	`SELECT avg(int_column) FROM t`
48	COUNT	`SELECT count(int_column) FROM t`
49	MAX	`SELECT max(int_column) FROM t`
50	MIN	`SELECT min(int_column) FROM t`
51	SUM	`SELECT sum(int_column) FROM t`
52	Дополнение ALL	`SELECT sum(ALL int_column) FROM t`
53	Дополнение DISTINCT	`SELECT sum(DISTINCT int_column) FROM t`
54	Оператор SELECT, возвращающий одну строку	`SELECT count(*) FROM t`
Базовая поддержка курсоров		
55	Колонки ORDER BY, отсутствующие в списке выборки	`SELECT int_column FROM t ORDER BY char_column`
56	Выражения значений в предложении ORDER BY	`SELECT int_column FROM t ORDER BY -int_column`
57	Поддержка NULL (NULL вместо значений)	`SELECT int_column FROM t WHERE int_column IS NULL`
Базовое соединение таблиц		
58	Внутреннее соединение (но не обязательно с ключевым словом INNER)	`SELECT a.int_column FROM t a JOIN t b ON a.int_column = b.int_column`
59	Ключевое слово INNER	`SELECT a.int_column FROM t a JOIN t b ON a.int_column = b.int_column`

№	Описание	Пример запроса
60	LEFT OUTER JOIN	`SELECT a.int_column, b.int_column FROM t a LEFT OUTER JOIN t b ON a.int_column = b.int_column`
61	RIGHT OUTER JOIN	`SELECT a.int_column, b.int_column FROM t a RIGHT OUTER JOIN t b ON a.int_column = b.int_column`
62	Внешние соединения могут быть вложенными	`SELECT a.int_column FROM t a LEFT OUTER JOIN t b ON a.int_column = b.int_column LEFT OUTER JOIN t2 c ON a.int_column = c.int_column`
63	Внутренняя таблица с левой или правой стороны внешнего соединения может также участвовать во внутреннем соединении	`SELECT t2.int_column FROM (t2 LEFT OUTER JOIN t ON t.int_column = t2.int_column) j INNER JOIN t2 ON j.int_column = t2.int_column`
64	Поддерживаются все операторы сравнения (а не только =)	`SELECT column FROM t WHERE 0 = 1 OR 0 > 1 OR 0 < 1 OR 0 <> 1`
Базовая поддержка даты и времени		
65	Тип данных DATE (включая поддержку строк DATE)	`SELECT ‘2012-07-12’ AS "DATE"`
66	Тип данных TIME (включая поддержку строк TIME) с точностью до секунд как минимум с 0 знаков после запятой	`SELECT ‘1:2:3’ AS "TIME"`
67	Тип данных TIMESTAMP (включая поддержку строк TIMESTAMP) с точностью до секунд как минимум с 0 и 6 знаками после запятой	`SELECT ‘2012-07-12’ AS "TIMESTAMP"`
68	Предикаты сравнения с типами данных DATE, TIME и TIMESTAMP	`SELECT column FROM t3 WHERE date_column = date_column AND time_column = time_column AND timestamp_column = timestamp_column`
69	Явное приведение (CAST) между типами даты/времени и типами символьных строк	`SELECT cast(date_column AS VARCHAR(10)) FROM t3`
70	CURRENT_DATE	`SELECT current_date FROM t`
71	LOCALTIME	`SELECT localtime FROM t`
72	LOCALTIMESTAMP	`SELECT localtimestamp FROM t`
Расширенная поддержка даты и времени		
73	FOR SYSTEM_TIME (запрос данных, актуальных на указанную дату и время)	`SELECT column FROM t FOR SYSTEM_TIME AS OF 'YYYY-MM-DD hh:mm:ss'` `SELECT column FROM t FOR SYSTEM_TIME AS OF TIMESTAMP 'YYYY-MM-DD hh:mm:ss'`
74	Указание «TIMESTAMP» опционально Вычисление интервала (с указанием единиц времени: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND)	`select ('YYYY-MM-DD hh:mm:ss' - 'YYYY-MM-DD hh:mm:ss') MONTH`
75	Функция CAST	`SELECT cast(int_column AS INT) FROM t`
Выражение CASE		
76	Простой оператор CASE	`SELECT CASE WHEN 1 = 0 THEN 5 ELSE 7 END FROM t`
77	Оператор CASE с условиями	`SELECT CASE 1 WHEN 0 THEN 5 ELSE 7 END FROM t`
78	NULLIF	`SELECT nullif(int_column, 7) FROM t`
79	COALESCE	`SELECT coalesce(int_column,7) FROM t`
80	Длинные идентификаторы	`SELECT 1 AS A123456789012345678901234567890123456789`
81	Спецсимволы Unicode в идентификаторах	`SELECT 1 AS Я123456789012345678901234567890123456789`
82	Спецсимволы Unicode в текстовых строках	`SELECT U&'\6553'`
83	Национальные символы	`SELECT 'Я'`
84	Скалярные значения подзапросов	`SELECT int_column  FROM t WHERE int_column = (SELECT`

№	Описание	Пример запроса
		`count(*) FROM t)`
85	Расширенный предикат NULL	`SELECT column FROM t WHERE row(int_column, int_column) IS NOT NULL`
Функции по управлению текстовым поиском		
86	Функция TO_TSVECTOR подготовки текстовых значения, выбранных запросом	`SELECT TO_TSVECTOR('russian', 'иванов иван')`
87	Функция PLAINTO_TSQUERY подготовки неформатированного текста без сохранения порядка слов	`SELECT PLAINTO_TSQUERY('russian', 'иванов иван')`
88	Функция TO_TSQUERY подготовки слов разделенных операторорами & (AND), | (OR), ! (NOT), и (или) <-> (FOLLOWED BY)	`SELECT TO_TSQUERY('russian', 'иванов & иван');`
89	Функция PHRASETO_TSQUERY подготавки неформатированного текста с сохранением порядка слов	`SELECT PHRASETO_TSQUERY('russian', 'иванов иван')`
90	Функция WEBSEARCH_TO_TSQUERY подготавки неформатированного текста с сохранением порядка слов или без него. Поддерживает операторы OR и - (NOT)	`SELECT WEBSEARCH_TO_TSQUERY('russian', 'иванов иван OR петр')`
91	Оператор @@ сравнения текстового вектора с подготовленным или неподготовленным текстом	`SELECT column FROM t WHERE (TO_TSVECTOR('russian', lastname) @@ PLAINTO_TSQUERY('russian', 'иванов иван')` `SELECT column FROM t WHERE (PLAINTO_TSQUERY('russian', 'иванов иван' @@ TO_TSVECTOR('russian', lastname))` `SELECT column FROM t WHERE (lastname @@ PLAINTO_TSQUERY('russian', 'иванов иван')`

2 Поддержка функции LISTAGG

2.1 LISTAGG

2.1.1 Описание

Функция LISTAGG объединяет значения measure_column для каждой группы на основе order_by_clause.

2.1.2 Поддержка в модулях

- Сервис исполнения запросов;
- ПОДД-адаптер – Модуль MPPR.

2.1.3 Синтаксис

Функция LISTAGG в модуле «ПОДД-адаптер – Модуль MPPR»

Для ПОДД-адаптер – Модуль MPPR реализована поддержка LISTAGG (expression, separator) [WITHIN GROUP (order_by_clause)].

Пример запроса:

```
{"requestId":"4ec61462-0cf5-4b41-84a8-70f215f4109c","subRequestId":"567fbc0a-04b9-43c8-819b-882d3414c2b1","datamartMnemonic":"demo_view","replyTo":"","sql":"SELECT LISTAGG(firstname, ', ') WITHIN GROUP (ORDER BY firstname) AS ¥"firstname_Listing¥" FROM v1_passenger","parameters":[],"tableParams":[],"isForEstimation":false,"rowCountThreshold": 0}
```


Пример ответа:


```
key = [{"requestId": "4ec61462-0cf5-4b41-84a8-70f215f4109c", "subRequestId": "567fbc0a-04b9-43c8-819b-882d3414c2b1", "replyTo": "", "chunkNumber": 1, "isLastChunk": true, "streamNumber": 1, "streamTotal": 1, "uncompressedSize": 0, "isFragmented": false}],

value = [{"firstname_listing": "Григорий, Иван10, Иван11, Иван5, Иван6, Иван7, Иван8, Иван9, Станислав, Станислав"}]
```


Функция LISTAGG в «Сервисе исполнения запросов»

Для Сервис исполнения запросов реализована поддержка функции LISTAGG для работы с множественными атрибутами. А(10) Р(10) Д(1) = 21 Простор: А(1) Р(3) Т(2) =6

Пример запроса:


```
{"requestId":"d58b1fa4-e674-4698-857f-f2fb779c9245","subRequestId":"6861b2a8-6821-424b-8b1f-ced06fba75a0","datamartMnemonic":"demo_view","replyTo":"","sql":"SELECT LISTAGG(firstname, ', ') WITHIN GROUP (ORDER BY firstname) AS ¥"firstname_Listing¥" FROM v1_passenger limit 10","parameters":[],"tableParams":[],"isForEstimation":false,"rowCountThreshold": 0}
```


Пример ответа:


```
key = [{"requestId": "d58b1fa4-e674-4698-857f-f2fb779c9245", "subRequestId": "6861b2a8-6821-424b-8b1f-ced06fba75a0", "replyTo": "", "chunkNumber": 1, "isLastChunk": true, "streamNumber": 1, "streamTotal": 1, "uncompressedSize": 0, "isFragmented": false}],

value = [{"firstname_listing": "Григорий, Иван10, Иван11, Иван5, Иван6, Иван7, Иван8, Иван9, Станислав, Станислав"}]
```

ПРИЛОЖЕНИЕ 3. ПРИМЕР XML-ФАЙЛА СО СТРУКТУРОЙ ВИТРИНЫ

```
<?xml version='1.0' encoding='utf-8'?>
<ns:PODDMetadataRequest
       xmlns:ns="urn://x-artefacts-podd-gosuslugi-local/metadata/datamart/2/1.6.0"
       xmlns:ns1="urn://x-artefacts-podd-gosuslugi-local/metadata/types/1.3">
   <ns:requestId>00000000-0000-0000-0000-000000000001</ns:requestId>
   <ns:metadata>
       <ns1:datamart>
           <ns1:id>1806436d-437a-400d-b32e-aa15c1a2d4bc</ns1:id>
           <ns1:mnemonic>demo_view</ns1:mnemonic>
           <ns1:description>demo_view</ns1:description>
           <ns1:tenantId>c52f062e-af97-4a44-a33f-d1a94024d0cf</ns1:tenantId>
           <ns1:version>
               <ns1:major>1</ns1:major>
               <ns1:minor>0</ns1:minor>
           </ns1:version>
           <ns1:supportedFrom>2021-01-01T00:00:00</ns1:supportedFrom>
           <ns1:datamartClass>
               <ns1:id>4c4ff97b-938b-4db6-9f4d-ae21046e4d20</ns1:id>
               <ns1:mnemonic>Passenger</ns1:mnemonic>
               <ns1:description>Passenger</ns1:description>
               <ns1:classAttribute>
                   <ns1:id>6fe29bdb-7db1-405a-a05c-b49c541c92bd</ns1:id>
                   <ns1:mnemonic>Code</ns1:mnemonic>
                   <ns1:description>Code</ns1:description>
                   <ns1:type>LONG</ns1:type>
               </ns1:classAttribute>
               <ns1:classAttribute>
                   <ns1:id>e590e7b3-b611-4891-bbd1-a5e256105e73</ns1:id>
                   <ns1:mnemonic>Id</ns1:mnemonic>
                   <ns1:description>Id</ns1:description>
                   <ns1:type>STRING</ns1:type>
               </ns1:classAttribute>
               <ns1:classAttribute>
                   <ns1:id>c97a8102-6ad0-4dbd-934d-c82b83a4d83f </ns1:id>
                   <ns1:mnemonic>FirstName</ns1:mnemonic>
                   <ns1:description>FirstName</ns1:description>
                   <ns1:type>STRING</ns1:type>
               </ns1:classAttribute>
               <ns1:classAttribute>
                   <ns1:id>d2312bfb-7ec0-4c95-9026-0f6dea48c5d9</ns1:id>
                   <ns1:mnemonic>MiddleName</ns1:mnemonic>
                   <ns1:description>MiddleName</ns1:description>
                   <ns1:type>STRING</ns1:type>
               </ns1:classAttribute>
               <ns1:classAttribute>
                   <ns1:id>7b63db89-bd0e-4c92-8bc0-e609175937b9</ns1:id>
                   <ns1:mnemonic>LastName</ns1:mnemonic>
                   <ns1:description>LastName</ns1:description>
                   <ns1:type>STRING</ns1:type>
               </ns1:classAttribute>
               <ns1:classAttribute>
                   <ns1:id>8f3e7f95-f66b-4d4a-b2eb-55a3e6134c3e</ns1:id>
                   <ns1:mnemonic>Birthday</ns1:mnemonic>
                   <ns1:description>Birthday</ns1:description>
                   <ns1:type>DATE</ns1:type>
               </ns1:classAttribute>
```

```
            <ns1:classAttribute>
                <ns1:id>e3658240-b405-4838-99af-d32cd063c463</ns1:id>
                <ns1:mnemonic>Passport</ns1:mnemonic>
                <ns1:description>Passport</ns1:description>
                <ns1:type>STRING</ns1:type>
            </ns1:classAttribute>
            <ns1:primaryKey>
                <ns1:id>6fe29bdb-7db1-405a-a05c-b49c541c92bd</ns1:id>
                <ns1:mnemonic>Code</ns1:mnemonic>
                <ns1:description>Code</ns1:description>
                <ns1:type>
                    <ns1:id>00000000-0000-0000-0000-000000000001</ns1:id>
                    <ns1:value>LONG</ns1:value>
                </ns1:type>
            </ns1:primaryKey>
        </ns1:datamartClass>
        <ns1:datamartClass>
            <ns1:id>cafe41db-3878-4796-ba60-cbd54f042c63</ns1:id>
            <ns1:mnemonic>Ticket</ns1:mnemonic>
            <ns1:description>Ticket</ns1:description>
            <ns1:classAttribute>
                <ns1:id>bc90563b-168a-4faa-9394-7b7390dd0d92</ns1:id>
                <ns1:mnemonic>Id</ns1:mnemonic>
                <ns1:description>Id</ns1:description>
                <ns1:type>STRING</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>ac93618f-752b-44d5-a77c-23a3c9eb069b</ns1:id>
                <ns1:mnemonic>PassengerId</ns1:mnemonic>
                <ns1:description>PassengerId</ns1:description>
                <ns1:type>STRING</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>51355519-2d59-426e-b199-9589930acaaa</ns1:id>
                <ns1:mnemonic>TripId</ns1:mnemonic>
                <ns1:description>TripId</ns1:description>
                <ns1:type>STRING</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>fe92c245-929e-4684-b9c9-22bda6939c09</ns1:id>
                <ns1:mnemonic>Number</ns1:mnemonic>
                <ns1:description>Number</ns1:description>
                <ns1:type>LONG</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>4a32ded4-c970-4874-b0b1-2e3eed8b6483</ns1:id>
                <ns1:mnemonic>ByCard</ns1:mnemonic>
                <ns1:description>ByCard</ns1:description>
                <ns1:type>BOOLEAN</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>35f59c80-fcc3-483c-9cd3-dc3afb606d66</ns1:id>
                <ns1:mnemonic>Price</ns1:mnemonic>
                <ns1:description>Price</ns1:description>
                <ns1:type>DOUBLE</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>8b46ff55-6853-458c-851d-6e1666da918b</ns1:id>
                <ns1:mnemonic>Sold</ns1:mnemonic>
                <ns1:description>Sold</ns1:description>
                <ns1:type>TIMESTAMP</ns1:type>
            </ns1:classAttribute>
```

```
            <ns1:primaryKey>
                <ns1:id>fe92c245-929e-4684-b9c9-22bda6939c09</ns1:id>
                <ns1:mnemonic>Number</ns1:mnemonic>
                <ns1:description>Number</ns1:description>
                <ns1:type>
                    <ns1:id>00000000-0000-0000-0000-000000000001</ns1:id>
                    <ns1:value>LONG</ns1:value>
                </ns1:type>
            </ns1:primaryKey>
        </ns1:datamartClass>
        <ns1:datamartClass>
            <ns1:id>76268090-60ee-4960-8268-1b91f4186e87</ns1:id>
            <ns1:mnemonic>Trip</ns1:mnemonic>
            <ns1:description>Trip</ns1:description>
            <ns1:classAttribute>
                <ns1:id>bd173e24-ea7e-4869-9d43-9f57f5b0a82f</ns1:id>
                <ns1:mnemonic>Id</ns1:mnemonic>
                <ns1:description>Id</ns1:description>
                <ns1:type>STRING</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>1ed32816-8bdb-4d35-9f66-8c08df13ad28</ns1:id>
                <ns1:mnemonic>Number</ns1:mnemonic>
                <ns1:description>Number</ns1:description>
                <ns1:type>INTEGER</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>78f587fa-b53e-4912-b631-0c4a249d20b6</ns1:id>
                <ns1:mnemonic>Duration</ns1:mnemonic>
                <ns1:description>Duration</ns1:description>
                <ns1:type>STRING</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>1750c564-20a7-4e07-988a-b382227123e4</ns1:id>
                <ns1:mnemonic>Length</ns1:mnemonic>
                <ns1:description>Length</ns1:description>
                <ns1:type>FLOAT</ns1:type>
            </ns1:classAttribute>
            <ns1:primaryKey>
                <ns1:id>1ed32816-8bdb-4d35-9f66-8c08df13ad28</ns1:id>
                <ns1:mnemonic>Number</ns1:mnemonic>
                <ns1:description>Number</ns1:description>
                <ns1:type>
                    <ns1:id>00000000-0000-0000-0000-000000000002</ns1:id>
                    <ns1:value>INTEGER</ns1:value>
                </ns1:type>
            </ns1:primaryKey>
        </ns1:datamartClass>
    </ns1:datamart>
</ns:metadata>
</ns:PODDMetadataRequest>
```

ПРИЛОЖЕНИЕ 4. ЭКСПЛУАТАЦИЯ CSV-UPLOADER

1 Инструкция по эксплуатации CSV-Uploader

1.1 Общие правила формата загружаемых CSV-файлов

Общие правила формата загружаемых CSV-файлов приведены в Таблица 11.1.

Таблица 11.1 Общие правила формата загружаемых CSV-файлов

Параметр	**Значение**
Разделитель строк	Любой вариант из: CR/LF (0x0D0A), CR (0x0D), LF (0x0A)
Разделитель полей	по настройке csv-parser/separator (Параметры конфигурации)
Строка заголовка	да (обязательно)
Порядок полей в строке	определяется строкой заголовка
Ограничитель текстового поля	по настройке csv-parser/quote-char (Параметры конфигурации)
Символ маскировки в текстовом поле	по настройке csv-parser/escape-char (Параметры конфигурации)
Обнаружение значения null	До релиза 1.5.0 (включительно): по настройке csv-parser/field-as-null (Параметры конфигурации) начиная с релиза 1.10.0: в текущей реализации парсера данная настройка не поддерживается
Кодирование символов	UTF-8
Десятичный разделитель	символ `.` (0x2E), может не указываться для целых значений
Формат даты	любой из: `dd.MM.yyyy`, `yyyy-MM-dd`
Формат времени	любой из: `HH:mm:ss`, `H:mm:ss`
Формат даты-времени	до релиза 1.5.0(включительно) любой из: `yyyy-MM-dd HH:mm:ss`, `dd.MM.yyyy HH:mm:ss` начиная с релиза 1.10.0 любой из: `yyyy-MM-dd HH:mm:ss.000000`, `dd.MM.yyyy HH:mm:ss.000000`

1.2 Загрузка структуры Витрины

> Внимание:
>
> XML-файл со структурой Витрины может быть загружен только один раз после установки «Витрина данных Лайт».

Для передачи xml-файла со структурой Витрины, выполните следующие действия:

1. Откройте программный интерфейс **CSV-uploader**.
2. Выберите вкладку **Загрузка структуры**.
3. В открывшемся окне **Загрузка структуры Витрины** нажмите кнопку **Выберите файл**, выберите XML-файла для загрузки и нажмите кнопку **Загрузить**. (см. Рисунок - 11.1)

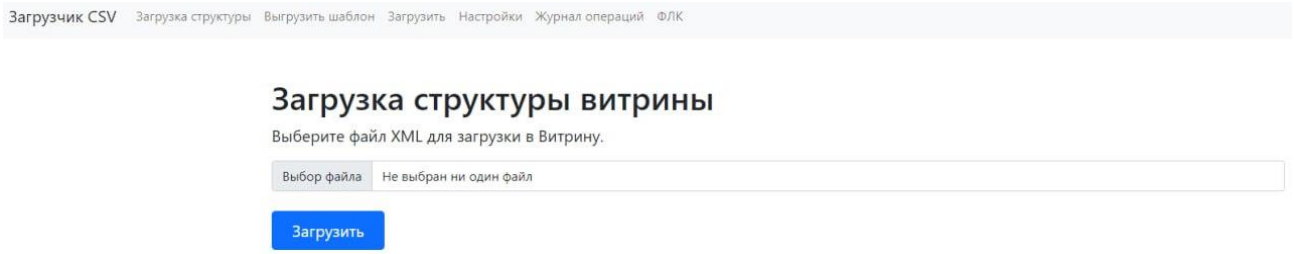

Рисунок - 11.1 Загрузка структуры Витрины

В случае успешного применения настроек отобразится информационное сообщение: Список таблиц загружен.

1.3 Выгрузка шаблона CSV

Для выгрузки существующего CSV-файла со структурой Витрины, выполните следующие действия:

1. Откройте программный интерфейс **CSV-uploader**.
2. Выберите вкладку **Выгрузка шаблона CSV**.
3. Выберите таблицу для выгрузки, например, **demo_view_podd.all_types_table**, для выгрузки примера CSV-таблиц
4. для **СМЭВ4** (см. Рисунок - 11.2).

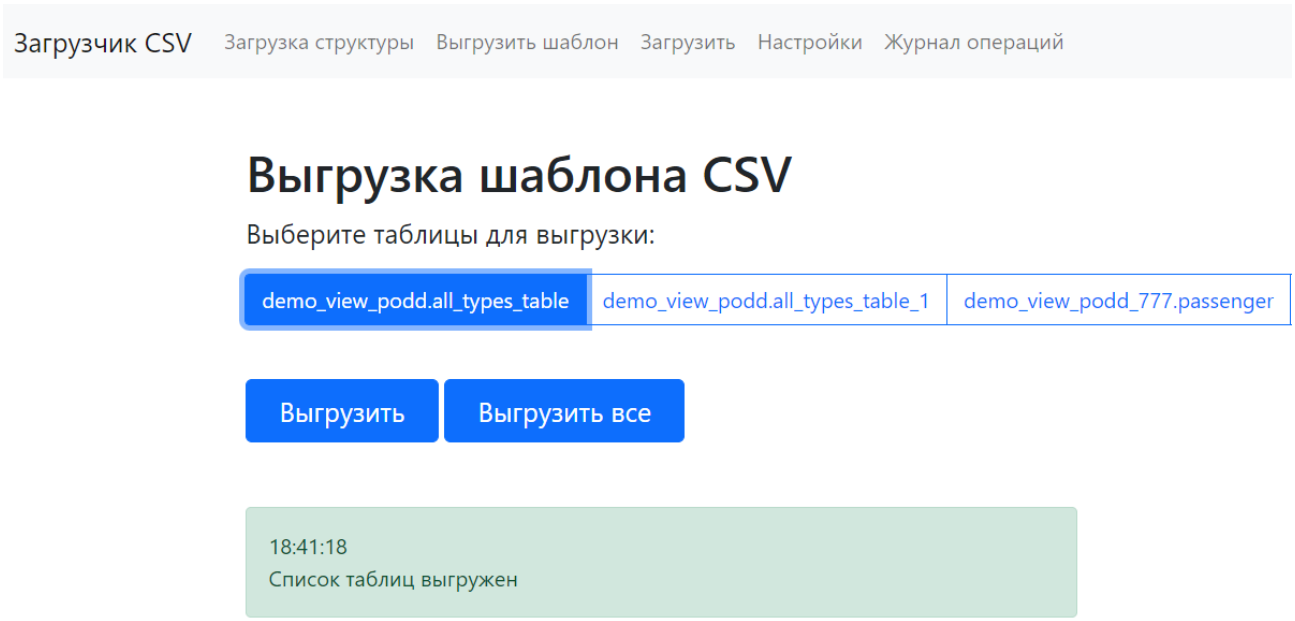

Рисунок - 11.2 Выгрузка шаблона CSV

4. Нажмите кнопку **Выгрузить**. Файл будет загружен на локальный компьютер. Если требуется выгрузить все таблицы, нажмите кнопку **Выгрузить все**.

В случае успешной выгрузки на экране монитора отобразится информационное сообщение: Список таблиц выгружен.

1.4 Загрузка CSV-файла

Для загрузки CSV-файла, выполните следующие действия:

1. Откройте программный интерфейс **CSV-uploader**.

2. Выберите вкладку **Загрузчик CSV**.
3. В открывшемся окне **Загрузка файла** выберите **Режим** загрузки:
   - **Вставка** - параметр определяет, что данные будут добавлены.
   - **Удаление** - параметр определяет, что данные будут удалены.

В случае, если в настройках модуля **CSV-uploader** включен ФЛК и прописан адрес модуля **REST-Uploader**, на странице отображается переключатель с текстом «Выполнять проверку форматно-логического контроля».

1. Для автоматического определения типа таблиц включите переключатель **Автоматическое определение таблицы**, если автоматическое определение таблиц не требуется, выключите переключатель и выберите таблицу, в которую требуется внести изменения, например, **demo_view_podd.all_types_table** (см. Рисунок - 11.3).

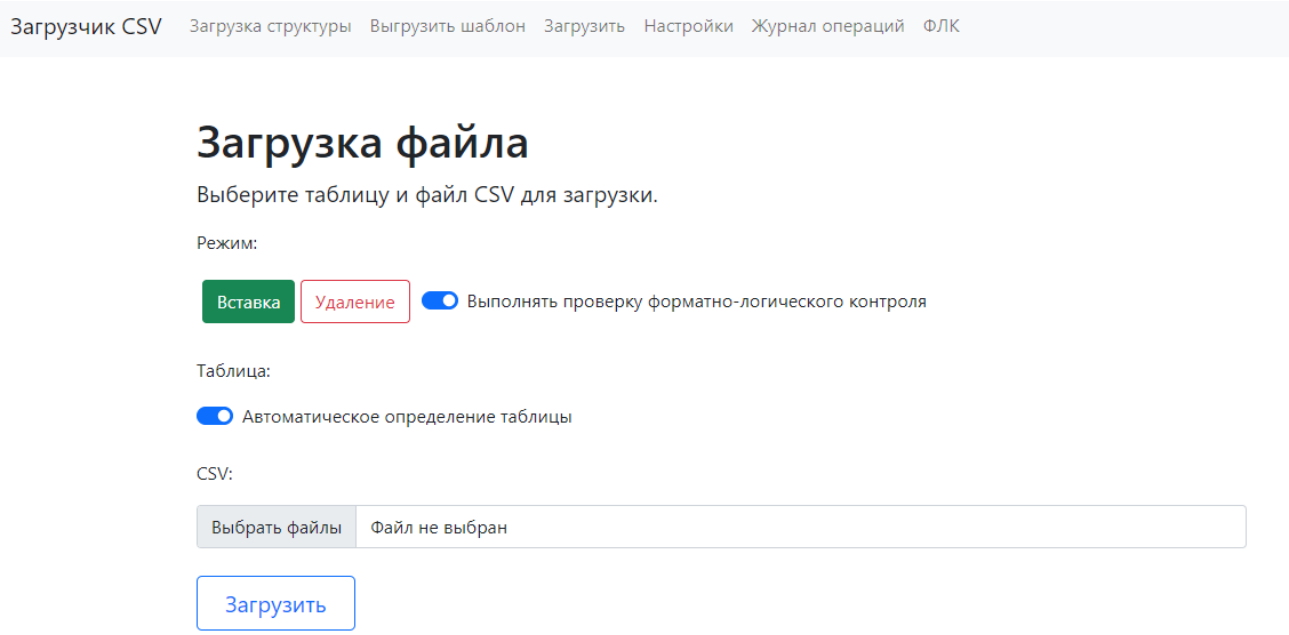

Рисунок - 11.3 Загрузка CSV-файла

2. Нажмите кнопку **Выберите файл** чтобы выбрать файл для загрузки.
3. Нажмите кнопку **Загрузить**.
4. Убедитесь, что файл с таблицами был загружен.

При включенной настройке определения таблиц после выбора и загрузки файла:

- модулем CSV-Uploader определяется таблица загрузки:
  - в случае, если активен переключатель «Автоматическое определение таблицы» по метаданным csv файла;
  - в случае если выбрана конкретная таблица, в соответствии с выбором пользователя;
- модуль CSV-Uploader обогащает url запроса на загрузку именем датамарта и таблицы;
- модуль CSV-Uploader выполняет запрос /v2/datamarts/{datamart_name}/tables/{table_name}/upload к модулю REST-Uploader;
- модуль CSV-Uploader отображает текст ответа на странице загрузки в формате:
  - время ответа;
  - код ответа;

- body ответа:
- в случае успешного ответа, модуль CSV-Uploader сохраняет requestId файла в топик flk_logs в внутренней Kafka.

1.5 Загрузка CSV-файла с предварительным форматно-логическим контролем

В случае, если в настройках модуля CSV-Uploader включена настройка `VALIDATION_ENABLE: true` и прописан адрес модуля REST-Uploader (`REST_UPLOADER_URL`) при активном режиме «Вставка» на странице отображается переключатель с текстом «Выполнять проверку форматно-логического контроля» , по умолчанию значение `вкл` (`true`) см. Рисунок - 11.4.

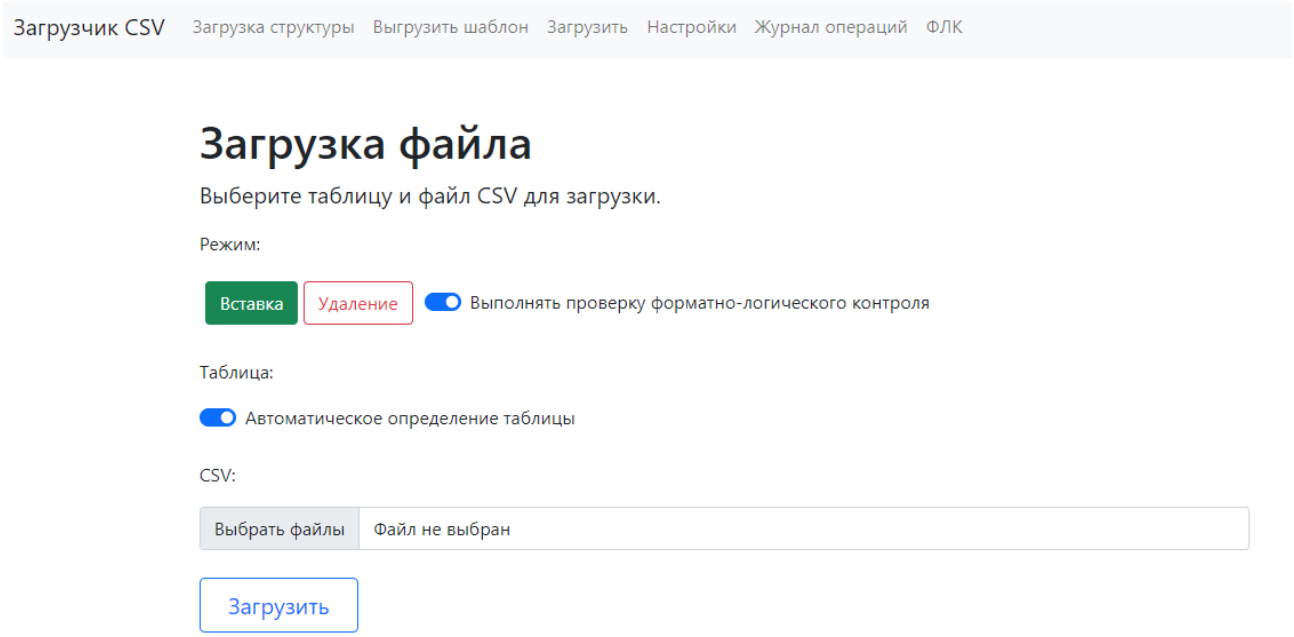

Рисунок - 11.4 Переключатель выполнения ФЛК

1. При включенном переключателе «Выполнять проверку форматно-логического контроля» после выбора файла и нажатия кнопки **Загрузить**:
   - модулем CSV-Uploader определяется таблица загрузки:
     - в случае, если включен переключатель «автоопределение таблицы» по метаданным CSV файла;
     - в случае если выбрана конкретная таблица, в соответствии с выбором пользователя;
   - модуль CSV-Uploader обогащает URL запроса на загрузку именем датамарта и таблицы;
   - модуль CSV-Uploader выполняет запрос `/v2/datamarts/{datamart_name}/tables/{table_name}/upload` к модулю REST-Uploader;
   - модуль CSV-Uploader отображает текст синхронного ответа на странице загрузки в формате:
     - время ответа;
     - код ответа;
     - body ответа.

2. При выключенном переключателе «Выполнять проверку форматно-логического контроля» загрузка выполняется стандартным способом через модуль CSV-Uploader.

1.6 Обязательная загрузка данных с предварительным форматно-логическим контролем

При включении настройки `VALIDATION_MANDATOR: true`, переключатель «Выполнять проверку форматно-логического контроля» неактивен и находится во включенном положении.

В данном режиме загрузка данных в ручном режиме с использованием CSV-Uploader невозможна, для всех загружаемых данных будут проводиться проверки форматно-логического контроля в модуле REST-Uploader см. Рисунок - 11.5.

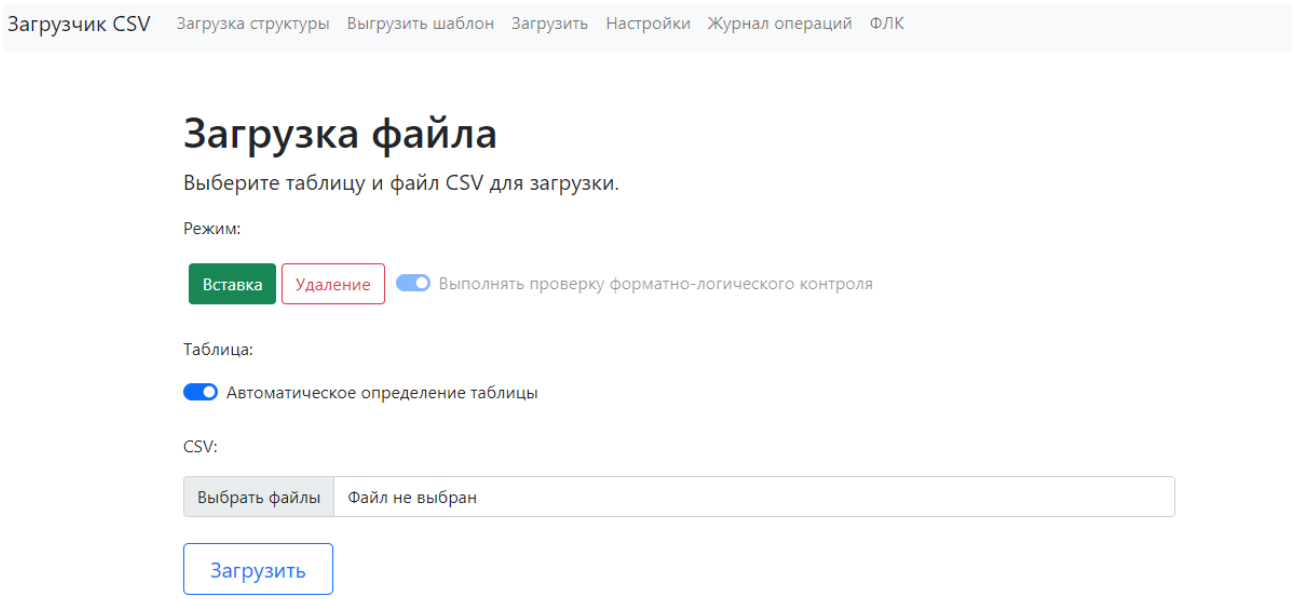

Рисунок - 11.5 Обязательная загрузка данных с предварительным форматно-логическим контролем

1.7 Аутентификация с использованием jwt-токена при включенной аутентификации в модуле REST-Uploader

Для использования jwt-токена при загрузке данных с предварительным ФЛК в случае, если в REST-Uploader включена аутентификация, необходимо включить следующие настройки в модуле CSV-Uploader:

- `VALIDATION_ENABLE:true`;
- `JWT_AUTH:true`.

В случае, если обе настройки имеют значение true, при открытии загрузчика CSV-uploader отображается модальное окно ввода токена пользователя, а на странице загрузки данных в витрину отображается поле «Изменить JWT» (см. Рисунок - 11.6).

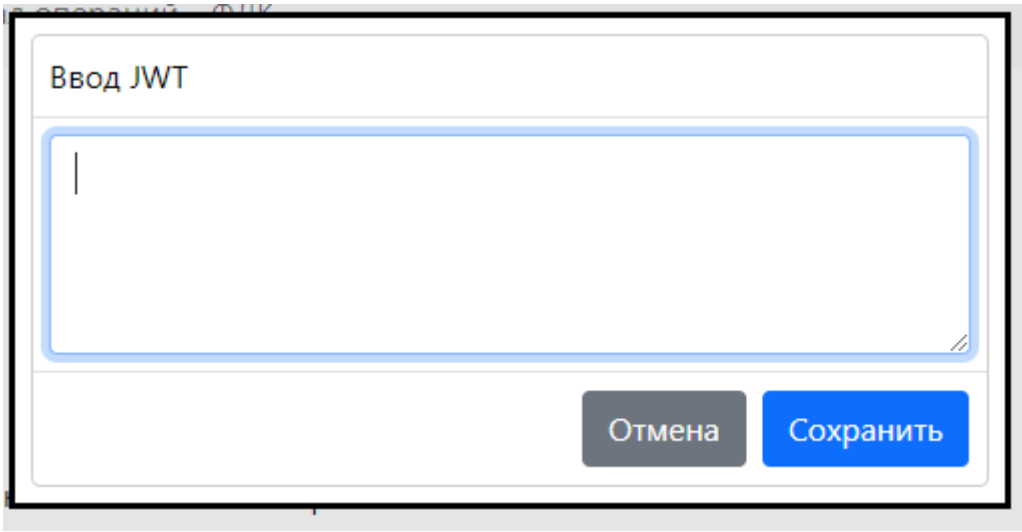

Рисунок - 11.6 Модальное окно ввода токена

Значение внесенного JWT-токена используется как барьерный токен при обращении к REST-Uploader.

Внесенное значение токена сохраняется в сессии пользователя и автоматически подставляется при включении переключателя выполнения ФЛК проверок. Для того, чтобы изменить JWT-токен для аутентификации, необходимо нажать кнопку **Изменить JWT** (см. Рисунок - 11.7).

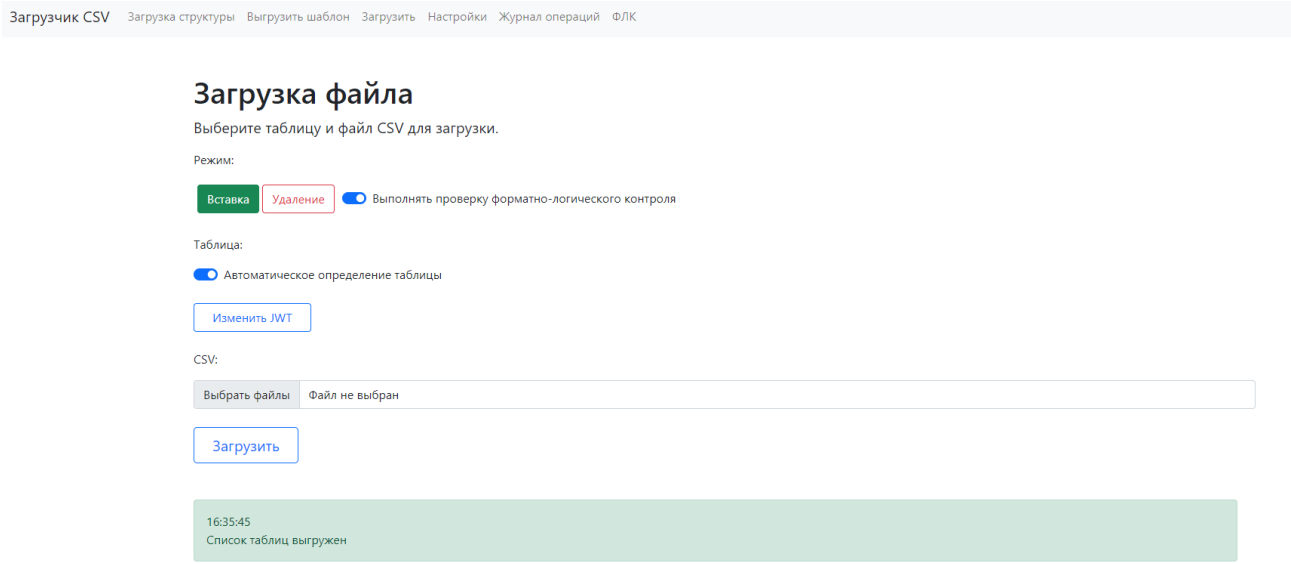

Рисунок - 11.7 Отображение кнопки **Изменить JWT**

1.8 Настройки CSV-uploader

Для CSV-uploader можно настроить следующие параметры:

- автоматический запуск загрузки CSV-файлов по расписанию;
- количество отображаемых записей для *Журнала операций*.

1.9 Автоматический запуск загрузки CSV-файлов по расписанию

Для настройки автоматического запуска загрузки CSV-файлов по расписанию, выполните следующие действия:

1. Откройте программный интерфейс **CSV-uploader**.
2. Выберите вкладку **Настройки**.
3. В открывшемся окне **Настройки** в поле **Запуск по расписанию**, укажите время в `Cron` формате (например, `0 15 10? * *` - загрузка файлов будет происходить каждый день в 10.15) и путь к каталогу с CSV-файлами (см. Рисунок - 11.8) .

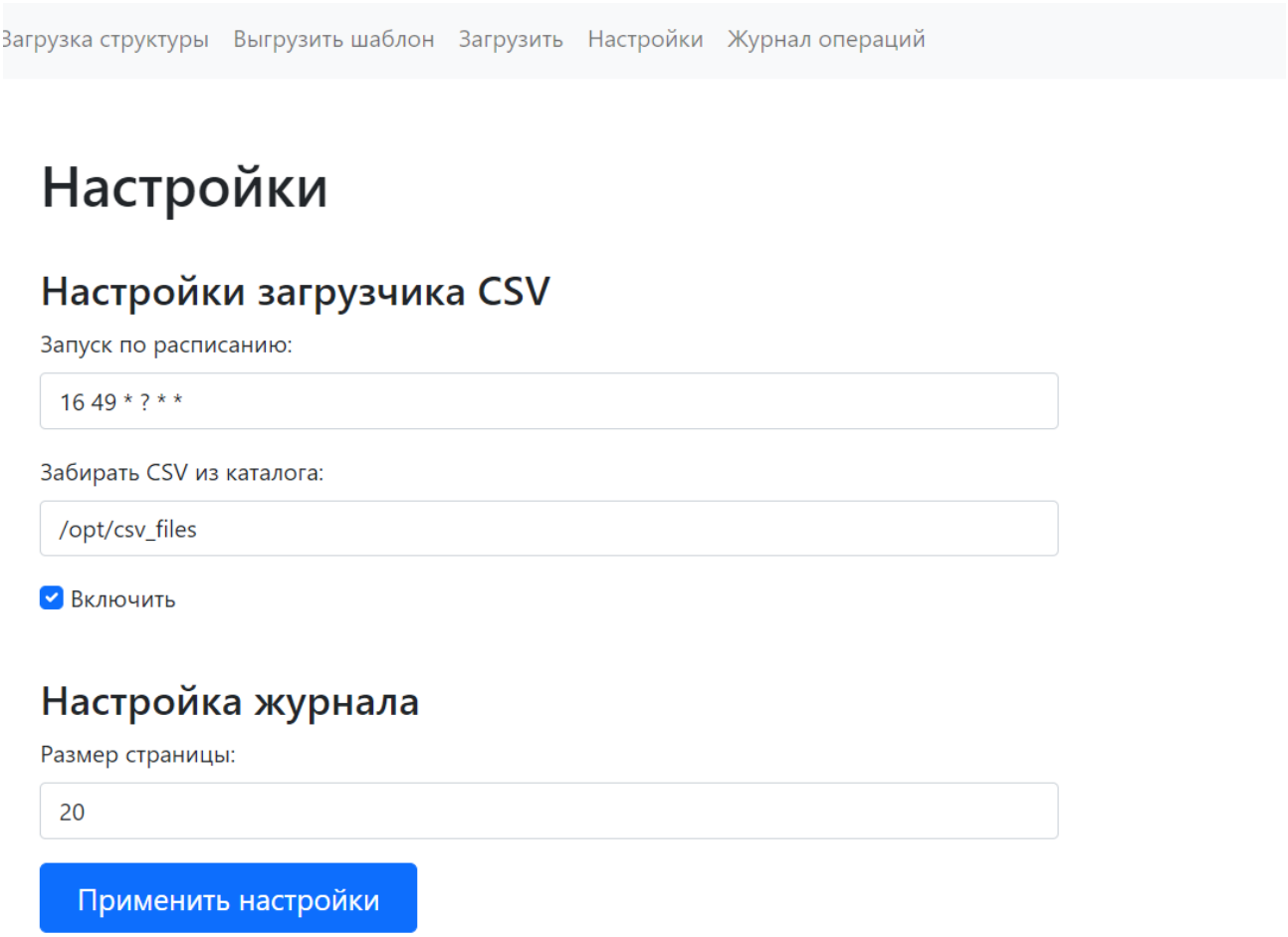

Рисунок - 11.8 Автоматический запуск загрузки CSV-файлов по расписанию

1. Установите маркер в поле **Включить**, для активации автоматического запуска загрузки.
2. Нажмите кнопку **Применить настройки**.

В случае успешного применения настроек отобразится информационное сообщение: *Конфигурация успешно получена*.

1.10 Настройка Журнала операций

Для настройки *Журнала операций*, выполните следующие действия:

1. Откройте программный интерфейс **CSV-uploader**.
2. Выберите вкладку **Настройки**.
3. В открывшемся окне **Настройки** в поле **Размер страницы**, укажите количество записей на страницу, например, `20` (см. Рисунок - 11.9).

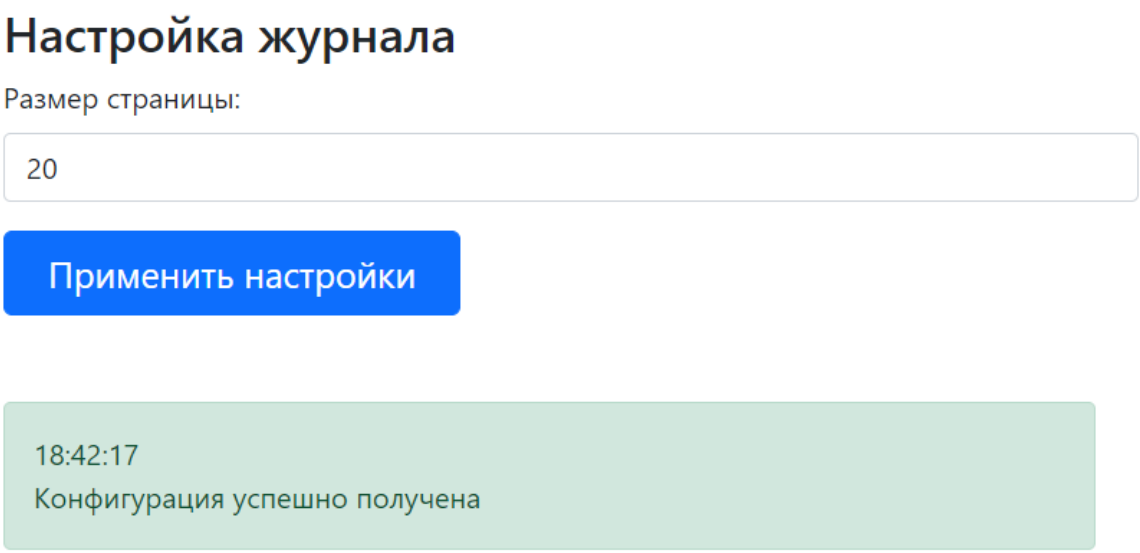

Рисунок - 11.9 Настройка *Журнала операций*

1. Нажмите кнопку **Применить настройки**.

В случае успешного применения настроек отобразится информационное сообщение: *Конфигурация успешно получена*.

1.11 Просмотр Журнала операций

В *Журнале операций* можно просмотреть действия выполненные в **CSV-uploader**:

- Время - время, когда операция была выполнена.
- Уровень - статус операции.
  - `ERROR` - ошибка загрузки;
  - `INFO` - описание операции.
- Сообщение - краткое информационное сообщение об операции.

Для просмотра *Журнала операций*, выполните следующие действия:

1. Откройте программный интерфейс **CSV-uploader**.
2. Выберите вкладку **Журнал операций**.
3. В открывшемся окне просмотрите операции, которые были выполнены в **CSV-uploader** (см. Рисунок - 11.10).

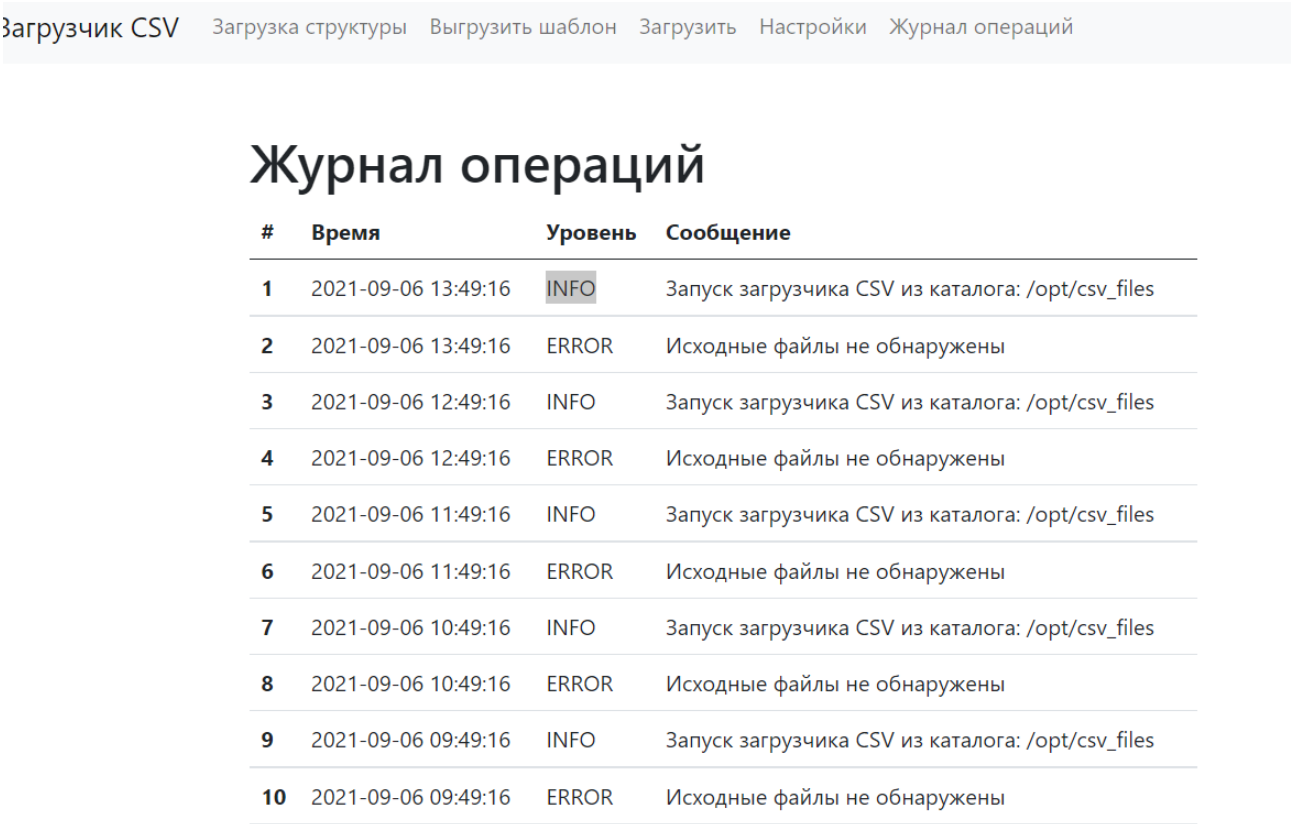

Рисунок - 11.10 Просмотр *Журнала операций*

4. Нажмите кнопку **Применить настройки**.

1.12 Интерфейс Форматно-логического контроля

На вкладке **Форматно-логический контроль** (см. `flk`) отображается:

- список последних отправленных файлов, определяемых настройками модуля CSV-Uploader - значение по умолчанию 20:
  - список requestId;
  - время записи в кафку;
  - статус загрузки файла;
- управляющий элемент для запроса отчета об ошибках для файла (кнопка отображается активной только в случае финальных статусов):
  - статус 3;
  - статус 4;
  - статус 7;
- элементы пагинации списка requestId.

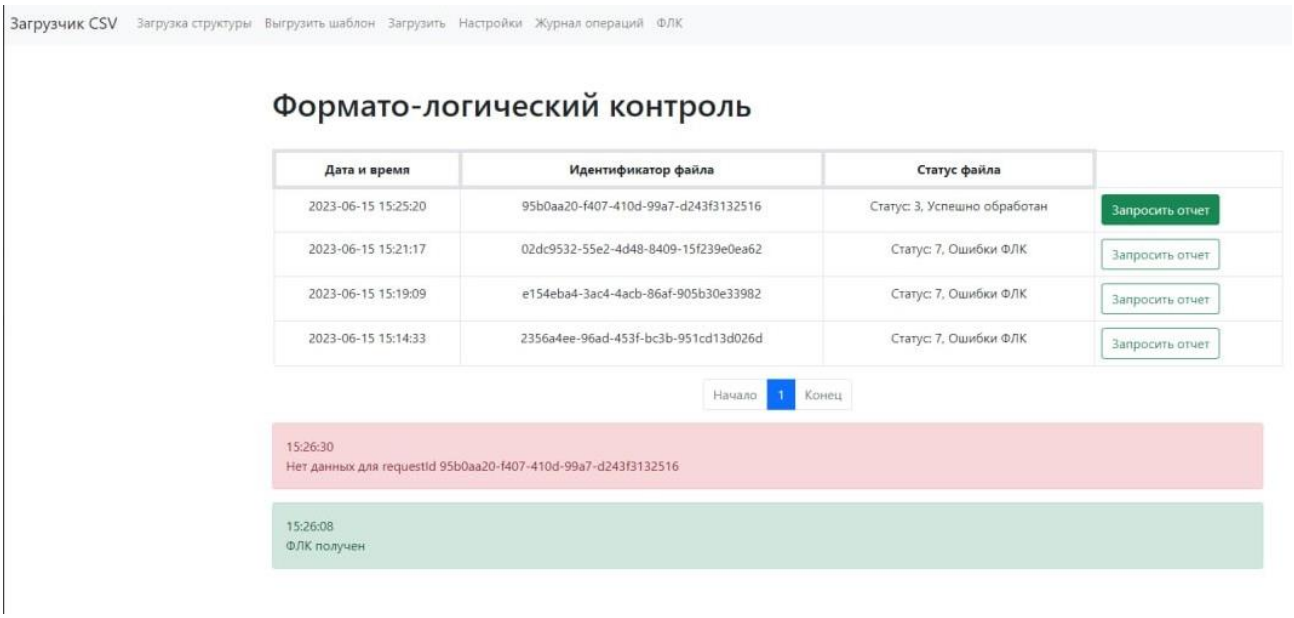

Рисунок - 11.11 Форматно-логический контроль

При нажатии на кнопку запроса отчета об ошибках для файла, модуль CSV-Uploader:

- вызывает метод `/v2/requests/{request_id}/report/`;
- получает CSV файл с именем `report_requestId.csv`;
- в зависимости от ответа:
  - если **response 200 ok**: скачивает файл на ПК пользователя автоматически или при нажатии на название отчета с выводом сообщения о загрузке файла;
  - если **response 400** - выводит сообщение Нет данных.

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

ADCM

Arenadata Cluster Manager (ADCM) - Универсальный оркестратор гибридного ландшафта. Он позволяет быстро устанавливать, настраивать все data-сервисы компании и управлять ими. Наиболее ярко преимущества ADCM раскрываются при работе с гетерогенной инфраструктурой, при которой появляется возможность размещать data-сервисы на различных типах инфраструктур: в облаке, on-premise или в качестве PaaS-сервисов.

ADS

Arenadata Streaming (ADS) - Масштабируемая отказоустойчивая система для потоковой обработки данных в режиме реального времени на базе Apache Kafka и Apache Nifi.

Airflow

открытое программное обеспечение для создания, выполнения, мониторинга и оркестровки потоков операций по обработке данных.

Apache

Организация-фонд, способствующая развитию проектов программного обеспечения Apache.

Apache Airflow

Платформа для программного создания, планирования и мониторинга рабочих процессов.

Apache Avro

Линейно-ориентированный (строчный) формат передачи наборов данных, используемый в качестве платформы сериализации, разрабатываемый в рамках фонда Apache.

Apache Hadoop

Свободно распространяемый набор утилит, библиотек и фреймворк для разработки и выполнения распределённых программ, работающих на кластерах из сотен и тысяч узлов.

Apache Kafka

Распределённый программный брокер сообщений, проект с открытым исходным кодом, разрабатываемый в рамках фонда Apache.

Apache Spark

Фреймворк с открытым исходным кодом для реализации распределённой обработки неструктурированных и слабоструктурированных данных.

API

Application programming interface (англ.) - Программный интерфейс приложения, описание сервисов взаимодействия компьютерной программы с другими программами.

BLOB-адаптер

Информационно-технологический компонент Витрины, обеспечивающий чтение бинарных файлов из **Хранилища BLOB-объектов ведомства**.

ClickHouse

Колоночная аналитическая СУБД с открытым кодом, которая позволяет выполнять аналитические запросы в режиме реального времени на структурированных больших данных, разрабатывается компанией Яндекс.

Counter-Provider

Сервис генерации уникального номера.

CSV

Comma-Separated Values (англ.) - текстовый формат, предназначенный для представления табличных данных.

CSV-extractor

Специализированное программное обеспечение, которое извлекает данные из csv-файлов в собственную БД-хранилища сервиса **Tarantool**.

CSV-Uploader

Программный модуль Витрины данных, который предназначен для загрузки csv-файлов в Витрину данных.

DAG

Файл, содержащий блок данных.

DATA-uploader

Модуль исполнения асинхронных заданий.

DBeaver

Клиентское приложение для управления базами данных (БД), которое использует программный интерфейс **JDBC** для взаимодействия с реляционными БД через драйвер **JDBC**.

DDL

Data definition language (англ.) - семейство компьютерных языков, используемых в компьютерных программах для описания структуры баз данных.

DNS

Domain Name System «система доменных имён» - компьютерная распределённая система для получения информации о доменах. Чаще всего используется для получения IP-адреса по имени хоста (компьютера или устройства), получения информации о маршрутизации почты и/или обслуживающих узлах для протоколов в домене.

Docker

Программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений.

Docker Compose

Платформа контейнеризации, предназначена для конфигурирования многоконтейнерных приложений. В Docker Compose можно управлять несколькими контейнерами **Docker**.

Endpoint

Шлюз (в переводе с англ. — конечная точка), который соединяет серверные процессы приложения с внешним интерфейсом. Простыми словами, это адрес, на который отправляются сообщения (работает с API).

ETL

Extract, transform, load (англ.) - решение, используемое при выгрузке данных из различных источников ведомств и дальнейшего хранения их в Витрине **ProStore** для чтения, использования и взаимодействия с другими ведомствами.

FileZilla

FTP-клиент.

Grafana

Веб-приложение для аналитики и интерактивной визуализации показателей мониторинга с открытым исходным кодом.

Greenplum

Массово-параллельная СУБД для хранилищ данных на основе PostgreSQL.

HikariCP

Hikari Connection Pool.

HTTP

HyperText Transfer Protocol (англ.) - протокол прикладного уровня передачи данных, в настоящий момент используется для передачи произвольных данных.

IAM

Сервисы управления идентификацией и контролем доступа (Identity&AccessManagement).

JDBC

Java DataBase connectivity (англ.) - платформенно-независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД.

JDBC-драйвер

Библиотека классов, реализующая стандарт JDBC и подключения к источнику данных с использованием специализированного протокола, поддерживаемого источником данных.

JDBC-extractor

Специализированное программное обеспечение, которое извлекает данные из jdbc-источника (ведомства) в собственную БД-хранилища сервиса (**Tarantool**).

JSON

JavaScript Object Notation - Общий формат для представления значений и объектов в соответствии со стандартом RFC 4627.

Kafka-loader

Специализированное программное обеспечение, которое загружает данные, извлеченные и приведенные в соответствие логической структуре данных Витрины, собственно в Витрину.

Loki

Приложение для агрегирования log-файлов, используется совместно с **Prometheus**.

MD5

128-битный алгоритм хеширования. Предназначен для создания «отпечатков» или дайджестов сообщения произвольной длины и последующей проверки их подлинности.

MPP

Массово-параллельная архитектура (*англ. massive parallel processing*, MPP, также «массивно-параллельная архитектура»).

NTP

Network Time Protocol — сетевой протокол для синхронизации внутренних часов компьютера с использованием сетей с переменной латентностью.

OpenAPI

The OpenAPI Specification (англ.) – Формализованная спецификация и экосистема множества инструментов, предоставляющая интерфейс между front-end системами, кодом библиотек низкого уровня и коммерческими решениями в виде API.

ProStore

Интеграционная система, обеспечивающая единый интерфейс к хранилищу разнородных данных. Определяет структуры данных, запись и чтение данных Витрины. Позволяет работать со входящими в состав хранилища СУБД одинаковым образом, используя единый синтаксис запросов SQL и единую логическую схему данных.

Prostore

Ядро интеграционной системы ProStore, сервис исполнения запросов.

Prometheus

Программное приложение, используемое для мониторинга событий и оповещения, которое записывает метрики в реальном времени в базу данных временных рядов, построенную с использованием модели HTTP-запроса, с гибкими запросами и оповещениями в режиме реального времени.

Proxy API

Проксирование запросов через Datamart Studio к инсталляциям приложений Витрин данных.

PSQL

Терминальный клиент для работы с PostgreSQL.

PuTTY

Свободно распространяемый клиент для различных протоколов удалённого доступа, включая SSH, Telnet, rlogin.

PXF

Фреймворк, позволяющий **ADB** (Greenplum) параллельно обмениваться данными со сторонними системами.

REST

Representational state transfer (англ.) – архитектурный стиль взаимодействия компонентов распределенного приложения в сети.

REST-адаптер

Сервис, реализующий публикацию конечных точек API для обработки запросов с использованием спецификации OpenAPI версии 3. Используется для сохранения обратной совместимости получения данных из ведомства по REST.

REST API

Набор правил, по которым различные программы могут взаимодействовать между собой и обмениваться данными с помощью протокола HTTP.

REST-Uploader

Модуль асинхронной загрузки данных из сторонних источников.

SOAP

(от англ. Simple Object Access Protocol — простой протокол доступа к объектам) — протокол обмена структурированными сообщениями в распределённой вычислительной среде.

SQL

Structured query language (англ.) – язык структурированных запросов. Декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных.

SQL-запрос

Запрос к Витрине данных Поставщика. Произвольный или регламентированный запрос к данным, сформулированный на языке SQL.

SSH

Secure Shell (англ.) – «безопасная оболочка». Сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и туннелирование TCP-соединений.

Tarantool

Платформа in-memory вычислений с гибкой схемой данных для создания высоконагруженных приложений. Включает в себя базу данных и сервер приложений на Lua.

UDP

Протокол передачи данных. С UDP компьютерные приложения могут посылать сообщения другим хостам по IP-сети без необходимости предварительного сообщения для установки специальных каналов передачи или путей данных.

URI

Унифицированный идентификатор ресурса. URI — последовательность символов, идентифицирующая абстрактный или физический ресурс.

UUID

Стандарт идентификации, используемый в создании программного обеспечения, стандартизированный Open Software Foundation как часть DCE — среды распределённых вычислений. Основное назначение UUID — это позволить распределённым системам уникально идентифицировать информацию без центра координации.

Vert.x

Библиотека для разработки асинхронных приложений, основанная на событиях.

VipNet

программное обеспечение (далее - ПО) для защиты сетевого трафика на рабочих местах пользователей.

XML

eXtensibe Markup Language (англ.) – универсальный текстовый формат для хранения и передачи структурированных данных.

XML-extractor

Специализированное программное обеспечение, для копирования данных из xml-файлов в собственную

БД-хранилища сервиса (**Tarantool**).

ZooKeeper

Сервер с открытым исходным кодом для высоконадежной распределенной координации облачных приложений.

Агент СМЭВ4 (Агент)

Типовое программное обеспечение, устанавливаемое в контуре ИС УВ и обеспечивающее сопряжение Витрин данных и ИС УВ с Ядром СМЭВ4.

База данных

Совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

(Большой) Двоичный объект (BLOB / БЛОБ)

Тип данных, значение которого представляет собой массив байт, размер которого существенно превышает размер базовых скалярных типов (int, float, double, date)

Брокер сообщений

Архитектурный паттерн в распределённых системах; приложение, которое преобразует сообщение по одному протоколу от приложения-источника в сообщение протокола приложения-приёмника, тем самым выступая между ними посредником.

Витрина данных

Комплекс программных и технических средств в составе информационно-телекоммуникационной инфраструктуры Участника взаимодействия, обеспечивающий хранение и предоставление данных другим Участникам взаимодействия с использованием СМЭВ4.

Вид сведения СМЭВ (ВС)

Комплекс документальных и программных компонентов, зарегистрированный в СМЭВ 3.x, обеспечивающий взаимодействие ИС ведомств в определённом формате и по определённым правилам.

ГОСТ

Нормативно-правовой документ, в соответствии требованиями которого производится стандартизация производственных процессов.

Дельта

Логически целостная совокупность изменений информации об объектах. Каждой дельте поставлено в соответствие целое число из монотонно возрастающей последовательности целых чисел начиная с 0, отражающее ее место в общей последовательности дельт и дата-время ее исполнения.

ЕИП

Единая информационная платформа.

ИС

Информационная система.

ИС УВ

Информационная система Участника взаимодействия.

КриптоПро

Разработанная одноименной компанией линейка криптографических утилит (вспомогательных программ) — так называемых криптопровайдеров. Они используются в других программах для генерации электронной подписи (ЭП), работы с сертификатами, организации структуры PKI и т.д.

ЛК УВ

Личный кабинет участника взаимодействия. Система, предназначенная для управления информационными системами и мониторинга информационных обменов в СМЭВ 3 и СМЭВ 4 участниками взаимодействия.

Логическая модель данных

Схема базы данных, выраженная в понятиях бизнес-требований.

Мнемоника Витрины

Уникальное строковое значение, определяющее модель данных Витрины.

Модель данных Витрины

Описание структуры Витрины (общая информация, перечень сущностей, атрибутный состав), загруженное в Ядро СМЭВ4.

Набор данных

Совокупность систематизированных данных (датасетов), представляющих собой базовый элемент для работы с данными.

НСУД

Национальная система управления данными.

ОГРН

Основной государственный регистрационный номер, присваивается юридическим лицам сразу же после регистрации в ФНС РФ.

Параметр запроса

Символическое имя, входящее в текст SQL-запроса и не содержащееся в Модели данных Витрины, в терминах которой сформулирован SQL-запрос.

ПО

Программное обеспечение.

СМЭВ4-адаптер

Программно-технический продукт, обеспечивающий взаимодействие витрины и СМЭВ4.

СМЭВ4-адаптер - Модуль исполнения запросов

Логический модуль СМЭВ4-адаптера, предназначен для исполнения запросов СМЭВ4 (через протокол коммуникации Агент СМЭВ4).

СМЭВ4-адаптер - Модуль MPPR

Логический модуль СМЭВ4-адаптера, предназначен для чтения данных в многопоточном режиме (massively parallel processing, **MPP**).

СМЭВ4-адаптер - Модуль MPPW

Логический модуль СМЭВ4-адаптера выполняет загрузку данных в многопоточном режиме.

Подписка (потребителя)

Предоставление права Потребителю данных СМЭВ4 на информационный обмен с использованием Регламентированного запроса типа «Рассылка».

Поставщик данных

Участник взаимодействия, являющийся источником данных для других участников и использующий СМЭВ4 для передачи данных.

Потребитель данных

Участник взаимодействия, получающий данные от Поставщиков данных для дальнейшей их обработки и использующий для передачи запросов и получения данных СМЭВ4.

Распределенный запрос

Регламентированный запрос, инициированный Потребителем, SQL-выражение которого содержит наборы данных из двух или более Витрин данных.

Регламентированный SQL-запрос (РЗ)

SQL-запрос, выраженный в терминах Модели данных, загруженной в СМЭВ4, и зарегистрированный в Ядре СМЭВ4 под символической мнемоникой, используемой ИС Потребителя СМЭВ4 для выполнения регламентированного запроса. Может иметь параметры, значения которых задаются Потребителем данных СМЭВ4 при выполнении регламентированного запроса.

Реплика

СУБД, хранящая реплицируемые наборы данных, полученные от Поставщика данных.

Сервис Формирования документов

Модуль витрины, предназначенный для работы с формируемыми документами.

СМЭВ

Система межведомственного электронного взаимодействия.

СМЭВ 3

Единая система межведомственного электронного взаимодействия, функционирующая в соответствии с Методическими рекомендациям по работе со СМЭВ версии `3.x`.

СМЭВ3-адаптер

Информационно-технологический компонент СМЭВ, устанавливается на стороне Участника взаимодействия. СМЭВ3-адаптер обеспечивает информационное взаимодействие через единый электронный сервис единой системы межведомственного электронного взаимодействия (**СМЭВ**).

СМЭВ4

единый сервис доступа к данным СМЭВ, предназначенный для автоматизации процесса передачи данных и уведомлений об изменении данных между организациями или органами власти, ответственными за формирование и ведение информационных ресурсов, зарегистрированных в НСУД.

Сообщение

Сведения в виде законченного блока данных, передаваемые при функционировании информационной системы.

СУБД

Система управления базами данных.

Табличный параметр (запроса)

Параметр, значение которого представляет собой двумерный массив с именованными колонками и неупорядоченными строками. Формальный табличный параметр может использоваться в инструкциях `FROM`, `JOIN` как источник данных.

Токен

Ключ безопасности (Цифровой сертификат).

Участник взаимодействия

Орган или организация, участвующий в информационном обмене через СМЭВ.

ФЛК

Форматно-логический контроль загружаемых в Витрину данных.

Хранилище BLOB-объектов

Место для хранения BLOB-объектов (бинарных данных). Располагается на стороне ведомства и не является частью Витрины данных. Взаимодействие с Хранилищем BLOB-объектов осуществляется через **BLOB-адаптер**.

Хранилище S3 (объектное хранилище S3)

Хранилище бинарных объектов, позволяющее хранить файлы любого типа и объема. Доступ к хранилищу предоставляется через API.

Чанк

Фрагмент результирующих данных оптимального для передачи по сети размера.