

ИНФРАСТРУКТУРА ЭЛЕКТРОННОГО ПРАВИТЕЛЬСТВА

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

Руководство администратора Компонента «Витрина данных»

Версия 2.2.0

Листов 176

Москва, 2025

СОДЕРЖАНИЕ

1 Общие сведения о Компоненте	7
1.1 Назначение Компонента	7
1.2 Возможности Компонента.....	7
1.3 Технические и программные средства.....	8
2 Настройка Компонента	9
2.1 Настройка технических средств	9
2.2 Настройка программных средств	9
2.2.1 Настройка ProStore	9
2.2.1.1 Настройка Сервиса исполнения запросов (query-execution).....	9
2.2.1.2 Настройка коннекторов.....	9
2.2.2 Настройка СМЭВ QL Сервера.....	9
2.2.2.1 Конфигурирование сервера	9
2.2.3 Настройка СМЭВ3-адаптера	22
2.2.3.1 Конфигурация СМЭВ3-адаптер (application.yml)	22
2.2.3.2 Параметры конфигурации	26
2.2.4 Настройка CSV-Uploader	36
2.2.4.1 Конфигурация CSV-uploader (application.yml)	36
2.2.4.2 Параметры конфигурации	37
2.2.5 Настройка DATA-Uploader – Модуль исполнения асинхронных заданий	42
2.2.5.1 Конфигурация модуля DATA-Uploader (application.yml).....	42
2.2.5.2 Параметры конфигурации	44
2.2.6 Настройка REST-Uploader – Модуль асинхронной загрузки данных из сторонних источников.....	53
2.2.6.1 Конфигурация модуля REST-Uploader (application.yml).....	53
2.2.6.2 Параметры конфигурации	56
2.2.6.3 Проверка форматно-логического контроля.....	66
2.2.6.4 Статусная модель.....	68
2.2.6.5 Спецификация модуля асинхронной загрузки данных из сторонних источников	69
2.2.7 Настройка BLOB-адаптера	80
2.2.7.1 Конфигурация BLOB-адаптера (application.yml)	80
2.2.7.2 Параметры конфигурации	82
2.2.8 Настройка Сервиса формирования документов	87
2.2.8.1 Конфигурация Сервиса Формирования документов (application.yml).....	87
2.2.8.2 Параметры конфигурации	89
2.2.8.3 Примеры rebble-шаблонов для Сервиса Формирования документов.....	94
2.2.9 Настройка Counter-provider - Сервиса генерации уникального номера	96

2.2.9.1 Конфигурация модуля Counter-Provider (application.yml)	96
2.2.9.2 Пример файла application.yml	96
2.2.9.3 Параметры конфигурации	97
2.2.10 Настройка Arenadata Cluster Manager (ADCM)	101
2.2.11 Настройка сервиса журналирования.....	101
2.2.12 Установка компонента сбора данных запросов и ответов Витрины данных	106
2.2.12.1 Процесс установки	106
2.2.13 Настройка Агента СМЭВ4.....	110
2.2.13.1 Настройка взаимодействия Компонента с Агентом СМЭВ4	110
2.3 Настройка сервиса мониторинга	111
2.3.1 Предоставление источника данных	111
2.3.2 Предоставление информационной панели.....	113
2.3.2.1 Настройка конфигурационного файла Prometheus.....	115
2.3.2.2 Health check	117
3 Запуск и остановка Компонента	119
3.1 Prostore	119
3.1.1 Запуск.....	119
3.2 СМЭВ QL Сервер.....	119
3.2.1 Быстрый старт	119
3.2.1.1 Создание и конфигурация.....	119
3.2.1.2 Запуск и управление.....	119
3.2.1.3 Работа с сервером	120
3.2.1.4 Сборка проекта	121
3.3 СМЭВ3-адаптер.....	122
3.3.1 Запуск модуля.....	122
3.3.2 Остановка модуля	122
3.4 CSV-Uploader.....	122
3.4.1 Запуск CSV-uploader.....	122
3.4.2 Остановка модуля	122
3.5 DATA-uploader – Модуль исполнения асинхронных заданий.....	123
3.5.1 Запуск модуля.....	123
3.5.2 Остановка модуля	123
3.6 REST-Uploader – Модуль асинхронной загрузки данных из сторонних источников	123
3.6.1 Запуск модуля.....	123
3.6.2 Остановка модуля	124
3.6.3 Добавление поставщика данных	124
3.7 BLOB-адаптер	125
3.7.1 Запуск модуля.....	125

3.7.2 Остановка модуля	125
3.8 Сервис формирования документов.....	125
3.8.1 Запуск модуля.....	125
3.8.2 Остановка модуля	125
3.9 ETL	126
3.9.1 Apache Airflow	126
3.9.1.1 Запуск	126
3.9.1.2 Остановка	126
3.9.2 Apache Spark	126
3.9.2.1 Запуск	126
3.9.2.2 Остановка	126
3.9.3 Apache Hadoop	127
3.9.3.1 Запуск	127
3.9.3.2 Остановка	127
3.9.4 Tarantool (Vynil).....	127
3.9.4.1 Запуск	127
3.9.4.2 Остановка	128
3.10 Counter-provider - Сервис генерации уникального номера	128
3.10.1 Запуск модуля.....	128
3.10.2 Остановка модуля	128
3.11 Arenadata Cluster Manager (ADCM)	129
3.11.1 Запуск.....	129
3.11.2 Остановка	129
4 Бекапирование Компонента «Витрина данных»	130
4.1 Реализация бекапирования в слое адаптеров Компонента «Витрина данных»	130
4.1.1 Работа модулей для обеспечения резервного копирования.....	130
4.1.1.1 Сообщения в топиках команд.....	130
4.1.1.2 Статусы модулей.....	130
4.2 Механизм приостановки модулей, требующих консистентности с Prostore.....	131
4.2.1 Приостановка модулей, требующих консистентности с Prostore	131
4.2.2 Восстановление модулей, требующих консистентности с Prostore.....	131
4.3 Механизм резервного копирования и восстановления из резервной копии в модулях слоя адаптеров.....	131
4.3.1 Механизм резервного копирования модулей слоя адаптеров.....	131
4.3.2 Механизм восстановления из резервной копии модулей слоя адаптеров	132
4.4 Поведение в случае ошибок при выполнении резервного копирования	133
4.4.1 Ошибки резервного копирования и восстановления из резервной копии	133
5 Дополнительные возможности	136
5.1 Дополнительные возможности конфигурации Стандарт	136

5.1.1 Установки опциональных приложений	136
5.1.2 Материализованные представления.....	136
5.1.3 Маршрутизация запросов к материализованным представлениям	141
5.1.4 Логирование	142
5.1.5 Миграция из Bare metal варианта установки в Kubernetes	142
5.1.5.1 Примеры инструкций по развертыванию Prostore в Kubernetes.....	143
5.2 Дополнительные возможности конфигурации Лайт	149
5.2.1 Логирование	150
5.2.2 Проверка версии компонентов	150
6 Сообщения администратору.....	151
6.1 Сообщения в ходе установки и настройки Компонента.....	151
6.2 Сообщения при эксплуатации Компонента	151
7 Метрики в модулях Компонента «Витрина данных».....	154
Приложение 1. Эксплуатация CSV-Uploader	157
1 Инструкция по эксплуатации CSV-Uploader.....	157
1.1 Общие правила формата загружаемых CSV-файлов	157
1.2 Загрузка структуры Витрины	157
1.3 Выгрузка шаблона CSV.....	158
1.4 Загрузка CSV-файла	158
1.5 Загрузка CSV-файла с предварительным форматно-логическим контролем	159
1.6 Обязательная загрузка данных с предварительным форматно-логическим контролем	160
1.7 Аутентификация с использованием jwt-токена при включенной аутентификации в модуле REST-Uploader	161
1.8 Настройки CSV-uploader.....	162
1.9 Автоматический запуск загрузки CSV-файлов по расписанию	162
1.10 Настройка Журнала операций	163
1.11 Просмотр Журнала операций	164
1.12 Интерфейс Форматно-логического контроля.....	165
Приложение 2. Пример XML-файла со структурой витрины.....	167
Термины и определения	170

АННОТАЦИЯ

В данном программном документе приведено Руководство администратора программного обеспечения «Витрина данных НСУД» (далее – Компонент, Программа), 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 Компоненте» указаны назначение и возможности Компонента и сведения о технических и программных средствах, обеспечивающих выполнение данного Компонента.

В разделе «Настройка Компонента» приведены описания действий по настройке Компонента, а также технических и программных средств.

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

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

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

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

В «Приложении 1. Эксплуатация CSV-Uploader» приведена инструкция по эксплуатации CSV-Uploader.

В «Приложении 2. Пример XML-файла со структурой витрины» приведен пример XML-файла со структурой витрины.

1 ОБЩИЕ СВЕДЕНИЯ О КОМПОНЕНТЕ

1.1 Назначение Компонента

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 – СМЭВ4), обеспечивающих транспорт и процессинг данных, а также агентов СМЭВ4, 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. Настоящая документация описывает применение Компонента «Витрина данных».

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

1.2 Возможности Компонента

Внимание:

Начиная с версии ПО 2.0.0 не работают подписки.

Ниже приведены возможности Компонента для конфигураций Компонента:

- Стандарт;
- Лайт.

Возможности конфигурации Стандарт

Компонент обеспечивает выполнение следующих задач:

- описание логической модели данных;
- настройка Компонента и структуры таблиц в ее БД для хранения публикуемых данных;

- загрузка и хранение публикуемых данных в БД Компонента;
- извлечение данных из внешних систем (внешних ИС по отношению к Витрине данных НСУД);
- выполнение запросов в соответствии с протоколом СМЭВ4 через механизмы СМЭВ4:
 - поддержка протокола коммуникации Агента СМЭВ4;
 - генерация формируемых документов на основании публикуемых данных.
- 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;
- журналирование событий функциональных блоков;
- мониторинг информации о работоспособности экземпляра Компонента.

Возможности конфигурации Лайт

Компонент обеспечивает выполнение следующих задач:

- автоматическая настройка взаимосвязей между модулями Компонента;
- автоматический запуск всех необходимых модулей Компонента 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 СМЭВ4 через механизмы СМЭВ4.

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 Компонента приведены в разделе [Рекомендуемые технические и программные средства](#) документа «Техническое описание системы».

2 НАСТРОЙКА КОМПОНЕНТА

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 Компонента «Витрина данных»», 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 Компонента, процесс установки и проверка корректной установки Компонента 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 модулей 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 writer 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`

```
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: "$STATES_DIR:states"
regulated-query:
  directory: "$QUERIES_DIR:queries"

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 # redis/postgres
  pool: # Настройка подключений
    - host: 127.0.0.1
      port: 6379
      database: "" # имя БД, используется для адаптера postgres
      schema: "" # схема БД, используется для адаптера postgres
  max-pool-size: 20 # Максимальный размер пула соединений
  user: "" # Пользователь для подключения к redis/postgres
  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
  omit-empty-array: false # Настройка, позволяющая исключить пустые строки массивов
из ответа
  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 влючения 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 steveql, 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: "misdms.readable_book"
#   writable-table: "misdms.writable_book"
#   anchor: "update_at"
#   soft-delete: "delete_at"

# Массив описания proxy таблиц
proxy-tables: [ ]
# Пример описания
# proxy-tables:
# - table: "misdms.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

```

```

request-timeout: 1m # Таймаут на исполнение http запроса на доставку push
уведомления потребителю
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:
  type: KAFKA # KAFKA | HTTP
  status-event-topic:
    topic: "$PS_STATUS_EVENT_TOPIC:status.event"
  property:
    bootstrap.servers: "$PS_KAFKA:localhost:9092"
    group.id: smeysql-server-status-event
    auto.offset.reset: earliest
retry:
  max-attempts: 30 # Количество попыток отправки нотификации
  min-period: 5s # Минимальный период ожидания перед повторной попыткой
  max-period: 10s # Максимальный период ожидания перед повторной попыткой
queue:
  max-attempts: 30 # Количество попыток отправки уведомлений из персистентной
очереди
  min-period: 1m # Минимальный период ожидания перед повторной попыткой
отправки уведомления из персистентной очереди
  max-period: 3m # Максимальный период ожидания перед повторной попыткой
отправки уведомления из персистентной очереди
  scan-waiting-period: 1m # Период сканирования отложенных уведомлений
process:
  check-expired-period: 5m # Период проверки уведомлений взятых в работу и
просроченных
  timeout: 5m # Предельное время обработки уведомления из очереди
poisoned:
  check-expired-period: 1h # Период проверки времени жизни "отравленных"
уведомлений
  timeout: 7d # Время хранения уведомлений в poisoned-очереди

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

agent: # Параметры конфигурирования агента ПОДД
  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

cls: # Конфигурация отправки событий в СЦЛ
  enabled: false # Признак включения отправки событий в СЦЛ
  url: "http://127.0.0.1:8192/api/v1/cls/event" # Адрес для отправки событий СЦЛ

```

```

# Настройки модуля сбора информации о компонентах витрины
component-info:
  enabled: true
  # Источник в папке задаваемой настройкой ${sources.directory}
  source: prostore
  # 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:
    - storage.user
    - storage.password
    - storage-queue.user
    - storage-queue.password

```

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

- **ktor** - настройки подключения фреймворка Ktor;
- **sources** - определение директории хранения источников;
- **models** - определение директории хранения моделей;
- **states** - определение директории хранения состояний;
- **regulated-query** - определение директории хранения регламентированных запросов;
- **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 дельта витрин данных;
- **state** - установка ограничений в машинах-состояний;
- **index_recommendations** - рекомендации по аналитике;
- **standalone-tables** - массив описания standalone таблиц;
- **proxy-tables** - массив описания proxy таблиц;
- **push** - настройки отправки push уведомлений;
- **storage-queue** - настройки хранилища очередей;
- **agent** - Параметры конфигурирования агента СМЭВ4;
- **signature** - блок настроек механизма подписания;
- **cls** - конфигурация отправки событий в СЦЛ;
- **component-info** - настройки модуля сбора информации о компонентах витрины.

2.2.2.1.1.1 Секция ktor

В секции **ktor** настраиваются параметры подключения фреймворка Ktor, например:

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

Параметры конфигурации

- **port** - порт хоста развертывания фреймворка;
- **modules** - модули приложения.

2.2.2.1.1.2 Секция sources

В секции **sources** определяется директория хранения источников, например:

```
sources:
  directory: "$SOURCES_DIR:sources"
```

Параметры конфигурации

- **directory** - директория хранения источников.

2.2.2.1.1.3 Секция models

В секции **models** определяется директория хранения моделей, например:

```
models:
  directory: "$MODELS_DIR:models"
```

Параметры конфигурации

- **directory** - директория хранения моделей.

2.2.2.1.1.4 Секция states

В секции **states** определяется директория хранения состояний, например:

```
states:
  directory: "$STATES_DIR:states"
```

Параметры конфигурации

- **directory** - директория хранения состояний.

2.2.2.1.1.5 Секция regulated-query

В секции **regulated-query** определяется директория хранения регламентированных запросов, например:

```
regulated-query:
  directory: "$QUERIES_DIR:queries"
```

Параметры конфигурации

- **directory** - директория хранения состояний.

2.2.2.1.1.6 Секция swagger

В секции **swagger** хранится настройка файла описания спецификации, например:

```
swagger:
  file: smevql-openapi.yaml
  servers:
    - "http://127.0.0.1:8080/smevql/api/v1"
```

Параметры конфигурации

- **file** - путь к файлу описания спецификации;
- **servers** - адрес сервера.

2.2.2.1.1.7 Секция storage

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

```
storage: # Блок параметров хранения информации
adapter: redis # redis/postgres
pool: # Настройка подключений
- host: 127.0.0.1
port: 6379
database: "" # имя БД, используется для адаптера postgres
schema: "" # схема БД, используется для адаптера postgres
max-pool-size: 20 # Максимальный размер пула соединений
user: "" # Пользователь для подключения к redis/postgres
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.

2.2.2.1.1.8 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!**

2.2.2.1.1.9 Секция request

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

```
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
omit-empty-array: false # Настройка, позволяющая исключить пустые строки массивов
из ответа
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 включения 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

```

Параметры конфигурации

- **strategy** - стратегия исполнения запросов `delegate``| ``atomic`;
- **timeout** - таймаут исполнения запросов;
- **base_path** - префикс для роута запросов. Например, `smevql/api/v1`;
- **max_nested_level** - предельная вложенность запрашиваемых ресурсов;
- **omit-empty-array** - настройка, позволяющая исключить пустые строки массивов из ответа;
- **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

2.2.2.1.1.10 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, сформированные вне дельт, не попадают;
- в результаты запросов к материализованным представлениям данные, сформированные вне дельт, не попадают;
- в рамках подписок данные, сформированные вне дельт, не передаются.

2.2.2.1.1.11 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.

2.2.2.1.1.12 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.

2.2.2.1.1.13 Секция `standalone-tables`

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

Например:

```
standalone-tables: [ ]
# Пример описания
# standalone-tables:
# - readable-table: "misdms.readable_book"
# writable-table: "misdms.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.

2.2.2.1.1.14 Секция `proxy-tables`

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

Например:

```
# Массив описания proxy таблиц
proxy-tables: [ ]
# Пример описания
# proxy-tables:
# - table: "misdms05.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.

2.2.2.1.1.15 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
request-timeout: 1m # Таймаут на исполнение http запроса на доставку push
уведомления потребителю
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:
  type: KAFKA # KAFKA | HTTP
  status-event-topic:
    topic: "$PS_STATUS_EVENT_TOPIC:status.event"
    property:
      bootstrap.servers: "$PS_KAFKA:localhost:9092"
      group.id: smeysql-server-status-event
      auto.offset.reset: earliest
  retry:
    max-attempts: 30 # Количество попыток отправки нотификации
    min-period: 5s # Минимальный период ожидания перед повторной попыткой
    max-period: 10s # Максимальный период ожидания перед повторной попыткой
    queue:
      max-attempts: 30 # Количество попыток отправки уведомлений из персистентной
очереди
      min-period: 1m # Минимальный период ожидания перед повторной попыткой
отправки уведомления из персистентной очереди
      max-period: 3m # Максимальный период ожидания перед повторной попыткой
отправки уведомления из персистентной очереди
      scan-waiting-period: 1m # Период сканирования отложенных уведомлений
    process:
      check-expired-period: 5m # Период проверки уведомлений взятых в работу и
просроченных
      timeout: 5m # Предельное время обработки уведомления из очереди
    poisoned:
      check-expired-period: 1h # Период проверки времени жизни "отравленных"
уведомлений
      timeout: 7d # Время хранения уведомлений в poisoned-очереди
```

Параметры конфигурации

- **notification_path** - шаблон пути агента, на который необходимо отправлять

- нотификации;
- **request-timeout** - таймаут на исполнение HTTP запроса на доставку push уведомления потребителю
- **state_machine_enabled** - признак публикации нотификаций на основе событий стейт машины;
- **status-prostore-enabled** - признак публикации нотификаций на основе событий статусов Prostore;
- **prostore** - настройки Prostore;
- **retry** - настройки попыток отправок нотификаций.

2.2.2.1.16 Секция **storage-queue**

В секции **storage-queue** указываются настройки хранилища очередей, например:

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

Параметры конфигурации

- **host** - хост хранилища;
- **port** - порт хранилища;
- **database** - БД хранилища;
- **schema** - схема хранилища;
- **user** - пользователь хранилища;
- **password** - пароль подключения.

2.2.2.1.17 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.

2.2.2.1.18 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.

2.2.2.1.1.19 Секция cls

В секции **signature** настраивается конфигурация отправки событий в СЦЛ, например:

```
cls:
enabled: false
url: "http://127.0.0.1:8192/api/v1/cls/event"
```

Параметры конфигурации

- **enable** - включение (true), отключение (false) отправки событий в СЦЛ;
- **url** - эндпоинт отправки событий;

2.2.2.1.1.20 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.

2.2.2.1.2 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.yml** и/или **application.yml**) 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

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

spring:
  main:
    allow-bean-definition-overriding: true
```

```

iua: # Блок настроек взаимодействия с сервисом ИУА
  it-system: "" # мнемоника информационной системы
  wsdl-url: http://localhost:7575/ws/?wsdl # адрес к wsdl веб сервиса ИУА
  endpoint-address: "" # Endpoint для запросов к сервису ИУА. По умолчанию указывать
  не требуется
  retry-count: 2
  retry-delay: 500ms

smev:
  # self | iua
  # self- собственная реализация подписи и взаимодействия с транспортом
  # iua- подписание и работа с транспортом через адаптер ИУА
  implementation: self
  #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
  #avaliabe 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:
  receiver-property:
    - selector: # селектор сообщений из смэв
      # используется при smeV3->implementation: self, должно быть пусто или может
      отсутствовать при использовании smeV3->implementation: iua
      namespace: a
      # используется при smeV3->implementation: self, должно быть пусто или может
      отсутствовать при использовании smeV3->implementation: iua
      root-element-name: b
      # используется при smeV3->implementation: iua, должно быть пусто или может
      отсутствовать при использовании smeV3->implementation: self
      router-extra-queue: some_request
      # пепл шаблон, который будет обрабатываться для определенного selector
      template: smeV3-adapter/templates/smeV.xml.peb
      # задержка между запросами, в случае если очередь пуста
      idle-delay: PT1m
      # файл, который будет отправлен в случае ошибки
      fallback-response: smeV3-adapter/templates/fallback.xml
      pebble-refresh: PT5m

```

```

- selector:
  namespace: urn://x-artefacts-testperson/1.0
  root-element-name: TestPersonRequest
  router-extra-queue: some_request
  template: smeV3-adapter/templates/smeV.xml.peb
  idle-delay: PT1m

response-receiver-property:
- selector: # селектор из смэв
  # используется при smeV3->implementation: self, должно быть пусто или может отсутствовать при использовании smeV3->implementation: iua
  namespace: a
  # используется при smeV3->implementation: self, должно быть пусто или может отсутствовать при использовании smeV3->implementation: iua
  root-element-name: b
  # используется при smeV3->implementation: iua, должно быть пусто или может отсутствовать при использовании smeV3->implementation: self
  router-extra-queue: some_request
  # пэбл шаблон, который будет обрабатываться для определенного selector
  template: smeV3-adapter/templates/smeV.xml.peb
  # задержка между запросами, в случае если очередь пуста
  idle-delay: PT1m

persistence-mode: prostore # prostore -default, zookeeper

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

# Параметры витрины персистентности в Prostore, используемой для технического функционала
prostore-persistence:
  persistence-datamart: persistence
  datasource: # по умолчанию пусто, тогда берется единственный датасорс из настроек Простора

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}

# Конфигурация хранилища параметров (для SDB)
paramstorage:
  base-path: '/smeV/paramstorage'
  table-variables: smeV3_adapter_variables

sign:
  #алгоритм подписи файла
  digest-algorithm: 1.2.643.7.1.1.2.2

```

```

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

rest:
  enabled: false # вкл/выкл
  get: /le # путь get запроса
  post: /le # путь post запроса
  template: smeV3-adapter/templates/smev.xml.peb # обрабатываемый шаблон

# рассылка смеv
scheduler:
  templates:
    - enabled: false # вкл/выкл
      interval: PT30s # интервал между запусками
      template: smeV3-adapter/templates/pfr-delta.peb # обрабатываемый шаблон

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

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

http-server:
  port: ${SERVER_PORT:8080}

backup:
  mode: ${BACKUP_MODE:rest} # kafka | rest
  rest:
    uri: ${BACKUP_URI:/backup}
  backupTopic: ${BACKUP_TOPIC:adapter.backup}
  statusTopic: ${STATUS_TOPIC:adapter.status}
  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: {}

# Параметры подключения динамических конфигураций
dynamic-config:
  # Включение процедуры сканирования источника на предмет наличия конфигураций
  enabled: false
  # Интервал проверки источника
  refresh: PT3m
  # Источник подключаемых конфигураций
  table-receiver-scheduler: smeV3_adapter_receiver_scheduler
  # Источник pebble-шаблонов
  table-pebble: smeV3_adapter_pebble

# 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:
    - keys

# Допускаются лишь скалярные строковые значения,
# так например переменная pebble-variables→host станет доступна в pebble как
${host}}
pebble-variables: { }

```

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);
- **iua** - блок настроек взаимодействия с сервисом ИУА;
- **smev** – настройки подключения к **СМЭВ3-адаптеру**;
- **receiver**- настройка взаимодействия СМЭВ-запросов с Pebble-шаблонами (для каждого **receiver** можно настроить количество **instance**);
- **persistence-mode** - настройки хранения данных;
- **prostore-rest-client** - блок параметров конфигурирования взаимодействия с **ProStore**;
- **prostore-persistence** – Параметры витрины персистентности в **Prostore**, используемой для технического функционала;
- **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;
- **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;
- **http-server** - указывается порт веб-сервера;
- **backup** - настройки бекапирования;
- **spf** - параметры подключения к сервису печатных форм (Указывается при использовании функции toSpf);
- **dynamic-config** - параметры подключения динамических конфигураций;
- **component-info** - настройки модуля сбора информации о компонентах витрины.

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:

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

2.2.3.2.2 Секция spring

Секция **spring** подключение к фреймворку [spring boot](#) (используется для разработки).
Например:

```
spring:
  main:
    allow-bean-definition-overriding: true
```

2.2.3.2.3 Секция iua

Секция **iua** хранит блок настроек взаимодействия с сервисом ИУА.
Например:

```
iua:
  it-system: ""
  wsdl-url: http://localhost:7575/ws/?wsdl
  endpoint-address: "" # Endpoint для запросов к сервису ИУА. По умолчанию указывать
не требуется
  retry-count: 2
  retry-delay: 500ms
```

Параметры настроек

- **it-system** - мнемоника информационной системы;
- **wsdl-url** - адрес к wsdl веб сервиса ИУА;
- **endpoint-address** - Endpoint для запросов к сервису ИУА. По умолчанию указывать не требуется;
- **idle-delay** - периодичность опроса очереди СМЭВЗ для получения новых запросов (в формате ISO 8601).
- **retry-count** - кол-во попыток осуществления запросов к сервису;
- **retry-delay** - период задержки в секундах.

2.2.3.2.4 Секция smev

Секция **smev** отвечает за настройки подключения к СМЭВЗ.
Например:

```
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 СМЭВЗ не отвечает на запрос, в новой версии СМЭВЗ-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**).

2.2.3.2.5 Секция **receiver**

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

Например:

receiver:

receiver-property:

```
- selector: # селектор сообщений из смэв
  # используется при smeV3->implementation: self, должно быть пусто или может
  отсутствовать при использовании smeV3->implementation: iua
  namespace: a
  # используется при smeV3->implementation: self, должно быть пусто или может
  отсутствовать при использовании smeV3->implementation: iua
  root-element-name: b
  # используется при smeV3->implementation: iua, должно быть пусто или может
  отсутствовать при использовании smeV3->implementation: self
  router-extra-queue: some_request
  # пепл шаблон, который будет обрабатываться для определенного selector
  template: smeV3-adapter/templates/smeV.xml.peb
  # задержка между запросами, в случае если очередь пуста
  idle-delay: PT1m
  # файл, который будет отправлен в случае ошибки
  fallback-response: smeV3-adapter/templates/fallback.xml
- selector:
  namespace: urn://x-artefacts-testperson/1.0
  root-element-name: TestPersonRequest
  router-extra-queue: some_request
  template: smeV3-adapter/templates/smeV.xml.peb
  idle-delay: PT1m
```

response-receiver-property:

```
- selector: # селектор из смэв
  # используется при smeV3->implementation: self, должно быть пусто или может
  отсутствовать при использовании smeV3->implementation: iua
  namespace: a
  # используется при smeV3->implementation: self, должно быть пусто или может
  отсутствовать при использовании smeV3->implementation: iua
  root-element-name: b
  # используется при smeV3->implementation: iua, должно быть пусто или может
  отсутствовать при использовании smeV3->implementation: self
  router-extra-queue: some_request
  # пепл шаблон, который будет обрабатываться для определенного selector
  template: smeV3-adapter/templates/smeV.xml.peb
  # задержка между запросами, в случае если очередь пуста
```

idle-delay: PT1m

Параметры настроек

- **selector** - селектор сообщений из СМЭВ;
- **namespace** - пространство имен в XML.
- **root-element-name** - имя корневого элемента запроса обрабатываемого ВС (как указано в заявке на регистрацию ВС);
- **router-extra-queue** - очередь роутера, используется при smeV3->implementation: iua, должно быть пусто или может отсутствовать при использовании smeV3->implementation: self;
- **template** имя файла, содержащего rebble-шаблон обработки запросов для данного ВС;
- **idle-delay** - периодичность опроса очереди СМЭВЗ для получения новых запросов (в формате ISO 8601).

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>
```

2.2.3.2.6 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 -default, zookeeper
```

Параметры настроек

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

2.2.3.2.7 Секция prostore-rest-client

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

Например:

```
prostore-rest-client:
  persistence-datamart: persistence
  datasource: # по умолчанию пусто, тогда берется единственный датасорс из настроек Простора
  table-variables: smeV3_adapter_variables
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9195}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}
  default-schema: demo_view
```

Параметры настроек

- `persistence-datamart` - датамарт, где будут располагаться таблицы хранения данных, используется при `persistence-mode = prostore`
- `datasource` - источник данных, например `persistence`;
- `table-variables` - таблица переменных СМЭВ3-адаптера, например `smev3_adapter_variables`;
- `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`.

2.2.3.2.8 Секция `prostore-persistence`

В секции `prostore-persistence` реализован блок параметров витрины персистентности в Prostore, используемой для технического функционала.

Например:

```
prostore-persistence:
  persistence-datamart: persistence
  datasource: # по умолчанию пусто, тогда берется единственный датасорс из настроек Простора
```

Параметры настроек

- `persistence-datamart` - датамарт, где будут располагаться таблицы хранения данных, используется при `persistence-mode = prostore`
- `datasource` - источник данных, например `persistence`.

2.2.3.2.9 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`.

2.2.3.2.10 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.

2.2.3.2.11 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}**.

2.2.3.2.12 Секция paramstorage

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

Например:

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

2.2.3.2.13 Секция deltastorage

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

Например:

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

2.2.3.2.14 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.

2.2.3.2.15 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:
  blob-source: # настройки подключения к BLOB адаптеру
    host: 'localhost'
    port: 8080
    path:
  ftp-destination:
    host: localhost # хост ftp смева
    port: 21 # порт ftp смева
    # path: aaa/bbb/ccc # корневой каталог
    user: user # пользователь
    password: 123 # пароль
```

Параметры настроек

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

2.2.3.2.16 Секция rest

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

Например:

```
rest:
  enabled: false # вкл/выкл
  get: /le # путь get запроса
  post: /le # путь post запроса
  template: smev3-adapter/templates/smev.xml.peb # обрабатываемый шаблон
```

Параметры настроек

- **enabled** - включение настроек;
- **get** - путь GET запроса;
- **post** - путь POST запроса;
- **template** - путь к Pebble-шаблону.

2.2.3.2.17 Секция scheduler

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

Например:

```
scheduler:
  templates:
    - enabled: false
  interval: PT30s
  template: smev3-adapter/templates/pfr-delta.peb
```

Параметры настроек

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

2.2.3.2.18 Секция pool

В секции **pool** указывается размер прерываемого кода.

Например:

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

Параметры настроек

- **reader-executor** - корутин пул для обработки смев шаблонов;
- **schedule-executor** - корутин пул для обработки шедулера.

2.2.3.2.19 Секция logging

В секции **logging** настраивается логирование работы модуля.

Например:

```

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

```

Параметры настроек

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

2.2.3.2.20 Секция `http-server`

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

Например:

```

server:
  port: ${SERVER_PORT:8080}

```

Параметры настроек

- `port` - порт веб-сервера, например: `SERVER_PORT:8080`.

2.2.3.2.21 Секция `backup`

Секция `backup` предназначена для настроек бекапирования модуля.

Например:

```

backup:
  mode: ${BACKUP_MODE:rest} # kafka | rest
  rest:
    uri: ${BACKUP_URI:/backup}
  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}

```

Параметры настроек

- `mode` - режим бекапирования, например `BACKUP_MODE:rest`;
- `uri` - путь к файлу бекапирования в случае выбора REST-сервисов для режима бекапирования;
- `backupTopic` - топик для отправки сохраненных данных, например: `{BACKUP_TOPIC:adapter.backup}`;
- `statusTopic` - топик для отправки статусов бекапирования, например: `{STATUS_TOPIC:adapter.status}`.

2.2.3.2.22 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.

2.2.3.2.23 Секция dynamic-config

В секции **dynamic-config** хранятся параметры подключения динамических конфигураций.

Например:

```
dynamic-config:
  # Включение процедуры сканирования источника на предмет наличия конфигураций
  enabled: false
  # Интервал проверки источника
  refresh: PT3m
  # Источник подключаемых конфигураций
  table-receiver-scheduler: smeV3_adapter_receiver_scheduler
  # Источник pebble-шаблонов
  table-pebble: smeV3_adapter_pebble
```

Параметры настроек

- **enabled** - включение процедуры сканирования источника на предмет наличия конфигураций;
- **refresh** - интервал проверки источника;
- **table-receiver-scheduler** - источник подключаемых конфигураций, например **smeV3_adapter_receiver_scheduler**;
- **table-pebble** - источник pebble-шаблонов, например **smeV3_adapter_pebble**;

2.2.3.2.24 Секция component-info

В секции **component-info** хранятся настройки компонента 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:
    - keys
```

Параметры настроек

- **enabled** - статус подключения компонента, указывается булево значение;
- **datasource** - датасорс Prostore;

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

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

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

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

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

environment:
  # Папка для ошибочных файлов
  error-folder: ${ENVIRONMENT_ERROR_FOLDER:error}

prostore-rest-client:
  persistence-datamart: ${PERSISTENCE_DATAMART:persistence} # датамарт, в котором
  # будут располагаться таблицы с данными сервиса
  datasource: ${PERSISTENCE_DATASOURCE:} # по умолчанию пусто, тогда берется
  # единственный датасорс из настроек Простора
  table-flk-journal: ${PERSISTENCE_FLK_JOURNAL_TABLE:csv_uploader_flk_journal} #
  # таблица журнала ФЛК
  table-upload-journal: ${PERSISTENCE_UPLOAD_JOURNAL_TABLE:csv_uploader_upload_journal}
  # таблица журнала загрузок
  table-settings: ${PERSISTENCE_SETTINGS_TABLE:csv_uploader_settings} # таблица
  # хранения настроек сервиса
  storage-duration: 14d # продолжительность хранения данных в журналах
  cleanup-interval: 10m # периодичность очистки данных из таблиц журналов
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9195}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}

validation:
  enabled: ${VALIDATION_ENABLE:true}
  rest-uploader-url: ${REST_UPLOADER_URL:http://localhost:8081}
```

```

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

# 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:
  mode: ${BACKUP_MODE:rest} # kafka / rest
  rest:
    uri: ${BACKUP_URI:/backup}
  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}

```

2.2.4.2 Параметры конфигурации

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

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

- `http-server` - настройки порта подключения;
- `file-size` - ограничение на размер отправляемого файла (мегабайты);
- `logging.level` - настройка сохранения лог-файла;
- `environment` - определяет папку для ошибочных файлов;
- `prostore-rest-client` - блок параметров конфигурирования взаимодействия с **ProStore**;
- `validation` - адрес подключения модуля **REST-Uploader**;
- `upload` - требование токена для аутентификации на **REST-Uploader**;
- `csv-parser` - настройка парсинга CSV;
- `metrics` - настройка получения метрик;
- `backup` - настройка бэкапирования модуля;

2.2.4.2.1 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`.

2.2.4.2.2 Секция `file-size`

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

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

Параметры конфигурации

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

2.2.4.2.3 Секция `logging.level`

В секции `logging.level` настраиваются записи логирования.

Например:

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

2.2.4.2.4 Секция `environment`

В секции `environment` указывается папка для ошибочных файлов.

Например:

```
environment:
  error-folder: ${ENVIRONMENT_ERROR_FOLDER:error}
```

Параметры конфигурации

- `error-folder` - папка для ошибочных файлов, например `ENVIRONMENT_ERROR_FOLDER:error`.

2.2.4.2.5 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:}
  table-flk-journal: ${PERSISTENCE_FLK_JOURNAL_TABLE:csv_uploader_flk_journal}
  table-upload-journal: ${PERSISTENCE_UPLOAD_JOURNAL_TABLE:csv_uploader_upload_journal}
  table-settings: ${PERSISTENCE_SETTINGS_TABLE:csv_uploader_settings}
  storage-duration: 14d
  cleanup-interval: 10m
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9195}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}

```

Параметры настроек

- **persistence-datamart** - датамарт, в котором будут располагаться таблицы с данными сервиса, например **PERSISTENCE_DATAMART:persistence**;
- **datasource** - по умолчанию пусто, берется единственный датасорс из настроек Prostore;
- **table-flk-journal** - таблица журнала ФЛК, например **PERSISTENCE_FLK_JOURNAL_TABLE:csv_uploader_flk_journal**;
- **table-upload-journal** - таблица журнала загрузок, например **PERSISTENCE_UPLOAD_JOURNAL_TABLE:csv_uploader_upload_journal**;
- **table-settings** - таблица хранения настроек сервиса, например **PERSISTENCE_SETTINGS_TABLE:csv_uploader_settings**;
- **storage-duration** - продолжительность хранения данных в журналах, например **14d**, указывается в днях;
- **cleanup-interval** - периодичность очистки данных из таблиц журналов, например **10m**, указывается в минутах;
- **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**.

2.2.4.2.6 Секция validation

В секции **validation** реализован механизм настройки валидации ФЛК.

Например:

```

validation:
  enabled: ${VALIDATION_ENABLE:true}
  rest-uploader-url: ${REST_UPLOADER_URL:http://localhost:8081}

```

Параметры конфигурации

- **enabled** - подключение к сервису REST-Uploader для выполнения валидации, например **VALIDATION_ENABLE:true**;
- **rest-uploader-url** - URL к сервису REST-Uploader для выполнения валидации, например **{REST_UPLOADER_URL:http://localhost:8081}**.

2.2.4.2.7 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}
mandatoryFlk: ${UPLOAD_MANDATORY_FLK:false}
```

Параметры конфигурации

- **jwt-auth** - требование токена для аутентификации на REST-Uploader у пользователя при отправке данных на загрузку из пользовательского интерфейса, например **{JWT_AUTH:false}**;
- **jmandatoryFlk** - обязательность ФЛК, значение false позволяет отключить проверку ФЛК в веб интерфейсе.

2.2.4.2.8 Секция csv-parser

Внимание:

При загрузке файлов с форматно-логическим контролем, важно, чтобы настройки секции csv-parser должны быть синхронизированы с соответствующими настройками [csvparser Prostore](#). Так же настройки секции csv-parser должны быть одинаковыми в модулях CSV-Uploader (если используется его UI), REST-Uploader и DATA-Uploader.

Секция **csv-parser** - настройка парсинга CSV.

Например:

```
# 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}
```

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»].

2.2.4.2.9 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**.

2.2.4.2.10 Секция **backup**

Секция **backup** предназначена для настроек бекапирования модуля.

Например:

```
backup:
  mode: ${BACKUP_MODE:rest} # kafka | rest
  rest:
    uri: ${BACKUP_URI:/backup}
  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}
```

Параметры настроек

- **mode** - режим бекапирования, например **BACKUP_MODE:rest**;
- **uri** - путь к файлу бекапирования в случае выбора REST-сервисов для режима бекапирования;
- **backupTopic** - топик для отправки сохраненных данных, например:

- `{BACKUP_TOPIC:adapter.backup};`
- `statusTopic` - топик для отправки статусов бэкапирования, например:
`{STATUS_TOPIC:adapter.status}.`

2.2.5 Настройка DATA-Uploader – Модуль исполнения асинхронных заданий

2.2.5.1 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.

2.2.5.1.1 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&redis

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 экземпляров 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-duration: 30
  idle-timeout-duration: 30

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
  llw-batch: ${UPLOAD_LLW_BATCH:1000} # Число записей в одной команде upsert/delete в
  # случае llw загрузки.
  stream-batch: ${UPLOAD_STREAM_BATCH:10000} # Число записей в одной команде
  # upsert/delete в случае потоковой загрузки частями.
  creating-delta-on-upload-request: ${DELTA_CREATING_MODE:enable} #
  # enable/disable/period
  period: ${CREATING_DELTA_ON_UPLOAD_REQUEST_PERIOD:300}s # Используется при 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
  preferred_mode: ${UPLOAD_MODE:stream} # stream/llw // Режим загрузки в prostore

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/prostore
  # Директория хранения файлов для типа dir
  dir: ${DATA_STORAGE_DIR:/upload/data}
  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}

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

stream:
  timeout: ${DELTA_TIMEOUT:300}
  row-max-count: ${DELTA_MAX_ROWS:500000}
  avro-codec: zstd

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

# 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 не активным 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

metrics:
  port: ${METRICS_PORT:9837}

```

2.2.5.2 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;
- `stream` - настройка потоковой загрузки, в случае если поток разбивается на части;
- `response` - настройка времени хранения ответов по заданию в **Redis**;
- `component-info` - настройки модуля сбора информации о компонентах витрины;
- `metrics` - настройка получения метрик.

2.2.5.2.1 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`.

2.2.5.2.2 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`.

2.2.5.2.3 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 экземпляров 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 health-check, например **PERSISTENCE_HEALTH_CHECK_TABLE:data_uploader_health_check**;
- **table-validation-complete** - таблица с задачами загрузки данных, например **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 health-check в секундах, например **HEALTH_CHECK_REFRESH_PERIOD:120**;
- **health-check-wait-period-sec** - период ожидания health-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**.

2.2.5.2.4 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-duration: 30
    idle-timeout-duration: 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-duration** - таймаут на соединение;
- **idle-timeout-duration** - таймаут ожидания ответа от редиса.

2.2.5.2.5 Секция upload

Секция **upload** определяет настройки параллельной загрузки данных.

Например:

```
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
  llw-batch: ${UPLOAD_LLW_BATCH:1000} # Число записей в одной команде upsert/delete в
  случае llw загрузки.
  stream-batch: ${UPLOAD_STREAM_BATCH:10000} # Число записей в одной команде
  upsert/delete в случае потоковой загрузки частями.
  creating-delta-on-upload-request: ${DELTA_CREATING_MODE:enable} #
  enable/disable/period
  period: ${CREATING_DELTA_ON_UPLOAD_REQUEST_PERIOD:300}s # Используется при 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
  preferred_mode: ${UPLOAD_MODE:stream} # stream/llw // Режим загрузки в prostore
```

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**;
- **llw-batch** - число записей в одной команде upsert/delete в случае llw загрузки, например **UPLOAD_LLW_BATCH:1000**;
- **creating-delta-on-upload-request** - создание дельты при запросе загрузки, например **DELTA_CREATING_MODE:enable** (возможные значения: **enable**/ **disable** / **period**);
- **period** - период создания дельты при запросе загрузки (в секундах), например **CREATING_DELTA_ON_UPLOAD_REQUEST_PERIOD:300**, используется при **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).
- **preffered_mode** - режим загрузки, например **UPLOAD_MODE:stream**;

2.2.5.2.6 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/prostore
  # Директория хранения файлов для типа dir
  dir: ${DATA_STORAGE_DIR:/upload/data}
  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:/upload/data**;
- **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`.

2.2.5.2.7 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`.

2.2.5.2.8 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 `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`;
- `chroot` - Zookeeper DS chroot path, например `ZOOKEEPER_DS_CHROOT:/adapter`.

2.2.5.2.9 Секция `csv-parser`

Внимание:

При загрузке файлов с форматно-логическим контролем, важно, чтобы настройки секции `csv-parser` должны быть синхронизированы с соответствующими настройками [csvparser Prostore](#). Так же настройки секции `csv-parser` должны быть 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}  
  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 многострочных 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»].

2.2.5.2.10 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.

2.2.5.2.11 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.**

2.2.5.2.12 Секция stream

Секция **stream** содержит настройки потоковой загрузки, в случае если поток разбивается на части.

Например:

```
stream:
  timeout: ${DELTA_TIMEOUT:300}
  row-max-count: ${DELTA_MAX_ROWS:500000}
  avro-codec: zstd
```

Параметры настроек

- **timeout** - максимальный интервал времени между первой считанной записью цикла загрузки и отправкой заданий на загрузку потока, например **DELTA_TIMEOUT:300**;
- **row-max-count** - максимальное количество обработанных записей для старта отправки заданий на загрузку потока, например **DELTA_MAX_ROWS:500000**;
- **avro-codec** - AVRO кодек, например **zstd**.

2.2.5.2.13 Секция response

Секция **response** определяет настройку 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>).

2.2.5.2.14 Секция component-info

В секции **component-info** хранятся настройки компонента сбора информации о компонентах витрины.

Например:

```
# 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 не активным 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
```

Параметры настроек

- **enabled** - статус подключения компонента, указывается булево значение;
- **datasource** - датаосrc Prostore;
- **datamart** - схема Prostore;
- **table-name** - имя таблицы для записи информации о компоненте;
- **create-table-period** - период попыток создания схемы, при неуспехе, указывается

в секундах;

- **publish-period** - период публикации health-check, указывается в секундах;
- **timeout-active** - период после которого компонент считается неактивным при отсутствии health-check, указывается в секундах;
- **secrets** - список элементов конфига маскируемых при отправке, если указан узловой элемент, то маскируются все вложенные в него элементы.

2.2.5.2.15 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**.

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

2.2.6.1 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.

2.2.6.1.1 Пример файла application.yml

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

```
http-server:
  port: ${SERVER_PORT: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/prostore
  # Директория хранения файлов для tuna dir
  dir: ${DATA_STORAGE_DIR:/upload/data}
  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: ${CONDITIONS_REST_TIMEOUT:60s}
  # период жизни группы
  save_group_time: 1d
  validation:
    # таймаут выполнения асинхронной проверки
    validation-timeout: 1h
    # таймаут получения сообщений из redis (используется при persistence-mode:
    zookeeper&redis)
    # значение должно быть меньше чем redis.net-client-options.tcp-user-timeout
    poll-timeout: 15s
    # таймаут периодического опроса prostore для получения данных из очереди
    (используется при persistence-mode: prostore)
    poll-interval: 5s
    # 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

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&redis

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 экземпляров 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-mpw
    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-duration: 30s
    idle-timeout-duration: 30s

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}
  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

```

2.2.6.2 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.

2.2.6.2.1 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**.

2.2.6.2.2 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**

2.2.6.2.3 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**.

2.2.6.2.4 Секция **environment**

Секция **environment** предназначена для настройки среды окружения.

Например:

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

Параметры настроек

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

2.2.6.2.5 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/prostore
  # Директория хранения файлов для типа dir
  dir: ${DATA_STORAGE_DIR:/upload/data}
  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:/upload/data;**
- **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.**

2.2.6.2.6 Секция conditions

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

Например:

```

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: ${CONDITIONS_REST_TIMEOUT:60s}
  # период жизни группы
  save_group_time: 1d
  validation:
    # таймаут выполнения асинхронной проверки
    validation-timeout: 1h

```

```

# таймаут получения сообщений из redis (используется при persistence-mode:
zookeeper&redis)
# значение должно быть меньше чем redis.net-client-options.tcp-user-timeout
poll-timeout: 15s
# таймаут периодического опроса prostore для получения данных из очереди
(используется при persistence-mode: prostore)
poll-interval: 5s
# 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

```

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 (используется при persistence-mode: zookeeper&redis) **15s**, значение должно быть меньше чем redis.net-client-options.tcp-user-timeout;
- **poll-interval** - таймаут периодического опроса prostore для получения данных из очереди (используется при persistence-mode: prostore);
- **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**;
- **valid-charsets** - допустимые кодировки для механизма автоопределения, например **UTF-8, US-ASCII, TIS-620**;
- **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. 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. 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. Записывается 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. 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.

2.2.6.2.7 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**.

2.2.6.2.8 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**.

2.2.6.2.9 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 экземпляров 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 health-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**.

2.2.6.2.10 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**.

2.2.6.2.11 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`.

2.2.6.2.12 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-duration: 30s
    idle-timeout-duration: 30s
```

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-duration** - таймаут на соединение;
- **idle-timeout-duration** - таймаут ожидания ответа от редиса.

2.2.6.2.13 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`.

2.2.6.2.14 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`.

2.2.6.2.15 Секция `csv-parser`

Внимание:

При загрузке файлов с форматно-логическим контролем, важно, чтобы настройки секции `csv-parser` должны быть синхронизированы с соответствующими настройками [csvparser Prostore](#). Также настройки секции `csv-parser` должны быть одинаковыми 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 многостроковых 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»]`.

2.2.6.2.16 Секция backup

Секция `backup` предназначена для настроек бекапирования модуля.

Например:

```
backup:
  mode: ${BACKUP_MODE:rest} # kafka / rest
  rest:
    uri: ${BACKUP_URI:/backup}
  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}
```

Параметры настроек

- `mode` - режим бекапирования, например `BACKUP_MODE:rest`;
- `uri` - путь к файлу бекапирования в случае выбора REST-сервисов для режима бекапирования;
- `backupTopic` - топик для отправки сохраненных данных, например: `{BACKUP_TOPIC:adapter.backup}`;
- `statusTopic` - топик для отправки статусов бекапирования, например: `{STATUS_TOPIC:adapter.status}`.

2.2.6.2.17 Секция component-info

В секции `component-info` указываются настройки модуля сбора информации о

компонентах витрины.

Например:

```
# 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
```

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.

2.2.6.3 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 2.12 Список реализованных проверок

Наименование проверки	Код ошибки	Кириллическое описание
Проверка уникальности	duplicate	Дубликат файла/группы
Проверка парсинга файла	parsingErr	Ошибка парсинга: <i>текст ошибки</i>
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 <i>требуемый тип</i>
Проверка уникальности полей	nonUniq	Значение не отвечает требованиям уникальности
Проверка регулярных выражений	nonMatchRegex	Значение не соответствует регулярному выражению <i>регулярное выражение</i>
Проверка соответствия условию	nonMatchConstant	Значение не соответствует условию <i>условие</i>
Таймаут валидации	validationTimeout	Истек таймаут валидации файла

2.2.6.3.1 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.

2.2.6.3.2 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.

2.2.6.3.2.1 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.

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
```

2.2.6.3.2.2 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>)

2.2.6.3.2.3 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.

2.2.6.4 Статусная модель

Статус запроса к модулю REST-Uploader 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 2.13 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 буферизирован	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 – 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 <code>"/v2/requests/{request_id}/report/"</code> Далее проанализировать и устранить выявленные недочеты в загружаемых данных или скорректировать проверки ФЛК.

2.2.6.5 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

DELETE	v2/datamarts/{datamart_name}/tables/{table_name}/delete	Логическое 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
POST	/v2/datamarts/{datamart_name}/tables/{table_name}/truncate	Физическое удаления и/или очистка исторических данных

```

openapi: 3.0.1
info:
  title: Rest-uploader
  description: This is a rest-uploader service for datamart filling
  version: 2.0.0
paths:
  /v2/datamarts/{datamart_name}/tables/{table_name}/upload:
    post:
      summary: Add a new data to the datamart
      operationId: v2-datamarts-datamart_name-tables-table_name-upload
      description: Загрузка данных из внешних источников. К телу прикладывается файл
      tags:
        - data
      parameters:
        - $ref: '#/components/parameters/datamartName'
        - $ref: '#/components/parameters/tableName'
        - $ref: '#/components/parameters/groupId'
        - $ref: '#/components/parameters/groupFileNum'
        - $ref: '#/components/parameters/groupFileCount'
      requestBody:
        $ref: '#/components/requestBodies/uploadData'
      responses:
        '200':
          description: OK
          headers:
            requestId:

```

```

        schema:
          $ref: '#/components/schemas/requestId'
      content:
        text/plain:
          schema:
            $ref: '#/components/schemas/requestId'
    '400':
      $ref: '#/components/responses/badRequest'
    '401':
      $ref: '#/components/responses/unauthorized'
    '500':
      $ref: '#/components/responses/internalServerError'
  security:
    - bearerAuth: []
/v2/requests/{request_id}/status:
  get:
    summary: Return request status
    description: Возвращение статуса запроса
    operationId: get-v2-request-request_id-status
    tags:
      - data
    parameters:
      - $ref: '#/components/parameters/requestId'
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              type: object
              required:
                - code
                - description
                - errorMessage
              properties:
                code:
                  type: integer
                  description: Код статуса
                description:
                  type: string
                  description: Описание статуса
                errorMessage:
                  type: string
                  description: Описание ошибки
      '401':
        $ref: '#/components/responses/unauthorized'
      '500':
        $ref: '#/components/responses/internalServerError'
  security:
    - bearerAuth: []
/v2/requests/{request_id}/reportFLK:
  get:
    summary: Return report FLK check
    description: Возвращает отчет по формату логическом контроле загружаемых данных
    operationId: get-v2-request-request_id-reportFLK
    tags:
      - report
    parameters:
      - $ref: '#/components/parameters/requestId'
    responses:
      '200':
        description: OK

```

```

        content:
          text/csv:
            schema:
              type: array
              items:
                type: array
                items:
                  type: string
      '401':
        $ref: '#/components/responses/unauthorized'
      '500':
        $ref: '#/components/responses/internalServerError'
    security:
      - bearerAuth: []
/v2/group/{groupId}/report:
  get:
    summary: Return report group
    description: Возвращает отчет по комплектности группы загружаемых файлов
    operationId: get-v2-group-groupId-report
    tags:
      - report
    parameters:
      - name: groupId
        in: path
        description: Identifier of group
        required: true
        schema:
          type: string
    responses:
      '200':
        description: OK
        content:
          text/csv:
            schema:
              type: array
              items:
                type: array
                items:
                  type: string
      '401':
        $ref: '#/components/responses/unauthorized'
      '500':
        $ref: '#/components/responses/internalServerError'
    security:
      - bearerAuth: []
/v2/datamarts/{datamart_name}/beginDelta:
  post:
    summary: begin delta
    operationId: post-v2-datamart_name-begin_delta
    tags:
      - delta
    parameters:
      - $ref: '#/components/parameters/datamartName'
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              type: object
              properties:
                deltaNum:

```

```

        type: integer
    '400':
        description: Bad Request
    '401':
        $ref: '#/components/responses/unauthorized'
    '500':
        $ref: '#/components/responses/internalServerError'
description: открытие дельты
security:
    - bearerAuth: []
/v2/datamarts/{datamart_name}/commitDelta:
post:
    summary: commit delta
    operationId: post-v2-datamart_name-commit_delta
    tags:
        - delta
    parameters:
        - $ref: '#/components/parameters/datamartName'
    responses:
        '200':
            description: OK
            content:
                application/json:
                    schema:
                        type: object
                        properties:
                            deltaNum:
                                type: integer
                            deltaDate:
                                type: string
                            cnFrom:
                                type: integer
        '400':
            description: Bad Request
        '401':
            $ref: '#/components/responses/unauthorized'
        '500':
            $ref: '#/components/responses/internalServerError'
description: применение дельты
security:
    - bearerAuth: []
/v2/datamarts/{datamart_name}/rollbackDelta:
post:
    summary: rollback delta
    operationId: post-v2-datamart_name-rollback_delta
    tags:
        - delta
    parameters:
        - $ref: '#/components/parameters/datamartName'
    responses:
        '200':
            description: OK
            content:
                application/json:
                    schema:
                        type: object
                        properties:
                            deltaNum:
                                type: integer
        '400':
            description: Bad Request
        '401':

```

```

    $ref: '#/components/responses/unauthorized'
  '500':
    $ref: '#/components/responses/internalServerError'
  description: отмена дельты
  security:
    - bearerAuth: []
/v2/datamarts/{datamart_name}/getDeltaOk:
  get:
    summary: get delta ok
    operationId: get-v2-datamart_name-get_delta_ok
    tags:
      - delta
    parameters:
      - $ref: '#/components/parameters/datamartName'
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              type: object
              properties:
                deltaNum:
                  type: integer
                deltaDate:
                  type: string
                cnFrom:
                  type: integer
      '400':
        description: Bad Request
      '401':
        $ref: '#/components/responses/unauthorized'
      '500':
        $ref: '#/components/responses/internalServerError'
  description: получение примененной дельты
  security:
    - bearerAuth: []
/v2/datamarts/{datamart_name}/getDeltaHot:
  get:
    summary: get delta hot
    operationId: get-v2-datamart_name-get_delta_hot
    tags:
      - delta
    parameters:
      - $ref: '#/components/parameters/datamartName'
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              type: object
              properties:
                deltaNum:
                  type: integer
                cnFrom:
                  type: integer
                cnMax:
                  type: integer
                isRollingBack:
                  type: boolean
                writeOpFinished:

```

```

        type: string
    '400':
        description: Bad Request
    '401':
        $ref: '#/components/responses/unauthorized'
    '500':
        $ref: '#/components/responses/internalServerError'
description: получение открытой дельты
security:
    - bearerAuth: []
/v2/LengthFlkQueue:
get:
    summary: get length flk queue
    operationId: get-v2-get_length_flk_queue
    tags:
        - data
    responses:
        '200':
            description: OK
            content:
                text/plain:
                    schema:
                        type: integer
        '401':
            $ref: '#/components/responses/unauthorized'
        '500':
            $ref: '#/components/responses/internalServerError'
description: |-
    получение длины очереди ФЛК
security:
    - bearerAuth: []
/v2/datamarts/{datamart_name}/LengthUploadQueue:
get:
    summary: get length upload queue
    operationId: get-v2-get_length_upload_queue
    tags:
        - data
    parameters:
        - $ref: '#/components/parameters/datamartName'
    responses:
        '200':
            description: OK
            content:
                text/plain:
                    schema:
                        type: integer
        '400':
            description: Bad Request
        '401':
            $ref: '#/components/responses/unauthorized'
        '500':
            $ref: '#/components/responses/internalServerError'
description: получение длины очереди загрузки данных в датамарт
security:
    - bearerAuth: []
/v2/datamarts/{datamart_name}/tables/{table_name}/delete:
post:
    summary: Delete data by primary key array
    operationId: post-v2-datamart_name-tables-table_name-delete
    tags:
        - data
    parameters:

```

```

- $ref: '#/components/parameters/datamartName'
- $ref: '#/components/parameters/tableName'
responses:
  '200':
    description: OK
    headers:
      requestId:
        schema:
          $ref: '#/components/schemas/requestId'
    content:
      text/plain:
        schema:
          $ref: '#/components/schemas/requestId'
  '400':
    $ref: '#/components/responses/badRequest'
  '401':
    $ref: '#/components/responses/unauthorized'
  '500':
    $ref: '#/components/responses/internalServerError'
description: Удаление данных по массиву первичных ключей
security:
  - bearerAuth: []
requestBody:
  $ref: '#/components/requestBodies/deleteData'
/v2/datamarts/{datamart_name}/tables/{table_name}/truncate:
  post:
    summary: Delete historical data by primary key array
    operationId: post-v2-datamart_name-tables-table_name-truncate
    tags:
      - data
    parameters:
      - $ref: '#/components/parameters/datamartName'
      - $ref: '#/components/parameters/tableName'
      - $ref: '#/components/parameters/forSystemTime'
    responses:
      '200':
        description: OK
        headers:
          requestId:
            schema:
              $ref: '#/components/schemas/requestId'
        content:
          text/plain:
            schema:
              $ref: '#/components/schemas/requestId'
      '400':
        $ref: '#/components/responses/badRequest'
      '401':
        $ref: '#/components/responses/unauthorized'
      '500':
        $ref: '#/components/responses/internalServerError'
    description: Удаление исторических данных по массиву первичных ключей
    security:
      - bearerAuth: [ ]
    requestBody:
      $ref: '#/components/requestBodies/deleteData'
/v2/conditions/{datamart}/{table}:
  parameters:
    - $ref: '#/components/parameters/datamartName'
    - $ref: '#/components/parameters/tableName'
  post:
    summary: Create verification conditions

```

```

description: Формирование правил проверки для датамарта/таблицы
operationId: post-v2-conditions
tags:
  - conditions
requestBody:
  content:
    application/x-yaml: {}
responses:
  '200':
    description: OK
  '401':
    $ref: '#/components/responses/unauthorized'
  '500':
    $ref: '#/components/responses/internalServerError'
security:
  - bearerAuth: []
put:
  summary: Update verification conditions
  description: Обновление правил проверки для датамарта/таблицы
  operationId: put-v2-conditions
  tags:
    - conditions
  requestBody:
    content:
      application/x-yaml: {}
  responses:
    '200':
      description: OK
    '401':
      $ref: '#/components/responses/unauthorized'
    '500':
      $ref: '#/components/responses/internalServerError'
  security:
    - bearerAuth: []
get:
  summary: Return verification conditions
  description: Возвращение правил проверки для датамарта/таблицы
  operationId: get-v2-conditions
  tags:
    - conditions
  responses:
    '200':
      description: OK
      content:
        text/yaml: {}
    '401':
      $ref: '#/components/responses/unauthorized'
    '500':
      $ref: '#/components/responses/internalServerError'
  security:
    - bearerAuth: []
delete:
  summary: Delete verification conditions
  description: Удаление правил проверки для датамарта/таблицы
  operationId: delete-v2-conditions
  tags:
    - conditions
  responses:
    '200':
      description: OK
    '401':
      $ref: '#/components/responses/unauthorized'

```

```

    '500':
      $ref: '#/components/responses/internalServerError'
  security:
    - bearerAuth: []
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
  schemas:
    requestId:
      type: string
      description: Идентификатор запроса
      example: e5091d55-a0c8-4cc0-ba70-b89d03dde1a6
  parameters:
    datamartName:
      name: datamart_name
      in: path
      description: 'Мнемоника витрины данных'
      required: true
      schema:
        type: string
    tableName:
      name: table_name
      in: path
      description: 'Наименование таблицы'
      required: true
      schema:
        type: string
    groupId:
      name: group_id
      in: header
      description: 'Идентификатор группы'
      required: false
      schema:
        type: string
    groupFileNum:
      name: group_file_num
      in: header
      description: 'Номер файла в группе'
      required: false
      schema:
        type: integer
    groupFileCount:
      name: group_file_count
      in: header
      description: 'Число файлов в группе'
      required: false
      schema:
        type: integer
    requestId:
      name: request_id
      in: path
      description: Идентификатор запроса
      required: true
      schema:
        $ref: '#/components/schemas/requestId'
    forSystemTime:
      name: for_system_time
      in: header
      description: Метка времени для удаления истории операций
      required: false
      schema:
        type: string
  responses:
    badRequest:
      description: Bad request
      content:
        text/plain:

```

```

    schema:
      type: string
      description: Сообщение об ошибке
      example: Отсутствует целевая таблица passenger в модели данных univer
  internalServerError:
    description: Internal server error
    content:
      text/plain:
        schema:
          type: string
          description: Сообщение об ошибке
          example: Internal server error

  unauthorized:
    description: Unauthorized
    content:
      text/plain:
        schema:
          type: string
          description: Сообщение об ошибке
          example: Unauthorized
  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
  - name: delta
    description: Управление дельтами

```

2.2.7 Настройка BLOB-адаптера

2.2.7.1 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.

2.2.7.1.1 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

```

```

blob:
  default-request-timeout: 60s
  storage:
    client: awssdk # 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}

prostore-rest-client:
  host: ${PS_HOST:localhost}
  port: ${PS_PORT:9095}
  http:
    max-pool-size: ${PS_MAX_POOL_SIZE:8}

# 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 не активным 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:
    - blob.storage.auth.user
    - blob.storage.auth.password
    - blob.storage.auth.token
    - blob.storage.auth.authorization-server.client-secret
    - blob.storage.auth.access-token
    - blob.storage.auth.secret-token

```

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

2.2.7.2 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;
- `blob` - настраивается подключение к Хранилищу BLOB-объектов;
- `logging` - настраивается логирование работы модуля;
- `prostore-rest-client` - блок параметров конфигурирования взаимодействия с **ProStore**;
- `component-info` - настройки модуля сбора информации о компонентах витрины;
- `metrics` - настраивается получение метрик.

2.2.7.2.1 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.

2.2.7.2.2 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`.

2.2.7.2.3 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.

2.2.7.2.4 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:
  default-request-timeout: 60s
  storage:
    client: awssdk # 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

- **default-request-timeout** - значение в секундах таймаута запроса;
- **client** - клиент сервера хранилища **BLOB-объектов**, например **BLOB_STORAGE_CLIENT:awssdk**;
- **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

- **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:.`

cURL (windows)

cURL (linux)

Пример сURL-запроса к Хранилищу BLOB-объектов

cURL (windows)

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&background=#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
    background: "#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=fdsf&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
```

2.2.7.2.5 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-response** - журналировать ответы, например **BLOB_RESPONSE_LOG_ENABLED:false**.

2.2.7.2.6 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:9095}
  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**;

2.2.7.2.7 Секция component-info

В секции **component-info** хранятся настройки компонента сбора информации о компонентах витрины.

Например:

```
# 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:
    - blob.storage.auth.user
    - blob.storage.auth.password
    - blob.storage.auth.token
    - blob.storage.auth.authorization-server.client-secret
    - blob.storage.auth.access-token
    - blob.storage.auth.secret-token

```

Параметры настроек

- **enabled** - статус подключения компонента, указывается булево значение;
- **datasource** - датаасорс Prostore;
- **datamart** - схема Prostore;
- **table-name** - имя таблицы для записи информации о компоненте;
- **create-table-period** - период попыток создания схемы, при неуспехе, указывается в секундах;
- **publish-period** - период публикации health-check, указывается в секундах;
- **timeout-active** - период после которого компонент считается неактивным при отсутствии health-check, указывается в секундах;
- **secrets** - список элементов конфига маскируемых при отправке, если указан узловой элемент, то маскируются все вложенные в него элементы.

2.2.7.2.8 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**.

2.2.8 Настройка Сервиса формирования документов

2.2.8.1 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 rebble-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.

2.2.8.1.1 Пример файла application.yml

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

```

http-server:
  port: ${HTTP_PORT: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:20}

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:
  # Таймаут по умолчанию для генерации отчётов, применяется если не задан
  конкретный таймаут на отчёт
  default-request-timeout: 60s
  # Список отчетов
  reports:
    # имя документа
    - report: document_1

```

```

# Таймаут для генерации отчёта
request-timeout: 30s
# 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

# 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:
    - keys

```

2.2.8.2 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;
- `component-info` - настройки модуля сбора информации о компонентах витрины.

2.2.8.2.1 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`.

2.2.8.2.2 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`

2.2.8.2.3 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:20}
```

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:20`.

2.2.8.2.4 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**

2.2.8.2.5 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**.

2.2.8.2.6 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**.

2.2.8.2.7 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:
    maxLength: ${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**;
- **maxLength** - 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**.

2.2.8.2.8 Секция printable-forms

В секции **printable-forms** настраивается сервис формирования документов.

Например:

```

printable-forms:
  # Таймаут по умолчанию для генерации отчётов, применяется если не задан
  конкретный таймаут на отчёт
  default-request-timeout: 60s
  # Список отчетов
  reports:
    # имя документа
    - report: document_1
    # Таймаут для генерации отчёта
    request-timeout: 30s
    # 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 rebble 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 rebble шаблона, который будет формировать подписываемый pdf
документ, задается в .pdf.template
# (для генерации "pdf без ЭП" и "pdf с ЭП" используется единый rebble-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 rebble-шаблонов должны быть либо: относительно директории запуска, либо абсолютные пути.

2.2.8.2.9 Секция component-info

В секции **component-info** хранятся настройки компонента сбора информации о компонентах витрины.

Например:

```

# 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:
    - keys

```

Параметры настроек

- **enabled** - статус подключения компонента, указывается булево значение;
- **datasource** - датаcorp Prostore;
- **datamart** - схема Prostore;
- **table-name** - имя таблицы для записи информации о компоненте;
- **create-table-period** - период попыток создания схемы, при неуспехе, указывается в секундах;

- **publish-period** - период публикации health-check, указывается в секундах;
- **timeout-active** - период после которого компонент считается неактивным при отсутствии health-check, указывается в секундах;
- **secrets** - список элементов конфига маскируемых при отправке, если указан узловой элемент, то маскируются все вложенные в него элементы.

2.2.8.3 Примеры rebble-шаблонов для Сервиса Формирования документов

2.2.8.3.1 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 rebble-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**.

2.2.8.3.2 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 }}

```

2.2.8.3.3 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>

```

2.2.8.3.4 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>

```

2.2.9 Настройка Counter-provider - Сервиса генерации уникального номера

2.2.9.1 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.

2.2.9.2 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}

backup:
  mode: ${BACKUP_MODE:rest} # kafka / rest
  rest:
    uri: ${BACKUP_URI:/backup}
  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}

# 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:
    - keys

metrics:
  port: ${METRICS_PORT:9837}

```

2.2.9.3 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;
- `component-info` - настройки модуля сбора информации о компонентах витрины;
- `metrics` - настройка порта для получения метрик.

2.2.9.3.1 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`.

2.2.9.3.2 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`.

2.2.9.3.3 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`.

2.2.9.3.4 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`.

2.2.9.3.5 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`.

2.2.9.3.6 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.

2.2.9.3.7 Секция migration

В секции **migration** настраиваются миграции **Zookeeper** для задачи бекапирования. Например:

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

Параметры настроек

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

2.2.9.3.8 Секция backup

В секции **backup** настраивается бекапирования модуля. Например:

```
backup:
  mode: ${BACKUP_MODE:rest} # kafka | rest
  rest:
    uri: ${BACKUP_URI:/backup}
  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 бекапирования;
- **backupTopic** - топик для отправки сохраненных данных, например: **{BACKUP_TOPIC:adapter.backup}**;
- **statusTopic** - топик для отправки статусов бекапирования, например: **{STATUS_TOPIC:adapter.status}**.

2.2.9.3.9 Секция component-info

В секции **component-info** хранятся настройки компонента сбора информации о компонентах витрины.

Например:

```
# 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:
    - keys
```

Параметры настроек

- **enabled** - статус подключения компонента, указывается булево значение;
- **datasource** - datasource Prostore;
- **datamart** - схема Prostore;
- **table-name** - имя таблицы для записи информации о компоненте;
- **create-table-period** - период попыток создания схемы, при неуспехе, указывается в секундах;
- **publish-period** - период публикации health-check, указывается в секундах;
- **timeout-active** - период после которого компонент считается неактивным при отсутствии health-check, указывается в секундах;
- **secrets** - список элементов конфига маскируемых при отправке, если указан узловой элемент, то маскируются все вложенные в него элементы.

2.2.9.3.10 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}**.

2.2.10 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/>.

2.2.11 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-
      parentsSpanid=%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
```

2.2.12 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.

2.2.12.1 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.

2.2.12.1.1 Настройка логирования модулей

Необходимо настроить формирование логов в формате JSON на стороне модулей:

- **BLOB-адаптер;**
- **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`.

2.2.12.1.2 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
```

2.2.12.1.3 Установка и настройка HaProxy

Установка производится по официальной документации: <http://docs.haproxy.org/>

Для настройки HaProxy в секции **backend** перечислите список установленных экземпляров **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
```

2.2.12.1.4 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.

2.2.12.1.5 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 }}"
```

2.2.12.1.6 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.

2.2.13 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](#).

2.2.13.1 Настройка взаимодействия Компонента с Агентом СМЭВ4

После установки Компонента 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 Компонента. Для этого используйте адрес сервера Kafka из конфигурационного файла СМЭВ4-адаптера (`application.yml`), параметр `kafkaUrl`.
2. Настройте названия топиков (см. [Таблица 2.14](#)) для обмена сообщениями в конфигурационном файле СМЭВ4-адаптера (`application.yml`).

Таблица 2.14 Название топиков для обмена сообщениями между СМЭВ4-адаптером и

№	Назначение	Настройка	Значение по умолчанию
1	Получение запросов	<code>client.kafka.query.consumer.rqTopicName</code>	<code>query.rq</code>
2	Ответы на запросы	<code>client.kafka.query.producer.rsTopicName</code>	<code>query.rs</code>
3	Ошибки запросов	<code>client.kafka.query.producer.errTopicName</code>	<code>query.err</code>
4	Результат запроса оценки	<code>client.kafka.query.estimateTopicName</code>	<code>query.query.estimate.rs</code>

Формат обмена электронными сообщениями с СМЭВ4-адаптером описан в разделе [Спецификация Модуля исполнения запросов](#).

2.3 Настройка сервиса мониторинга

Для мониторинга состояния работы Компонента «Витрина 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.

4. Откройте Grafana — Configuration — Data sources, нажмите **Add data source** и выберите **Prometheus**.

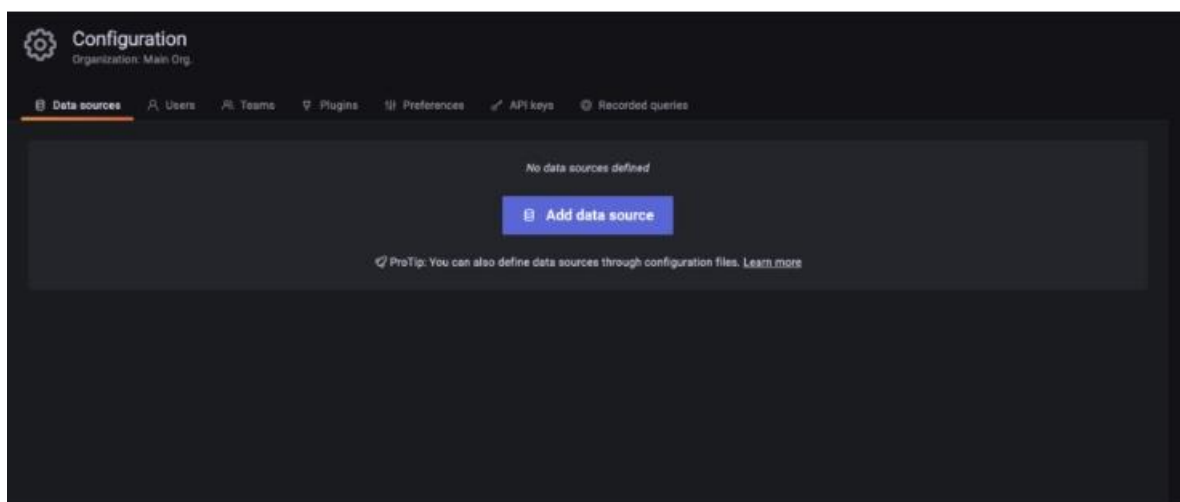


Рисунок - 2.1 Выбор Prometheus в качестве источника получения данных

5. В поле URL введите адрес и порт, по которому доступен Prometheus.

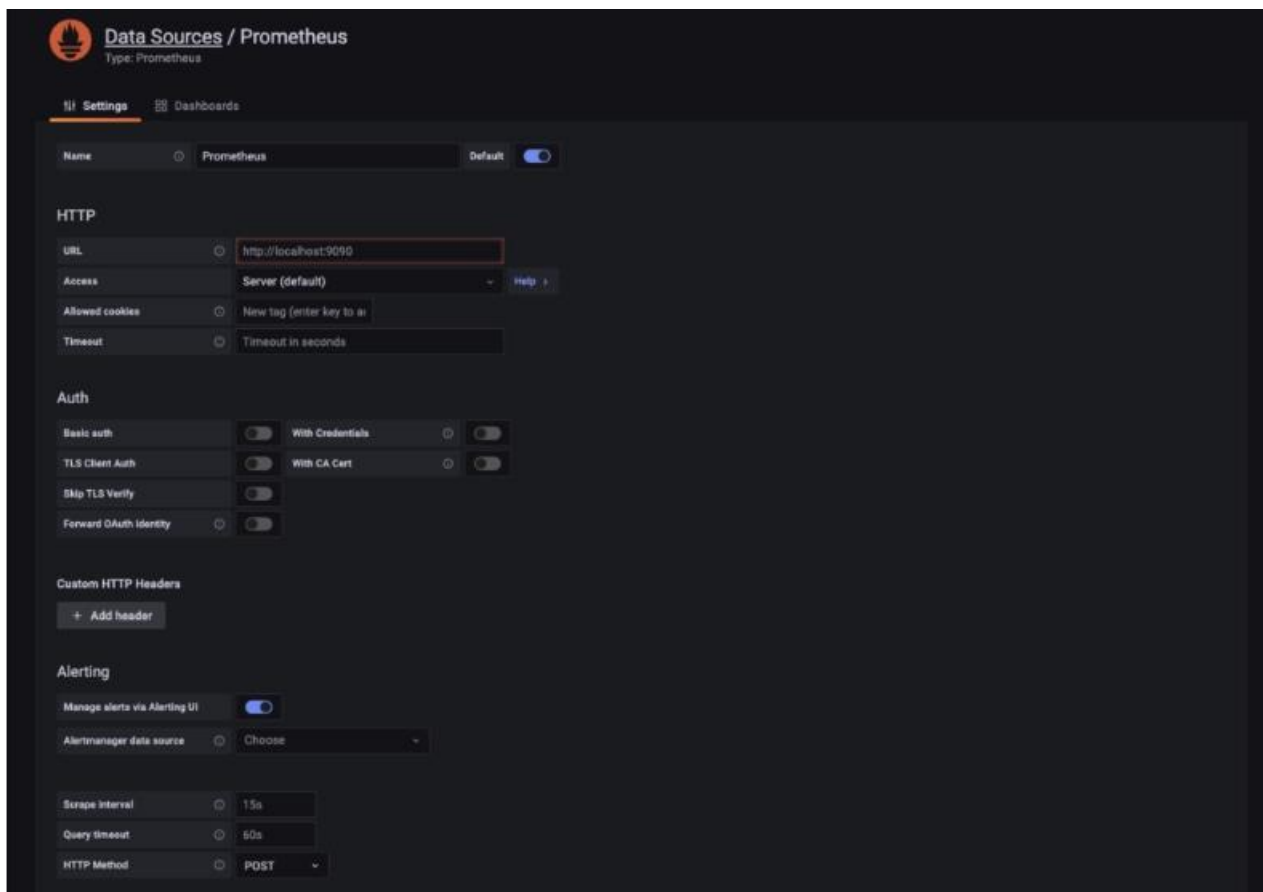


Рисунок - 2.2 Ввод URL Prometheus

6. Внизу страницы нажмите кнопку **Save & test**

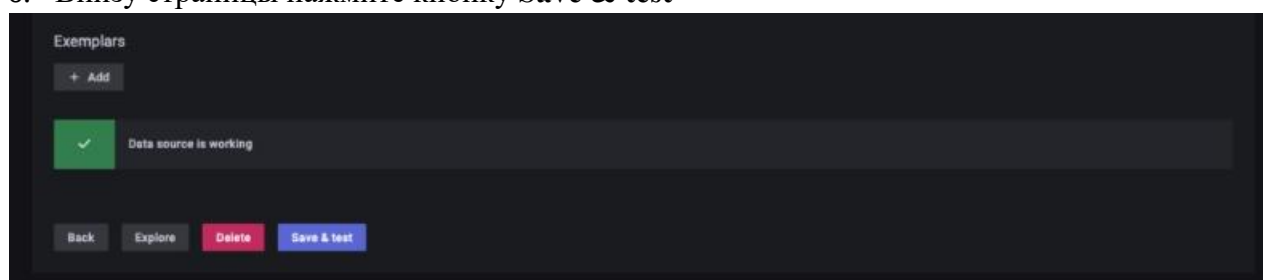


Рисунок - 2.3 Применение настроек

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».

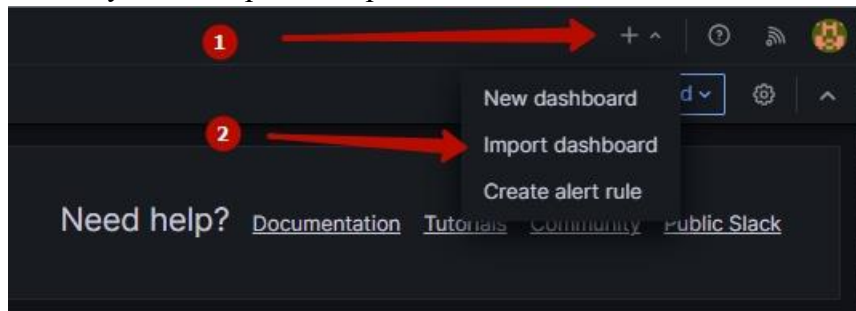


Рисунок - 2.4 Меню импорта Панели

2. В открывшемся окне нажмите на плашку «Upload dashboard JSON file» и загрузите файл нужной панели.

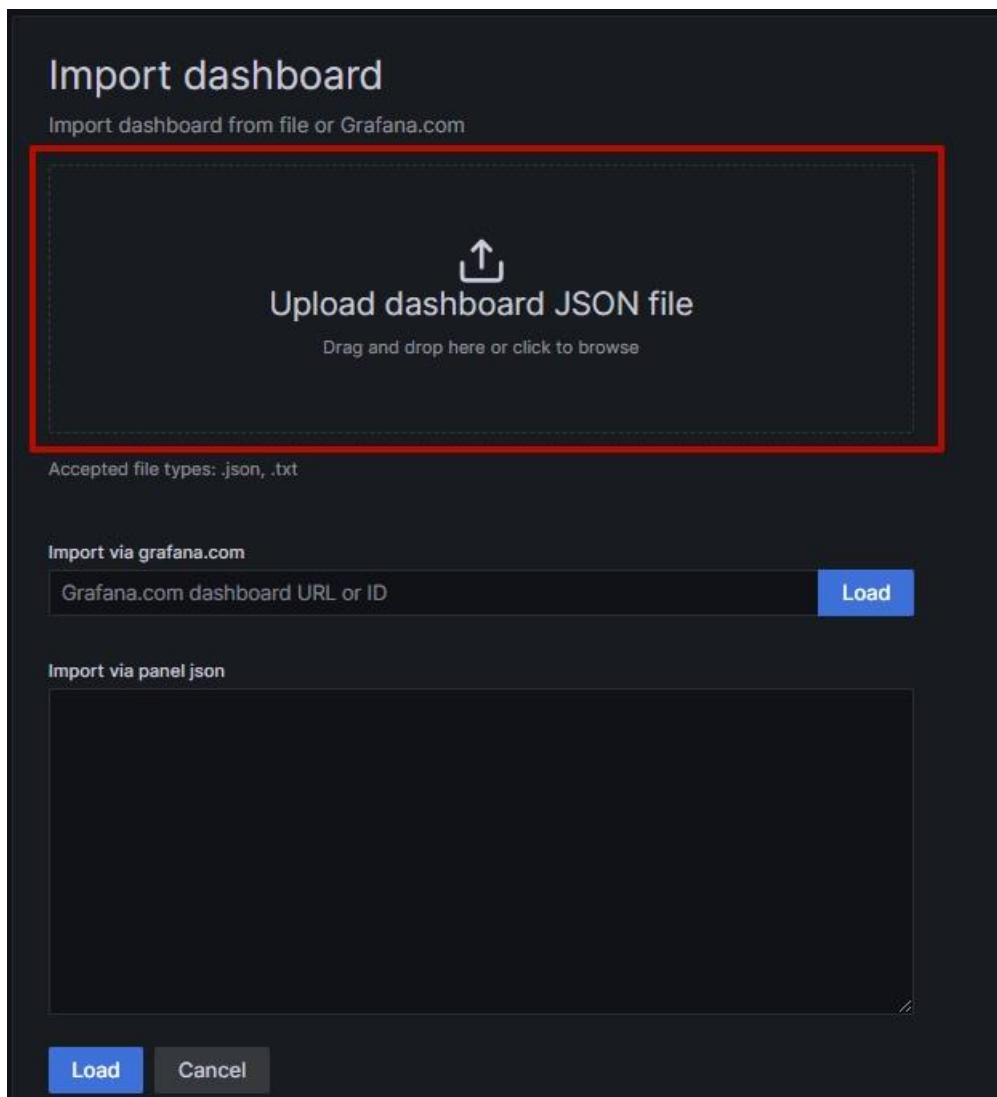


Рисунок - 2.5 Загрузка файла панели

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']
```

2.3.2.2 Health check

В данной концепции приведены health check по СМЭВ3, ПОДД, REST, BLOB и печатным формам. Health check по ETL реализуется штатными способами Spark, Airflow, Hadoop, Tarantool, так же как и health check по Kafka и Zookeeper.

Каждый сервис, из перечисленных выше, дорабатывается таким образом, чтобы возвращать информацию **liveness** и **readiness** в виде метрик Prometheus. Дополнительно, по каждому сервису реализуется соответствующие REST запросы: `../api/v1/liveness` и `../api/v1/readiness`, показывающим liveness и readiness проверки соответственно. Обращение к каждому конкретному сервису из представленных ниже по REST, обусловлено передачей хоста и порта в URL, соответствующих сервису согласно схеме развертывания.

liveness возвращает информацию о работоспособности сервиса.

readiness сообщает о готовности сервиса к работе.

Готовность сервиса к работе характеризуется доступностью окружающих его сервисов, с которыми ведется прямое взаимодействие.

Когда **liveness** проверка не проходит, то это сигнализирует о том что сервис мертв и должен быть как минимум перезагружен.

Когда **readiness** проверка не проходит, то это говорит о том, что проверяемый сервис не готов к принятию входящего сетевого трафика.

В будущем это приложение может прийти в готовность, но сейчас оно не должно принимать трафик.

Liveness и **readiness** проверки проходят успешно только в том случае, если все входящие в них проверки (свои для каждого сервиса) прошли успешно.

Внимание:

Для управления всеми типами health check наружу выставляются рычаги/конфиги, позволяющие включать/отключать проверки а также устанавливать частоту вызова и таймауты по каждой проверке. Readiness и Liveness проверки проходят в realtime. Таймауты для данных проверок должны быть выставлены с учетом того, что проверка сможет пройти в данный срок.

Liveness

Для **liveness** проверок вертикалов принимается следующее ограничение:

Если, например, для работы вертикала требуется Kafka, и она в данный момент времени недоступна - то, при условии что вертикал задеплоен, проверка **liveness** вернет положительный ответ. Отловить подобную ситуацию можно будет только на проверке **Readiness**.

2.3.2.2.1 BLOB-Adapter

Liveness

Проверяется состояние вертикалов. В рамках проверки осуществляется корректность деплоя вертикалов. Берется список всех вертикалов по сервису, и выполняется проверка, что они есть в списке задеплоенных вертикалов. В случае если все вертикалы задеплоены - проверка проходит успешно.

Readiness

Хранилище (внешнее или внутренне). Получение адреса хранилища. Проверка, существует ли хранилище по конкретному адресу. Порт доступен, открыт и доступен для подключения.

2.3.2.2.2 Сервис печатных форм

Liveness

Проверяется состояние вертикалов. В рамках проверки осуществляется корректность деплоя вертикалов. Берется список всех вертикалов по сервису, и выполняется проверка, что они есть в списке задеплоенных вертикалов. В случае если все вертикалы задеплоены - проверка проходит успешно.

Readiness

- ProStore: получения адреса, подключение по JDBC и проверка выполнения запроса «CHECK_VERSIONS()».
- Kafka агента ПОДД: получение строк подключения. Проверка, что Kafka существует по конкретному адресу, порт доступен, открыт и доступен для подключения.

2.3.2.2.3 СМЭВ3 адаптер

Liveness

Проверяется состояние вертикалов. В рамках проверки осуществляется корректность деплоя вертикалов. Берется список всех вертикалов по сервису, и выполняется проверка, что они есть в списке задеплоенных вертикалов. В случае если все вертикалы задеплоены - проверка проходит успешно.

Readiness

- ProStore: получения адреса, подключение по JDBC и проверка выполнения запроса «CHECK_VERSIONS()».
- VipNet SignerCP: Если в файле конфигурации указан доступ к VipNet, то по указанной строке подключения проверяется доступность VipNet.
- Postgress: Получения строки подключения, подключение по JDBC и проверка выполнения запроса «select 1».
- СМЭВ3: Проверить telnet порт СМЭВ3 на доступность командой

Опционально, проверить зафиксированные результаты обращений вертикала ресивера в СМЭВ3.

3 ЗАПУСК И ОСТАНОВКА КОМПОНЕНТА

Компонент не имеет графического интерфейса и запускается автоматически после запуска сервера.

Все модули Компонента 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 модулей приведен в разделе [Состав модулей в дистрибутиве](#) документа «Техническое описание Компонента «Витрина данных»».

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 smeysql-server-all.jar new <new-app-name>
```

Данная команда создаст структуру папок сервера и исполняемый файл **smeysql** внутри **<new-app-name>**.

3.2.1.2 Запуск и управление

Запуск СМЭВ QL Сервера осуществляется командой:

```
./smeysql 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.smeysql.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
- inaccessible_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 DATA-uploader – Модуль исполнения асинхронных заданий

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 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.

3.5.2 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
```

3.6 REST-Uploader – Модуль асинхронной загрузки данных из сторонних источников

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 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.

3.6.2 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
```

3.6.3 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 сотрудниками, загружающим данные необходимо изучить возврат REST-uploader. Если ошибка внутренняя, то нужно обратиться к администратору Витрины. Администратор изучит логи REST-uploader / Data-uploader.

3.7 BLOB-адаптер

Описание настроек модуля приведено в разделе [Настройка программных средств](#).

3.7.1 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).

3.7.2 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».

3.8 Сервис формирования документов

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 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.

3.8.2 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
```

3.9 ETL

3.9.1 Apache Airflow

Apache Airflow представляет собой набор контейнеров, управляемых **Docker**.

Ниже приведено описание запуска и остановки **Apache Airflow**.

3.9.1.1 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
```

3.9.1.2 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.yml` выполните команду:

```
docker-compose stop
```

3.9.2 Apache Spark

Apache Spark представляет собой контейнер, управляемый **Docker**.

Ниже приведено описание запуска и остановки **Apache Spark**.

3.9.2.1 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
```

3.9.2.2 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.yml` 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
```

3.9.3 Apache Hadoop

Apache Hadoop представляет собой набор контейнеров, управляемых **Docker**.

Ниже приведено описание запуска и остановки **Apache Hadoop**.

3.9.3.1 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
```

3.9.3.2 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.yml** 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
```

3.9.4 Tarantool (Vynil)

Tarantool (Vynil) представляет собой контейнер, управляемый **Docker**.

Описание запуска и остановки **Tarantool (Vynil)** приведено в файле **docker-compose.yml** директории *Tarantool*.

3.9.4.1 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
```

3.9.4.2 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.yml** выполните команду:

```
docker-compose stop
```

Для остановки отдельно каждого контейнера **Tarantool (Vynil)** можно использовать команды **Docker**:

Пример команды:

```
docker stop tarantool1  
docker stop tarantool2
```

3.10 Counter-provider - Сервис генерации уникального номера

Описание настроек модуля приведено в [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 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.

3.10.2 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
```

3.11 Arenadata Cluster Manager (ADCM)

3.11.1 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_address_of_server>:8000.
```

3. Авторизуйтесь в веб-интерфейсе.

3.11.2 Остановка

Остановка **ADCM** выполняется путём остановки **Docker** командой:

```
docker stop adcm
```

4 БЕКАПИРОВАНИЕ КОМПОНЕНТА «ВИТРИНА ДАННЫХ»

4.1 Реализация бекапирования в слое адаптеров Компонента «Витрина данных»

4.1.1 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.

4.1.1.1 Сообщения в топиках команд

Сообщения в топиках команд

Назначение команды	Топик	Ключ	Значение
Приостановка обработки запросов для модулей, которым требуется консистентность с Prostore	<code>adapter.command.broadcast</code>	<code>backup.pause</code>	<code>null</code>
Возобновление обработки запросов для модулей, которым требуется консистентность с Prostore	<code>adapter.command.broadcast</code>	<code>backup.resume</code>	<code>null</code>
Запрос персистированных данных из Zookeeper для резервной копии	<code>adapter.command</code>	<code>backup.get</code>	<code>backupId</code>
Применение данных резервной копии и запись персистированных данных в Zookeeper	<code>adapter.command</code>	<code>backup.set</code>	<code>null</code>

4.1.1.2 Статусы модулей

Статусы модулей возвращаются через компактный топик `adapter.status`

Формат сообщения статуса:

ключ в формате json (идентификатор вертикаля 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"
}
```

4.1.1.2.1 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

4.2 Механизм приостановки модулей, требующих консистентности с Prostore

Все экземпляры модулей, требующих консистентности с Prostore подписаны на топик `adapter.command.broadcast` с уникальной `groupId` консьюмера: (имя модуля)_(UUID) `groupId`.

4.2.1 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`.

4.2.2 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`.

4.3 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;
- counter-provider.

Резервное копирование выполняет только один из экземпляров каждого типа модулей, который успел считать сообщение из топика `adapter.command`.

4.3.1 Механизм резервного копирования модулей слоя адаптеров

При выполнении резервного копирования все модули, участвующие в бекапировании:

1. Подписаны на топик `adapter.command` с общей `groupId` 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 персистируемые 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":[]
    }
  ]
}
```

4.3.2 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 groupId 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`.

4.4 Поведение в случае ошибок при выполнении резервного копирования

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 находиться в не консистентном состоянии, требуется оперативный разбор ошибок и повторное восстановление из резервной копии.

4.4.1 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. [backup_error](#))

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) (instance) (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) (instance) (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 stopping (module)»	Таймаут приостановки одного из	– отправляет команду backup.resume на	Требуется анализ логов модуля, в котором возникла ошибка, после ее

(instance) (verticle) (key), see modul log for detail»	инстансов модулей, требующих консистентности с Prostore	восстановление работы модулей в топик adapter.command.broad cast без ожидания статусов о восстановлении – завершает процесс бекапирования/ восстановления с выводом ошибки: «timeout stopping (module) (instance) (verticle) (key), see modul log for detail» – выводит финальный статус: BACKUP/RESTORE is failed	устранения, повтор процесса бекапирования/восстановле ния
«timeout starting (module) (instance), see modul log for detail»	Таймаут восстановления работоспособнос ти одного из инстансов модулей, требующих консистентности с Prostore	– завершает процесс бекапирования/ восстановления завершается с выводом ошибки : «timeout starting (module) (instance), see modul log for detail» – завершает процесс восстановления с выводом ошибки: «error restoring (module) (instance), 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/восстановле ния
«timeout restoring (module) (instance), 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.broad cast без ожидания статусов о восстановлении – завершает процесс восстановления с выводом ошибки: «timeout restoring	Требуется анализ логов модуля, в котором возникла ошибка, после ее устранения, повтор процесса бекапирования/восстановле ния

		(module) (instance), see modul log for detail» – выводит финальный статус: RESTORE is failed	
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.broad cast без ожидания статусов о восстановлении – завершает процесс бекапирования с выводом ошибки, переданной DTM-tools – выводит финальный статус: BACKUP is failed	Требуется анализ логов модуля, в котором возникла ошибка, после ее устранения, повтор процесса бекапирования/восстановления
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.broad cast 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

5 ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ

5.1 Дополнительные возможности конфигурации Стандарт

Примечание:

Инструкции данного раздела не выполняются в рамках первичной установки частей Компонента. Необходимость выполнения действий данного раздела определяется в процессе эксплуатации Компонента.

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 программное обеспечение СМЭВ3-адаптер для работы со СМЭВ:

Агент СМЭВ4 не входит в состав данного решения и устанавливается отдельно, согласно соответствующей документации.

5.1.2 Материализованные 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 - 4.1](#)).

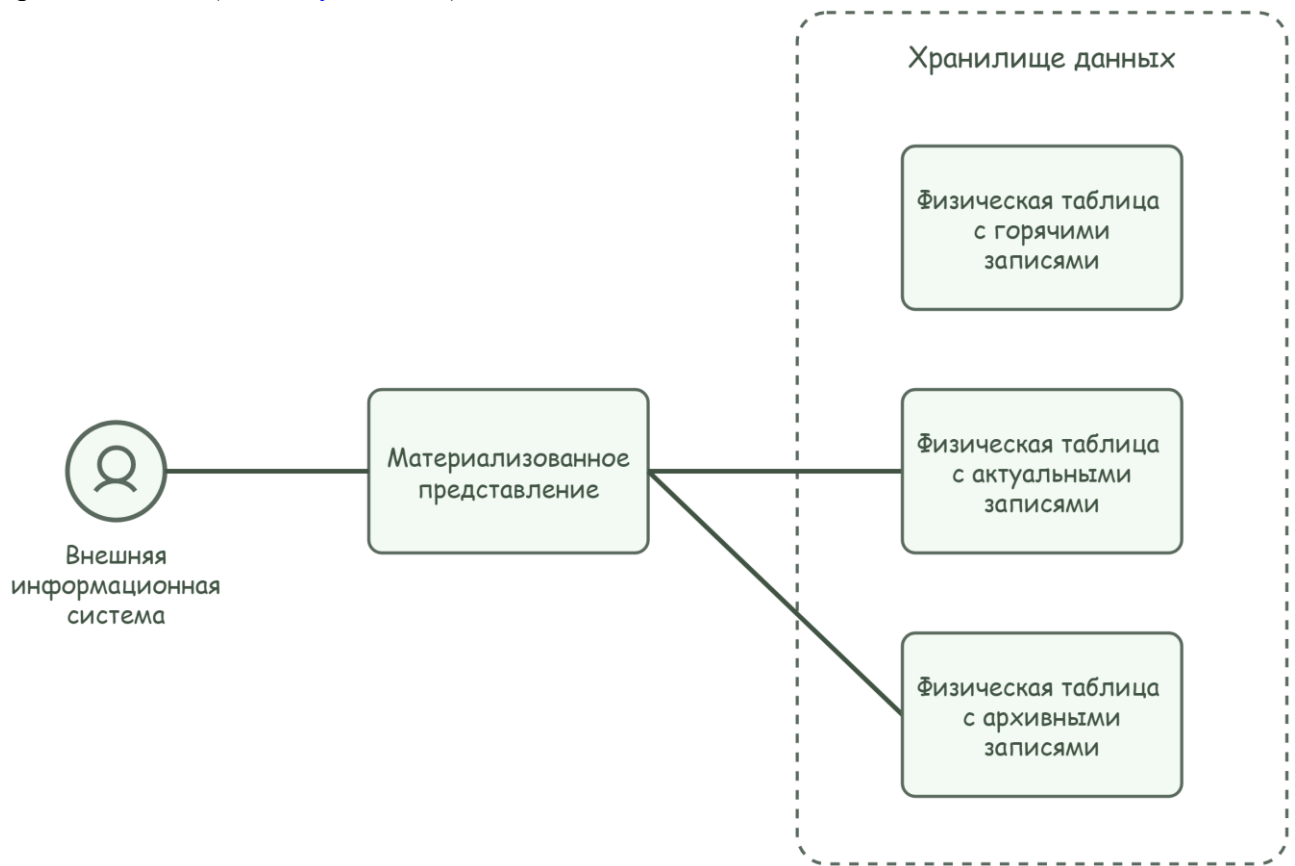


Рисунок - 4.1 Связи материализованного представления с физическими таблицами

Система поддерживает целостность данных материализованных представлений, размещенных в СУБД-приемнике, периодически синхронизируя их с СУБД-источником (см. [Рисунок - 4.2](#)).

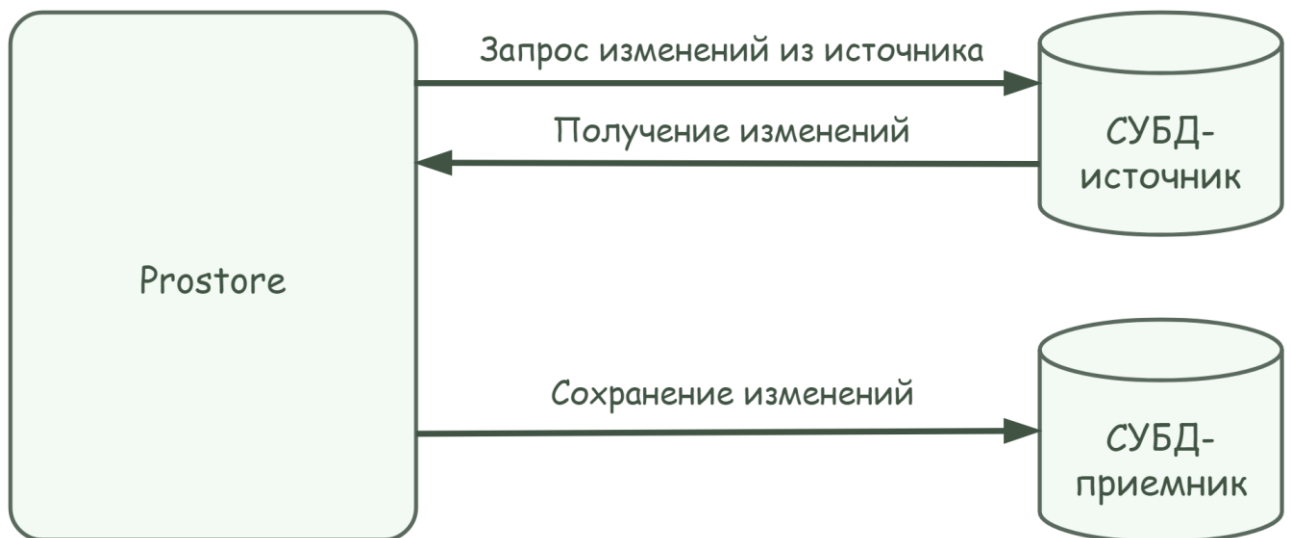


Рисунок - 4.2 Синхронизация материализованных представлений

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 - 4.3](#) и [Рисунок - 4.4](#)) 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.

Логическая таблица sales

id	product_code	product_units	store_id
100	AAA000	3	1111
101	XXX111	2	1111

Синхронизация

Материализованное представление sales_by_stores

```
SELECT store_id, product_code, SUM(product_units)
FROM sales GROUP BY store_id, product_code
```

store_id	product_code	product_units
1111	AAA000	3
1111	XXX111	2

Рисунок - 4.3 Состояние данных на момент дельты 0

Логическая таблица sales

id	product_code	product_units	store_id
100	AAA000	3	1111
101	XXX111	2	1111
102	AAA000	1	2222
103	AAA000	4	1111

Синхронизация

Материализованное представление sales_by_stores

```
SELECT store_id, product_code, SUM(product_units)
FROM sales GROUP BY store_id, product_code
```

store_id	product_code	product_units
1111	AAA000	7
1111	XXX111	2
2222	AAA000	1

Рисунок - 4.4 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 4.5](#)):

Таблица 4.5 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
СМЭВ3-адаптер	/opt/smev3-adapter/logs/application.log
ETL	/opt/Airflow/logs /opt/spark/logs /opt/hadoop/logs

5.1.5 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 Компонента «Витрина данных» и модули Prostore.

СУБД остается 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 - 4.5](#)) представлена миграция модуля исполнения запросов и Prostore.

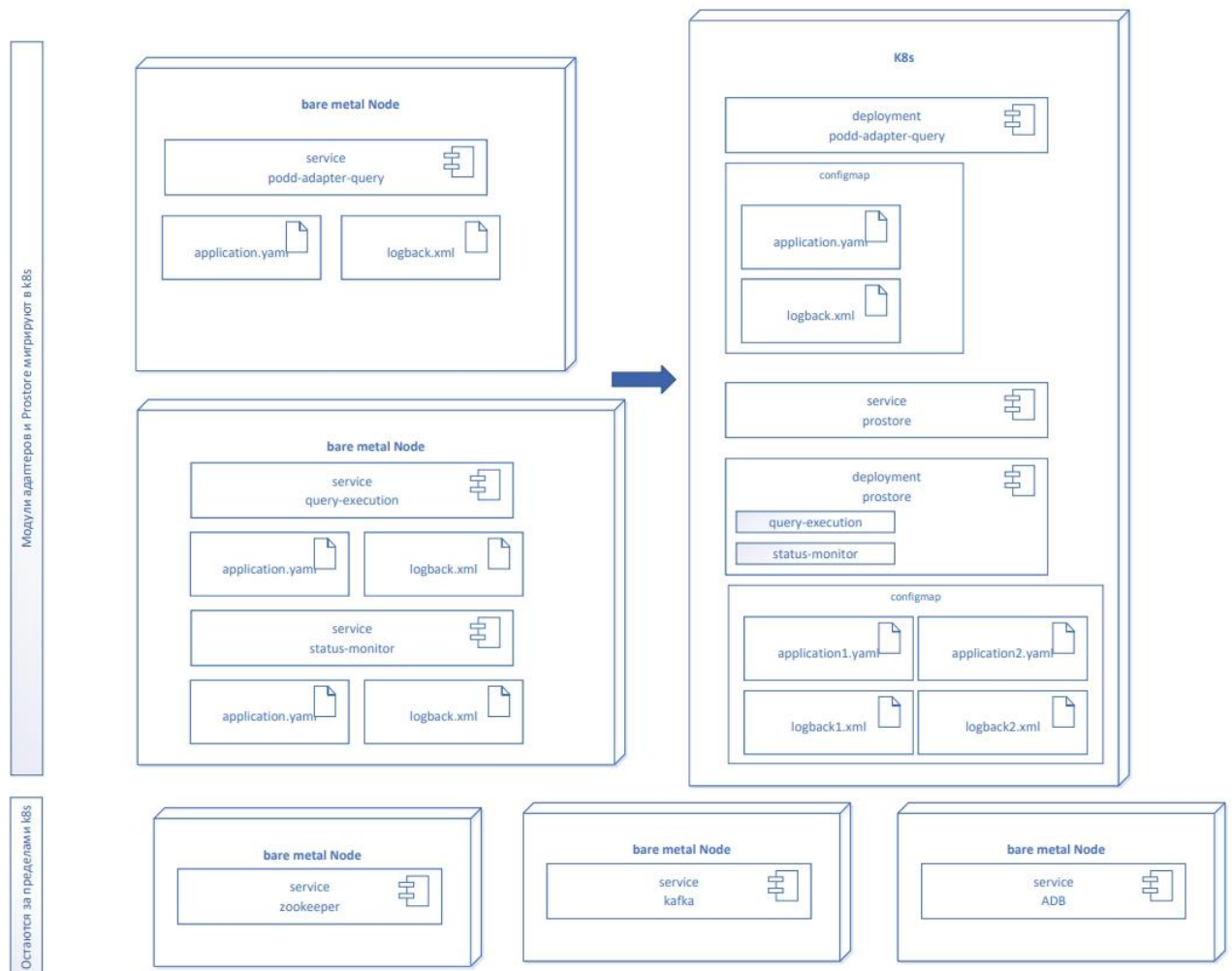


Рисунок - 4.5 Миграция в Kubernetes

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>
```

5.1.5.1 Примеры инструкций по разворачиванию Prostore в Kubernetes

Пример создания файла **deployment**

```
apiVersion: apps/v1
kind: Deployment
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
        kubect1.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.goslugi.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 Компонента.

5.2.1 Логирование

Сбор лог-файлов Компонента, 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 - 4.6](#)) по адресу <http://0.0.0.0:9010/> (авторизация: admin/somepasswordpepper).

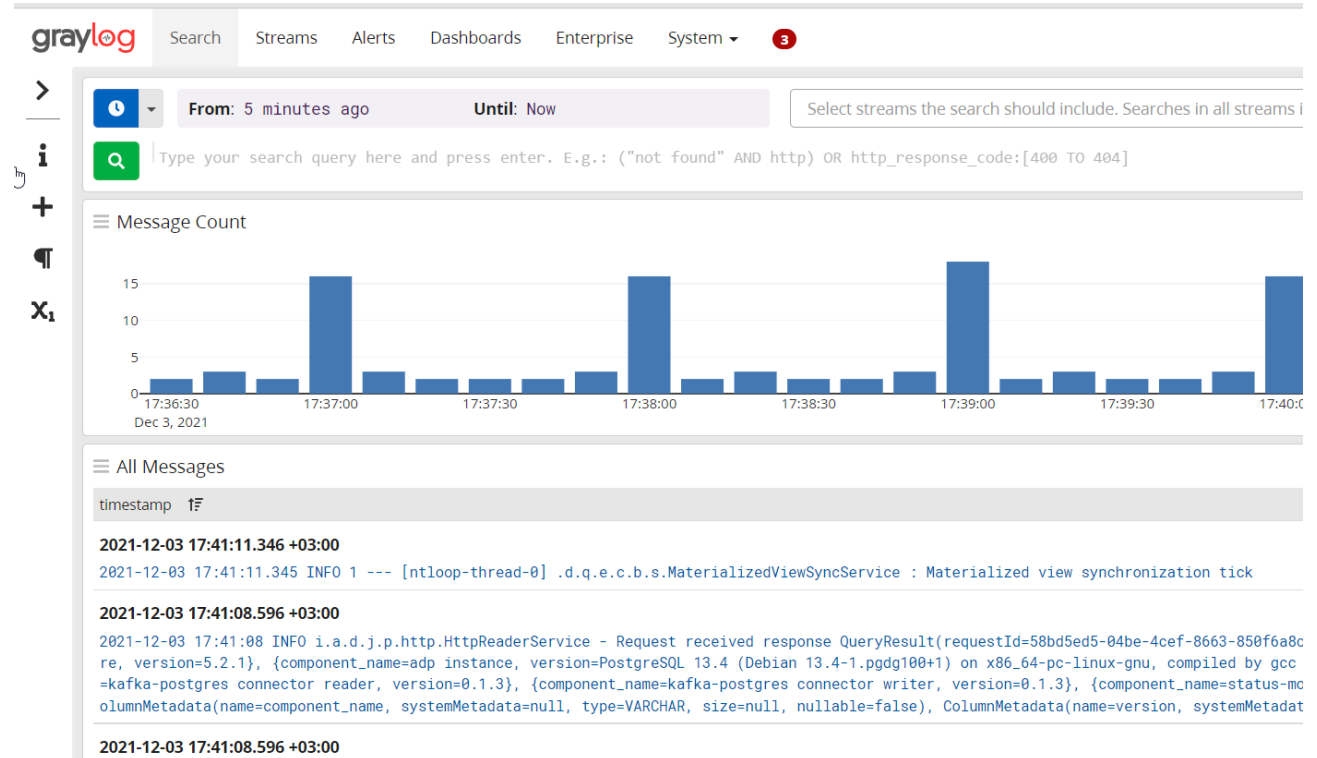


Рисунок - 4.6 Просмотр записей лог-файлов в Graylog

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 Компонента можно проверить с помощью запроса `CHECK_VERSIONS`.

6 СООБЩЕНИЯ АДМИНИСТРАТОРУ

6.1 Сообщения в ходе установки и настройки Компонента

Сообщения в ходе выполнения настройки конфигурации Стандарт

Выполнение настройки Компонента представляет собой процесс ручного или автоматизированного формирования конфигурационных файлов Компонента, 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 Компонента.

Поток сообщений о ходе выполнения настройки является частным случаем потока сообщений при выполнении проверки Компонента.

Сервисы Компонента 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 Компонента представляет собой процесс автоматизированного формирования конфигурационных файлов Компонента 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 Компонента.

Сервисы Компонента 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 Компонента

В ходе эксплуатации части Компонента формируют сообщения и выводят их в стандартный порт вывода, перенаправленный в соответствующие лог-файлы. Генерация сообщений администратору в ходе эксплуатации Компонента подчиняется следующей блок-схеме (см. [Рисунок - 5.1](#)).

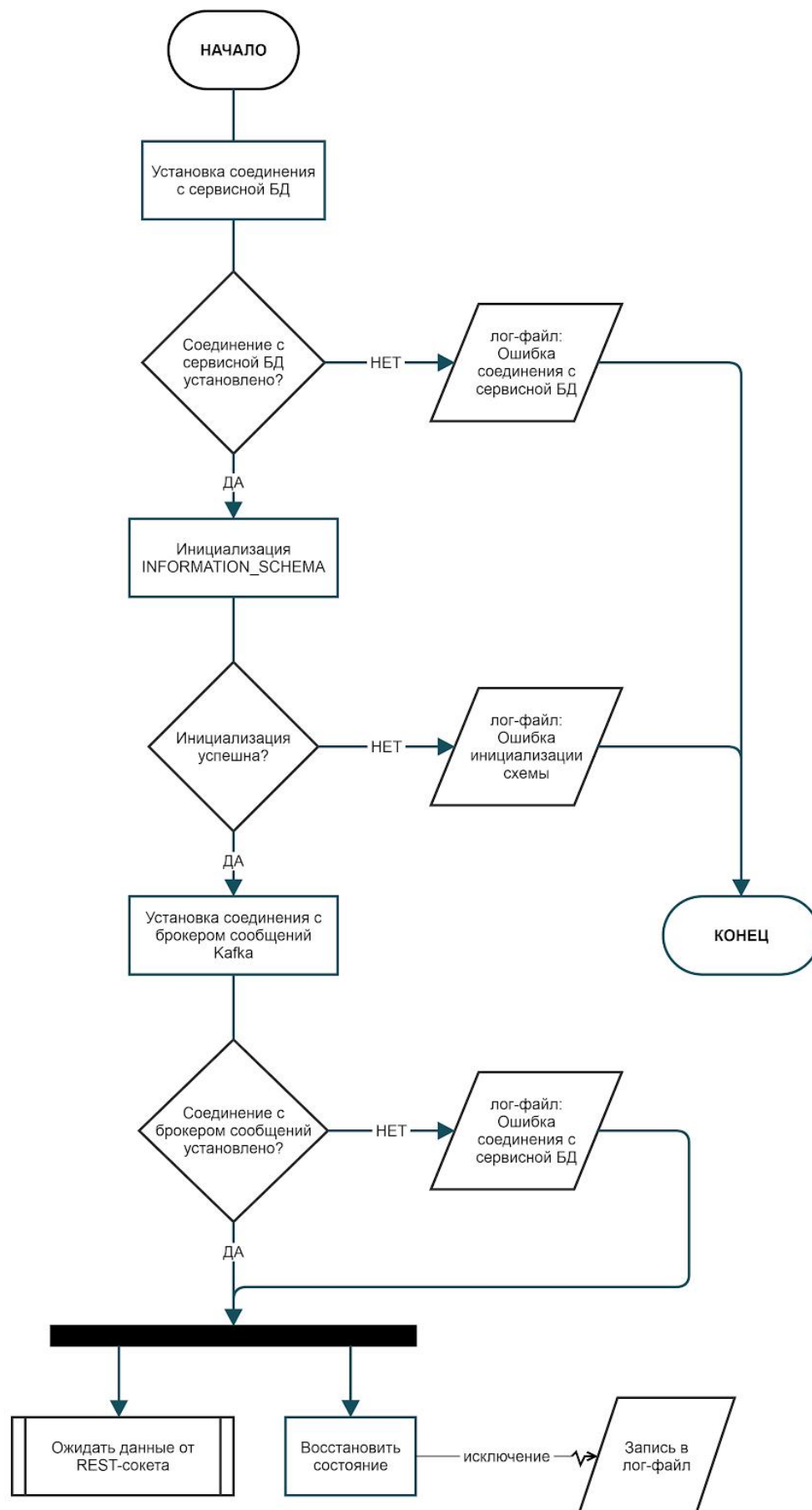


Рисунок - 5.1 Блок-схема журналирования сообщений в лог-файлы при запуске Компонента

Дополнительно, система может формировать следующие сообщения:

Таблица 5.1 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 Компонент «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 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 Сервиса генерации уникального номера (counter-provider) реализованы метрики

для функции:

- `get counter`: обработка запроса на получение номера.

Таблица 6.1 Метрики в модулях Компонента «Витрина данных»

Модуль	Сервис	Функции	Метрика	Насыщение ресурсов
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.	<code>req_count_total</code> - число запросов по функциям; <code>req_time_seconds_sum</code> - время обработки запроса для каждого функционального блока; <code>req_time_seconds_sum/req_count_total</code> - среднее время обработки.	1. <code>process_cpu_usage</code> - использование процессора; 2. <code>jvm_memory_used_bytes</code> - использование памяти: <code>jvm_memory_used_bytes{area="heap",id="PS Survivor Space",};</code> <code>jvm_memory_used_bytes{area="heap",id="PS Old Gen",};</code> <code>jvm_memory_used_bytes{area="heap",id="PS Eden Space",};</code> <code>jvm_memory_used_bytes{area="nonheap",id="Metaspace",};</code> <code>jvm_memory_used_bytes{area="nonheap",id="Code Cache",};</code> <code>jvm_memory_used_bytes{area="nonheap",id="Compressed Class Space",};</code> <code>jvm_buffer_memory_used_bytes{id="direct",};</code> <code>jvm_buffer_memory_used_bytes{id="mapped",};</code>
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.		
Сервис генерации уникального номера	counter-provider	get counter.		

ПРИЛОЖЕНИЕ 1. 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 7.1](#).

Таблица 7.1 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 Компонент «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 - 7.1](#))

Загрузчик CSV Загрузка структуры Выгрузить шаблон Загрузить Настройки Журнал операций ФЛК

Загрузка структуры витрины

Выберите файл XML для загрузки в Витрину.

Выбор файла Не выбран ни один файл

Загрузить

Рисунок - 7.1 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 - 7.2](#)).

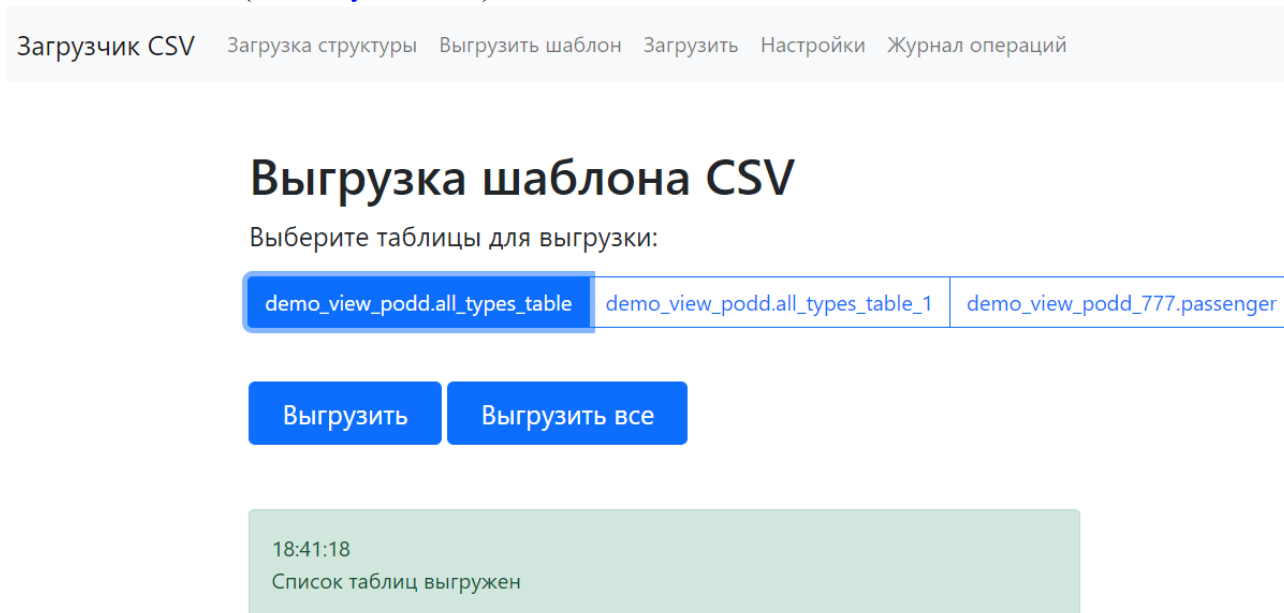


Рисунок - 7.2 Выгрузка шаблона CSV

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 - 7.3](#)).

Загрузчик CSV Загрузка структуры Выгрузить шаблон Загрузить Настройки Журнал операций ФЛК

Загрузка файла

Выберите таблицу и файл CSV для загрузки.

Режим:

Вставка

Удаление

☒ Выполнять проверку форматно-логического контроля

Таблица:

☒ Автоматическое определение таблицы

CSV:

Выбрать файлы

Файл не выбран

Загрузить

Рисунок - 7.3 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 - 7.4](#).

Загрузка файла

Выберите таблицу и файл CSV для загрузки.

Режим:

☒ Вставка
 ☐ Удаление
 ☒ Выполнять проверку форматно-логического контроля

Таблица:

☒ Автоматическое определение таблицы

CSV:

Рисунок - 7.4 Переключатель выполнения ФЛК

- 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.
- 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 - 7.5](#).

Загрузка файла

Выберите таблицу и файл CSV для загрузки.

Режим:

Вставка

Удаление

☒ Выполнять проверку форматно-логического контроля

Таблица:

☒ Автоматическое определение таблицы

CSV:

Выбрать файлы

Файл не выбран

Загрузить

Рисунок - 7.5 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 - 7.6](#)).

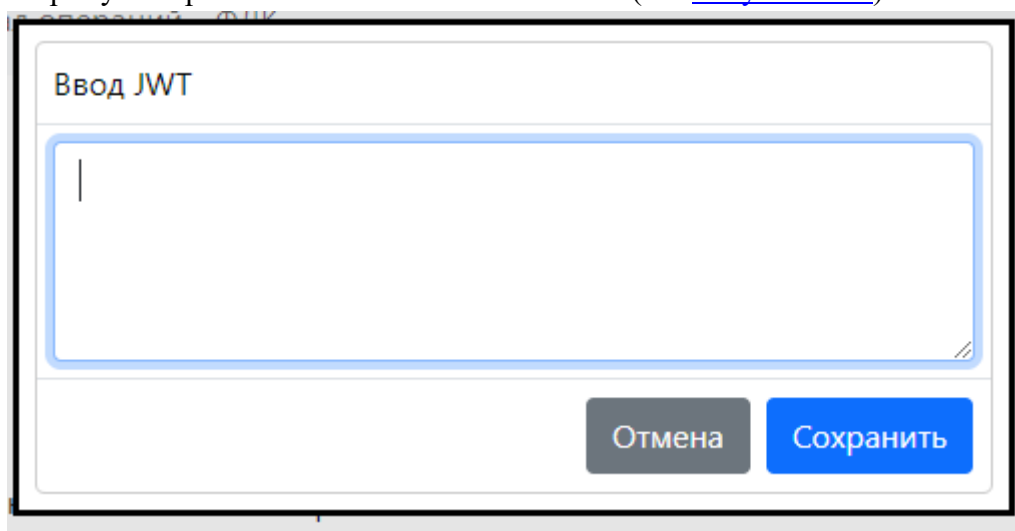


Рисунок - 7.6 Модальное окно ввода токена

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 - 7.7](#)).

The screenshot shows the 'Загрузка файла' (File Upload) section of a web application. At the top, there is a navigation bar with links: 'Загрузчик CSV', 'Загрузка структуры', 'Выгрузить шаблон', 'Загрузить', 'Настройки', 'Журнал операций', and 'ФЛК'. The main heading is 'Загрузка файла' (File Upload), followed by the instruction 'Выберите таблицу и файл CSV для загрузки.' (Select a table and CSV file for upload.). Below this, there is a 'Режим:' (Mode) section with two buttons: 'Вставка' (Insert) and 'Удаление' (Delete), and a toggle switch for 'Выполнять проверку форматно-логического контроля' (Perform format-logical control check). The 'Таблица:' (Table) section has a toggle switch for 'Автоматическое определение таблицы' (Automatic table determination). The 'CSV:' section contains a button 'Выбрать файлы' (Select files) and a status 'Файл не выбран' (File not selected). A blue button labeled 'Изменить JWT' (Change JWT) is prominently displayed. At the bottom, there is a green status bar showing the time '16:35:45' and the message 'Список таблиц выгружен' (Table list exported).

Рисунок - 7.7 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 - 7.8](#)) .

Настройки

Настройки загрузчика CSV

Запуск по расписанию:

16 49 * ? * *

Забирать CSV из каталога:

/opt/csv_files

☒ Включить

Настройка журнала

Размер страницы:

20

Применить настройки

Рисунок - 7.8 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 - 7.9](#)).

Настройка журнала

Размер страницы:

20

Применить настройки

18:42:17

Конфигурация успешно получена

Рисунок - 7.9 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 - 7.10](#)).

Журнал операций

#	Время	Уровень	Сообщение
1	2021-09-06 13:49:16	INFO	Запуск загрузчика CSV из каталога: /opt/csv_files
2	2021-09-06 13:49:16	ERROR	Исходные файлы не обнаружены
3	2021-09-06 12:49:16	INFO	Запуск загрузчика CSV из каталога: /opt/csv_files
4	2021-09-06 12:49:16	ERROR	Исходные файлы не обнаружены
5	2021-09-06 11:49:16	INFO	Запуск загрузчика CSV из каталога: /opt/csv_files
6	2021-09-06 11:49:16	ERROR	Исходные файлы не обнаружены
7	2021-09-06 10:49:16	INFO	Запуск загрузчика CSV из каталога: /opt/csv_files
8	2021-09-06 10:49:16	ERROR	Исходные файлы не обнаружены
9	2021-09-06 09:49:16	INFO	Запуск загрузчика CSV из каталога: /opt/csv_files
10	2021-09-06 09:49:16	ERROR	Исходные файлы не обнаружены

Рисунок - 7.10 Просмотр Журнала операций

4. Нажмите кнопку **Применить настройки**.

1.12 Интерфейс Форматно-логического контроля

На вкладке **Форматно-логический контроль** (см. [f1k](#)) 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.

Формато-логический контроль

Дата и время	Идентификатор файла	Статус файла	
2023-06-15 15:25:20	95b0aa20-f407-410d-99a7-d243f3132516	Статус: 3. Успешно обработан	Запросить отчет
2023-06-15 15:21:17	02dc9532-55e2-4d48-8409-15f239e0ea62	Статус: 7. Ошибки ФЛК	Запросить отчет
2023-06-15 15:19:09	e154eba4-3ac4-4acb-86af-905b30e33982	Статус: 7. Ошибки ФЛК	Запросить отчет
2023-06-15 15:14:33	2356a4ee-96ad-453f-bc3b-951cd13d026d	Статус: 7. Ошибки ФЛК	Запросить отчет

Начало 1 Конец

15:26:30
Нет данных для requestId 95b0aa20-f407-410d-99a7-d243f3132516

15:26:08
ФЛК получен

Рисунок - 7.11 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**.

ПРИЛОЖЕНИЕ 2. 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>

```

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

eXtensible 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.х, 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 РКИ 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 (P3)

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.х.

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.