ЮНИВЕРС
ДАТА

Руководство пользователя

Юниверс SmartETL 1.x

2024

ООО «ЮНИВЕРС ДАТА» оставляет за собой право вносить изменения в настоящий документ без предварительного уведомления.

Данный документ и его отдельные части в любом порядке их расположения не подлежат воспроизведению, публикации и передаче третьим лицам (вне зависимости от конечной цели совершения указанных действий) без письменного разрешения ООО «ЮНИВЕРС ДАТА».

Редакция от 02.12.2024.

Содержание

Аннотация

Документ предназначен для ознакомления с операциями по настройке и запуску ETL-процессов с поддержкой интеллектуальных помощников.

Документ рассчитан на системных администраторов, администраторов данных, системных аналитиков.

1 Сценарий работы со SmartETL

1.1 Общие сведения

Работу со SmartETL можно разделить на несколько этапов:

- Подготовка моделей данных.
- Загрузка данных.
- Трансформация данных.
- Выгрузка данных.

Этапы выполняются последовательно. Для указанных этапов работы задействуются модули Реестра моделей и Трансформации данных.

Используемая терминология

Модель данных в SmartETL – это описание структуры информации в виде json-схем и служебные данные (такие, как дата создания, номер ревизии и т.д.). Модель данных поддерживает версии draft v4, v6, v7 и 2019-09.

Маппинг в SmartETL – jolt-трансформация, используемая для преобразования json-схем из модели данных А в модель данных Б. Также включает в служебные данные моделей А и Б.

Преобразование данных – общее понятие для ETL-процессов. Включает в себя этапы загрузки, трансформации, выгрузки данных, а также может содержать дополнительные этапы, например, рассылка сообщений в сторонний сервис.

Поток данных – последовательность шагов преобразования данных. Внутри схемы преобразования данных может содержаться несколько потоков, которые могут отличаться системами-источниками, правилами качества и так далее.

1.2 Концепция ETL-решений

ETL-решения предназначены для извлечения данных из информационных систем, трансформации данных под целевую систему и выгрузки данных. Трансформация данных может иметь разный характер, который зависит от задач, которые должен решать ETL-инструмент, и от архитектуры самого решения. Как правило, в трансформацию входит:

- Изменение структуры данных (через создание маппинга с целевой системой);
- Очистка данных от ошибок (исправление явных проблем, таких, как двойные пробелы);

- Дополнение новыми данными (автозаполнение пустых атрибутов по определенным правилам).

Каждое ETL-решение имеет собственный набор возможностей и глубину проработки инструментов трансформации.

Перед трансформацией возможна валидация: проверка на полноту, корректность данных. Некоторые факторы, например, разнородность и сложность входных данных, могут делать валидацию неактуальной или невозможной.

Решаемые задачи с помощью ETL:

- Загрузка данных в систему. Периодическая или разовая.
- Обработка ошибок, возникших при вводе, переносе данных и т.д.
- Исправление различий между описаниями одной сущности и/или объединение нескольких описаний в единую сущность.

Также важной частью ETL-решения является отслеживание всех действий при трансформации данных. С помощью аудита возможно восстановить полный путь преобразований и понять, из каких именно исходных данных образовалась та или иная строка. Возможность проследить полный путь позволяет искать ошибки в трансформировании и создавать отчеты данных.

1.3 Этап подготовки моделей данных

Для подготовки используется приложение реестра моделей. Подготовка считается завершенной, когда готовы:

1. Исходная модель данных.
2. Целевая модель данных.
3. Маппинг между моделями данных.

Используйте следующие инструкции:

- Добавление модели данных: пп. 2.1.
- Редактирование модели данных: пп. 2.2.
- Добавление маппинга: пп. 2.3.
- Редактирование маппинга: 2.4.

Модели данных не делятся по признаку «источник – получатель». Таким образом, в одном маппинге модель может использоваться как исходная, а в другом маппинге как целевая.

1.4 Этап загрузки данных

На этом этапе используется приложение трансформации данных.

Для получения данных из систем-источников требуется наличие процессора загрузки данных. Ввиду того, что системы-источники могут иметь значительные отличия – использовать разные модели данных, разный формат информационного взаимодействия – то для загрузки данных в SmartETL необходимо написание собственного процессора загрузки.

Создание процессора загрузки данных для приложения трансформации выполняется силами клиента (заказчика) с учётом следующих параметров:

- Должна использоваться модель данных, добавленная в реестр моделей.
- Процессор загрузки должен впоследствии взаимодействовать с процессором трансформации данных.

После создания и настройки процессора рекомендуется все используемые блоки объединить в группу. Убедитесь, что процессор имеет input-порт и output-порт.

В комплекте поставки SmartETL присутствует стандартный шаблон, который содержит пример реализации в виде процессора загрузки файлов с локального компьютера (с того же сервера, на котором развернуто приложение трансформации).

1.5 Этап трансформации данных

На этом этапе используется приложение трансформации данных. Процессор трансформации загружается в SmartETL через шаблон.

Этап подразумевает предварительную настройку связи с реестром моделей. Инструкцию по настройке связи см. в пп. 3.2.

Трансформация производится для данных, которые приходят из процессора загрузки. Далее данные будут выгружаться в целевую систему через соответствующий процессор.

1.6 Этап выгрузки данных

На этом этапе используется приложение трансформации данных.

Выгрузку данных можно организовать следующими способами:

- **Использовать процессор выгрузки данных в Юниверс MDM**. Процессор загружается в приложение трансформации в составе шаблона для SmartETL. Для выгрузки необходимо соблюдение условий: в Юниверс MDM должна быть создана модель данных; целевая модель данных в реестре моделей должна совпадать с моделью Юниверс MDM. Процессор выгрузки в Юниверс MDM осуществляет

дополнительную трансформацию данных из канонического вида, в вид, подходящий для Юниверс.

- **Написать собственный процессор выгрузки**, учитывающий особенности целевой системы-источника. Создание процессора для приложения трансформации выполняется силами клиента (заказчика).

После создания и настройки процессора рекомендуется все используемые блоки объединить в группу. Убедитесь, что процессор имеет input-порт.

После реализации процессоров загрузки и выгрузки данных необходимо соединить три процессора в единый поток данных. Подробнее см. 3.4.

2 Модуль реестра моделей

2.1 Добавление новой модели данных

Предварительные условия:

- Войдите в приложение «Реестр моделей», если это не сделано ранее.
- Авторизуйтесь, используя логин и пароль.
- В результате действия загрузится приложение.
- Убедитесь, что активна закладка «Модель данных».

Чтобы **добавить новую модель данных** в реестр моделей:

1. Убедитесь, что в левой части экрана размещается таблица с перечнем загруженных моделей данных. Если модели не загружены, то таблица будет пустой.
2. Нажмите кнопку «Создать модель», расположенную в нижней части модели данных.
3. В правой части экрана отобразятся данные новой модели данных (Рисунок 1).
4. Укажите имя модели.
5. Укажите описание модели (не обязательно).
6. Укажите схему модели данных.
    - Схема может быть подготовлена заранее.
    - Поддерживаемые форматы схемы: JSON-schema.
    - Схема должна содержать обязательный параметр "$schema", в котором должна быть указана ссылка на используемую JSON-schema. Сама схема должна придерживаться правил указанной версии. Пример указания версии схемы: `"$schema": "http://json-schema.org/draft-07/schema#"`. Пример документации на версию схемы: https://json-schema.org/draft-07/json-schema-release-notes.html. Список поддерживаемых версий: draft v4, v6, v7 и 2019-09.
    - Схема модели данных проверяется на ошибки. Ошибки синтаксиса будут подсвечены, и работа с ошибочной схемой будет невозможна.
7. Чтобы применить изменения нажмите кнопку «Сохранить» в правом нижнем углу.
8. Сохраненной схеме данных присваивается ID. ID используется для того, чтобы различать схемы данных. Например, при получении схемы во время валидации входных данных.

Доступен альтернативный способ создания модели данных: автогенерация на основе данных.

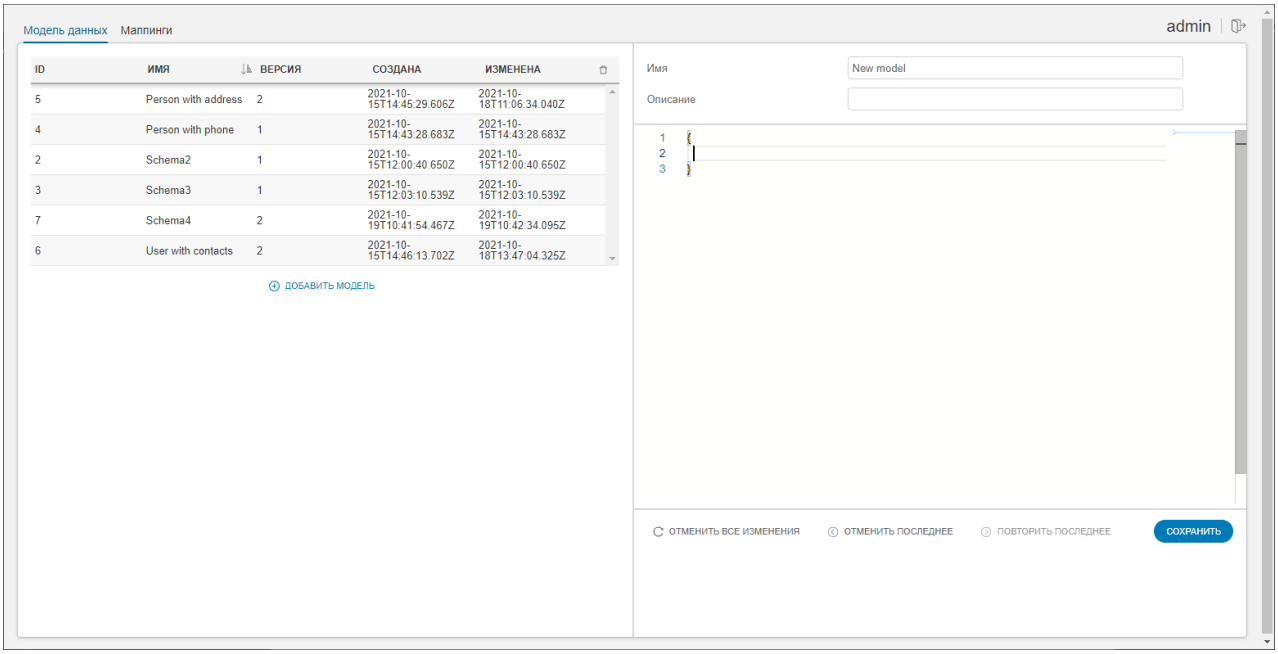

Рисунок 1 – Добавление новой модели

2.2 Редактирование модели данных

Предварительные условия:

- Войдите в приложение «Реестр моделей», если это не сделано ранее.
- Авторизуйтесь, используя логин и пароль.
- В результате действия загрузится приложение.
- Убедитесь, что активна закладка «Модель данных».

Чтобы **редактировать модель данных**:

Редактирование моделей данных производится под ответственность пользователя. Если схема модели данных используется в маппингах, то после редактирования схемы, следует **исправить все маппинги вручную**. В настоящий момент в маппингах не производится проверка актуальности используемых моделей данных и настроек соответствия.

1. Найдите требуемую модель данных в списке.

2. Выберите модель данных. Для этого кликните по соответствующей строке в списке.
3. В результате в правой части экрана отобразятся данные модели данных (Рисунок 2).
4. Внесите требуемые изменения в соответствующие поля.
5. При изменении схемы модели данных убедитесь, что измененная схема не содержит ошибок.
6. Чтобы применить изменения нажмите кнопку «Сохранить» в правом нижнем углу.

Если модель данных уже используется в маппингах, то в нижней части экрана будет выведено предупреждение со списком задействованных маппингов.

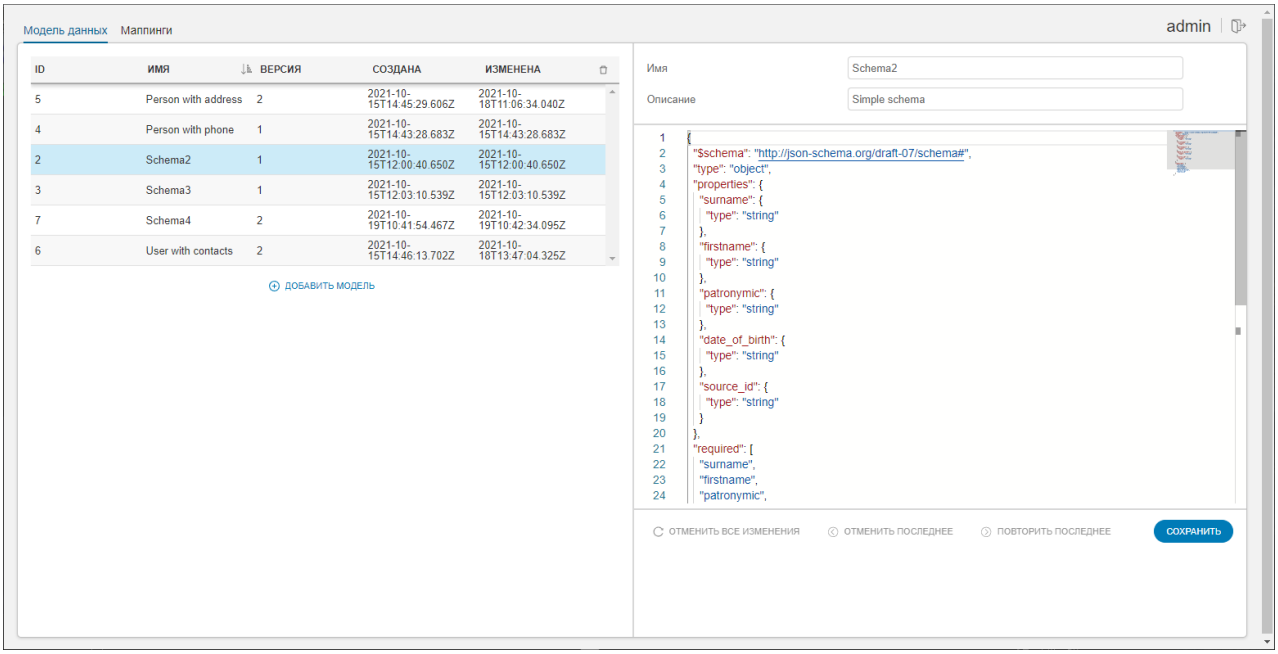

Рисунок 2 – Редактирование модели

2.3 Добавление маппинга между моделями данных

Предварительные условия:

- Войдите в приложение «Реестр моделей», если это не сделано ранее.
- Авторизуйтесь, используя логин и пароль.
- В результате действия загрузится приложение.
- Убедитесь, что создано минимум 2 модели данных, между которыми необходимо создать маппинг.
- Убедитесь, что активна закладка «Маппинги».

Чтобы **добавить новый маппинг** в реестр моделей:

1. На экране отображается таблица с перечнем маппингов между моделями данных. Если маппингов нет, то таблица будет пустой.
2. Нажмите кнопку «Добавить маппинг», расположенную в нижней части списка.
3. В результате действия откроется выдвижная панель с настройками маппинга (Рисунок 3).
    - Настройки разделены на три столбца. Левый и правый столбцы используются для выбора начальной и конечной схем. Центральный столбец используется для задания соответствия элементов одной схемы к другой.
4. В левом столбце выберите исходную схему модели данных.
5. В правом столбце выберите целевую схему модели данных.
6. Настройте соответствие каждого элемента первой схемы каждому элементу второй схемы.
    - Если в схеме присутствуют элементы с вложенностью 2 уровня и ниже, то в выпадающем списке такие элементы имеют вид [уровень1.уровень2].
    - В Расширенном режиме задание соответствия элементов в центральном столбце производится с помощью библиотеки Jolt.
7. Чтобы применить изменения нажмите кнопку «Сохранить» в правом нижнем углу.

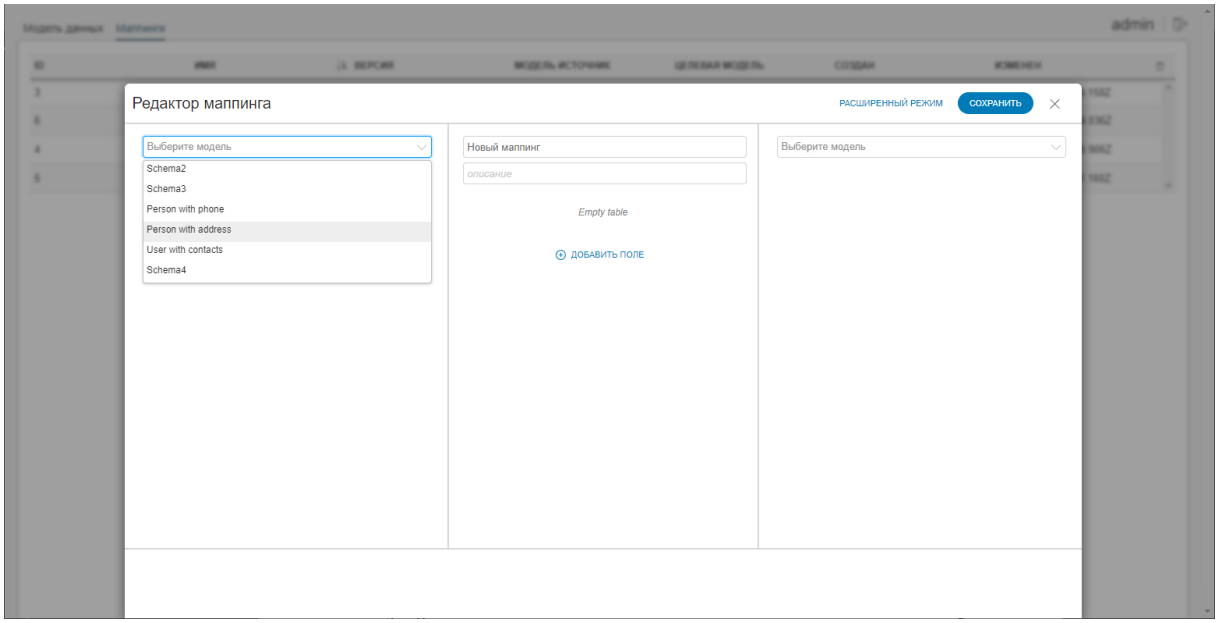

Рисунок 3 – Добавление маппинга

2.4 Редактирование маппинга между моделями данных

Предварительные условия:

- Войдите в приложение «Реестр моделей», если это не сделано ранее.
- Авторизуйтесь, используя логин и пароль.
- В результате действия загрузится приложение.
- Убедитесь, что активна закладка «Mappings».

Чтобы **редактировать маппинг**:

1. Найдите требуемый маппинг в списке.
2. Выберите маппинг. Для этого кликните по соответствующей строке в списке.
3. В результате действия откроется выдвижная панель с настройками маппинга (Рисунок 4).
4. Внесите требуемые изменения в центральном столбце.
5. Если при добавлении маппинга был включен Расширенный режим, то редактирование соответствия элементов будет доступно с помощью библиотеки Jolt. Смена режима сбрасывает все изменения предыдущего режима.
6. Чтобы применить изменения нажмите кнопку «Сохранить» в правом нижнем углу.

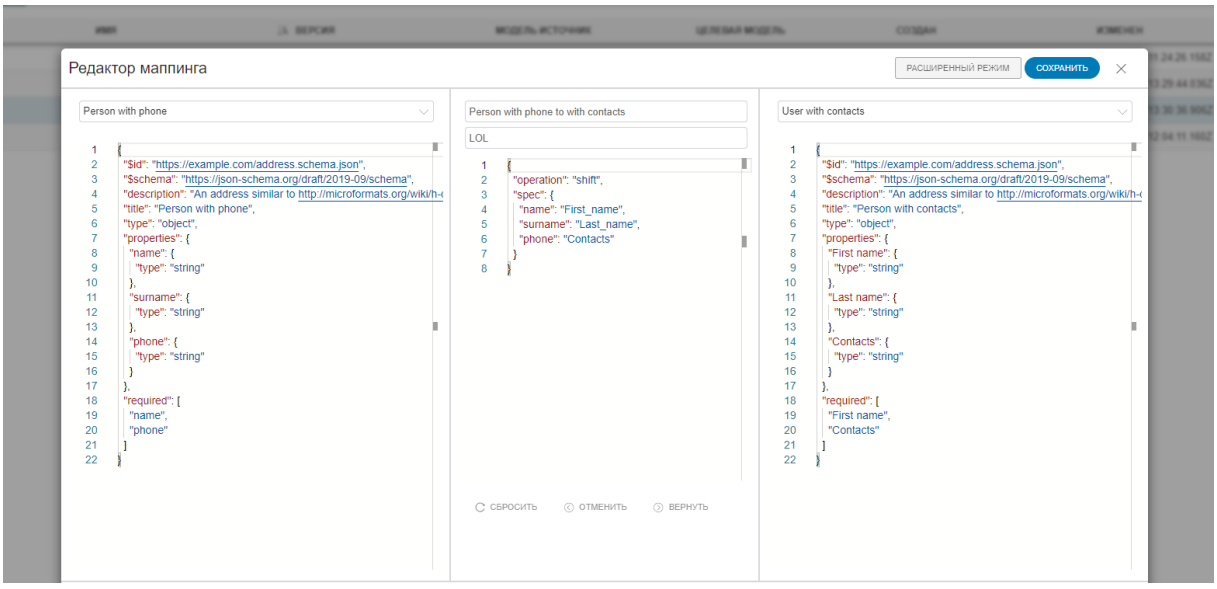

Рисунок 4 – Редактирование маппинга

2.5 Автогенерация модели данных на основе данных

2.5.1 Общие сведения

В Модуле реестра модели доступен Rest API, который может генерировать JSON-схему модели данных, основываясь на массиве JSON-описаний записей.

Для функции автогенерации используется специальный процессор для Модуля трансформации данных.

Создание модели на основе данных доступно после настройки системы.

2.5.2 Получение массива JSON-записей из Excel

Доступна возможность получения массива JSON-записей из файлов .xlsx для дальнейшей генерации JSON-схем. Для этого используется специальный шаблон Apache NiFi.

Шаблон вычитывает содержимое.xlsx, конвертирует его в .csv, затем в массив .json.

Настройка функции

Предварительные условия:

- Установлен Модуль трансформации данных.
- Загружен основной шаблон для SmartETL (пп. 3.1).

Выполните действия:

1. Установите и настройте процессор SchemaGenerator, если это не сделано ранее. См. пп. 4.6.
2. Распакуйте шаблон XLSX_to_JSON.xml, поставляемый вместе с дистрибутивом SmartETL.
3. Установите шаблон. Инструкцию по установке см. в пп. 3.2.
4. Добавьте шаблон в требуемую группу, и используйте содержимое шаблона для построения потока данных, в который включено получение массива JSON-записей из Excel.

2.5.3 Создание модели на основе данных

Предварительные условия:

- Установлен Модуль трансформации данных.

- Загружен основной шаблон для SmartETL (пп. 3.1).
- Загружен шаблон XLSX_to_JSON.xml (пп. 2.5.2).
- В Модуле трансформации данных настроен поток данных, использующий процессор SchemaGenerator (пп. 4.6).
- Войдите в приложение «Реестр моделей», если это не сделано ранее.
- Авторизуйтесь, используя логин и пароль.
- Убедитесь, что активна закладка «Модель данных».

Чтобы **сгенерировать модель данных**:

1. Войдите в интерфейс Модуля реестра моделей.
    - В левой части экрана размещается таблица с перечнем загруженных моделей данных. Если модели не загружены, то таблица будет пустой.
2. Нажмите кнопку «Добавить модель», расположенную в нижней части модели данных.
3. В раскрывшемся меню выберите пункт «Сгенерировать на основе данных».
4. В результате откроется модальное окно для ввода данных. Доступно два вариант действий:
    - Либо вставьте один или несколько объектов данных в виде массива.
    - Либо загрузите файл с массивом JSON-записей.
5. Нажмите «Далее».
6. Укажите имя и описание создаваемой модели данных. При необходимости внесите изменения в сгенерированную схему.
7. Нажмите «Создать».
8. После подтверждения шага новая модель добавится к существующим моделям в таблице, и будет доступна для выбора на экране маппингов.

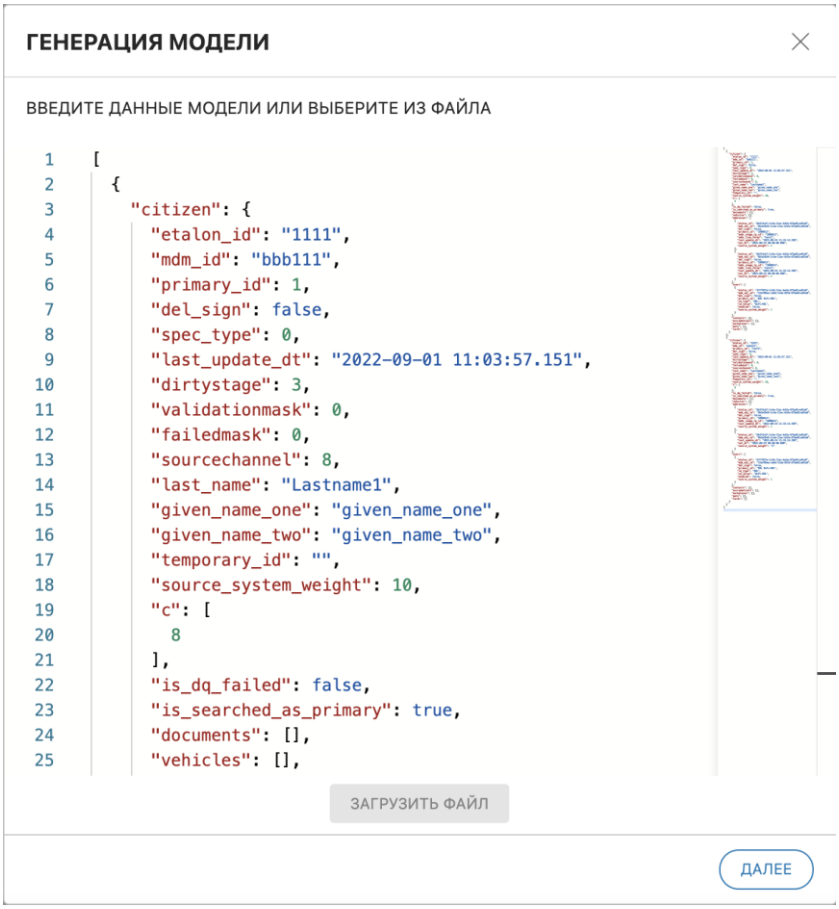

Рисунок 5 – Ввод данных, на основе которых будет генерироваться модель

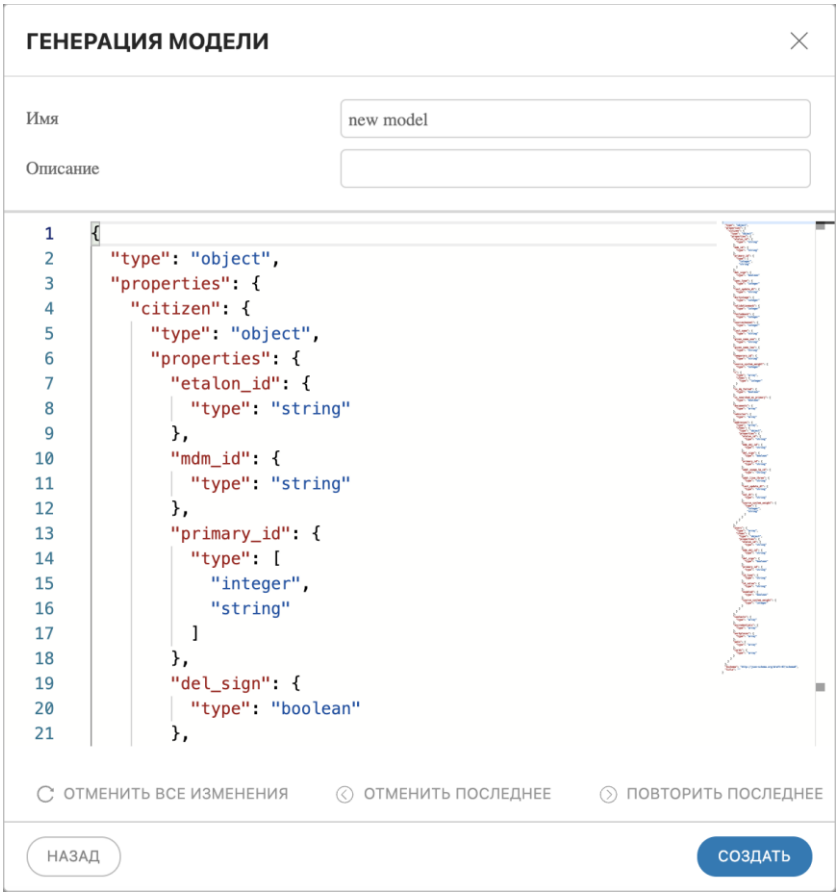

Рисунок 6 – Завершение генерации модели

2.6 Загрузка модели данных Юниверс MDM

При интеграции Юниверс SmartETL с Юниверс MDM доступна возможность загрузки модели данных в Модуль реестра моделей.

Предварительные условия:

- Войдите в приложение «Реестр моделей», если это не сделано ранее.
- Авторизуйтесь, используя логин и пароль.
- В результате действия загрузится приложение.

Чтобы **загрузить модель данных** Юниверс MDM:

1. Перейдите на вкладку «Настройки».
2. Укажите параметры импорта.
    - В поле «Имя источника» укажите отображаемое имя источника.

- В поле «Тип источника» выберите тип: Юниверс.
- В поле «URL» укажите ссылку на приложение Юниверс MDM.
- В поле «Логин» укажите логин пользователя, от имени которого будет осуществлен вход в Юниверс MDM и будет запущен импорт. Пользователь должен иметь соответствующие права доступа.
- В поле «Пароль» укажите пароль пользователя.

3. Нажмите «Сохранить». Настройки будут сохранены.
4. Перейдите на вкладку «Модель данных».
5. Нажмите кнопку «Импорт», расположенную в нижней части экрана.
6. Выберите источник данных, который был указан в настройках (Он должен быть единственным в списке).
7. Нажмите кнопку «Продолжить».
8. В результате произойдет подключение к Юниверс MDM. Будет отображен список моделей данных, доступных для добавления в реестр моделей.
   - Если модель Юниверс MDM ранее уже загружалась, то будет обозначено, что она уже существует. При необходимости можно обновить модель.
9. После выбора модели нажмите кнопку «Импорт».
10. В результате выбранные модели будут импортированы и станут доступными для дальнейшего редактирования в интерфейсе приложения.

3 Модуль трансформации данных

Модуль трансформации строится на основе Apache NiFi и специального шаблона процессов, включает в себя процессоры обработки данных. Инструкции в настоящем разделе описывают работу в интерфейсе модуля трансформации.

3.1 Основной шаблон SmartETL

Установка

Шаблоны SmartETL поставляются вместе с дистрибутивом системы (отдельным архивом). Основным шаблоном является SmartETL General Template. Для работы Модуля должен быть установлен шаблон. Инструкцию по установке см. в пп. 3.2.

Описание шаблона

При запуске приложения отобразится блок «SmartETL», который является группой для всех вложенных процессоров (Рисунок 7).

Внутри группы расположено несколько групп, каждая из которой содержит процессор:

- Read Data from Files – процессор загрузки файлов с локального компьютера.
- Transform To Canonical Model – процессор трансформации данных к каноничному виду.
- Load To Universe – процессор выгрузки данных в Юниверс MDM.

Все процессоры, кроме Smart To Canonical Model, связаны в единую цепочку. При реализации собственных процессоров связи между процессами могут быть перестроены, однако в любом случае должна соблюдаться последовательность Extract – Transform – Load (в отдельных случаях Extract – Load – Transform).

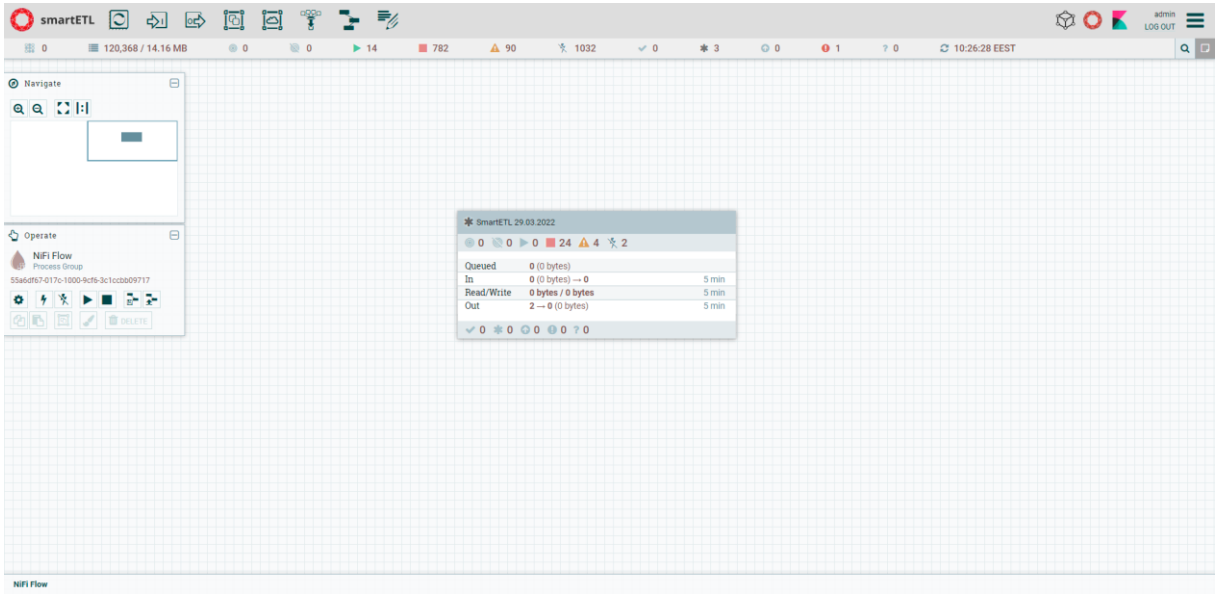

Рисунок 7 – Корень шаблона SmartETL

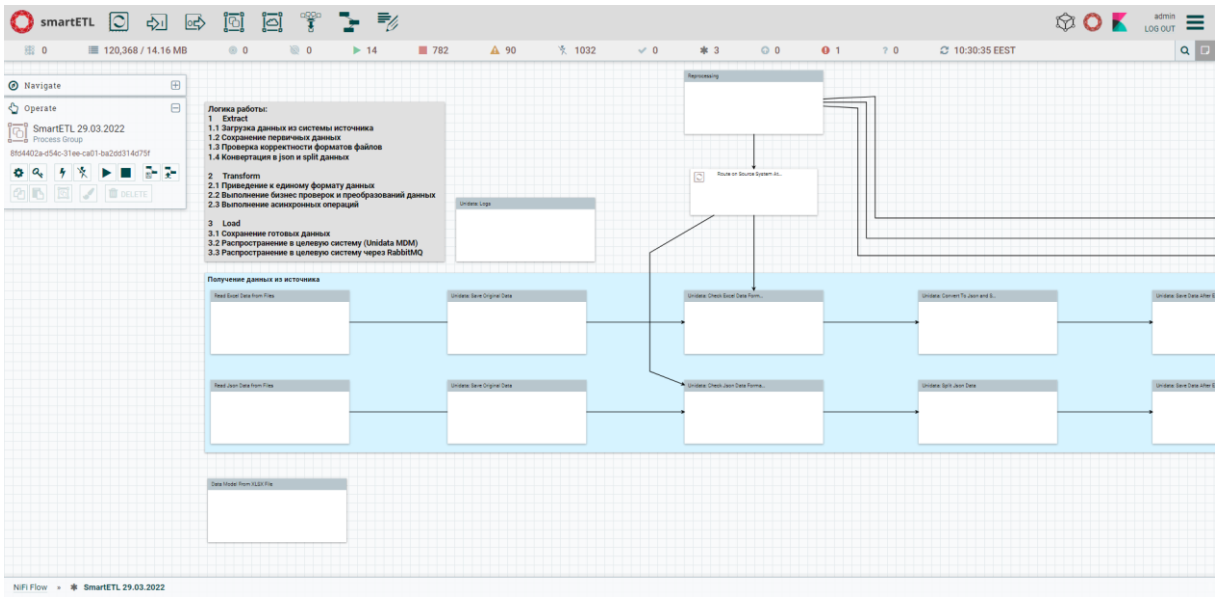

Рисунок 8 – Второй уровень шаблона SmartETL

Содержимое процессоров представлено на рисунках ниже (Рисунок 9 – Рисунок 11):

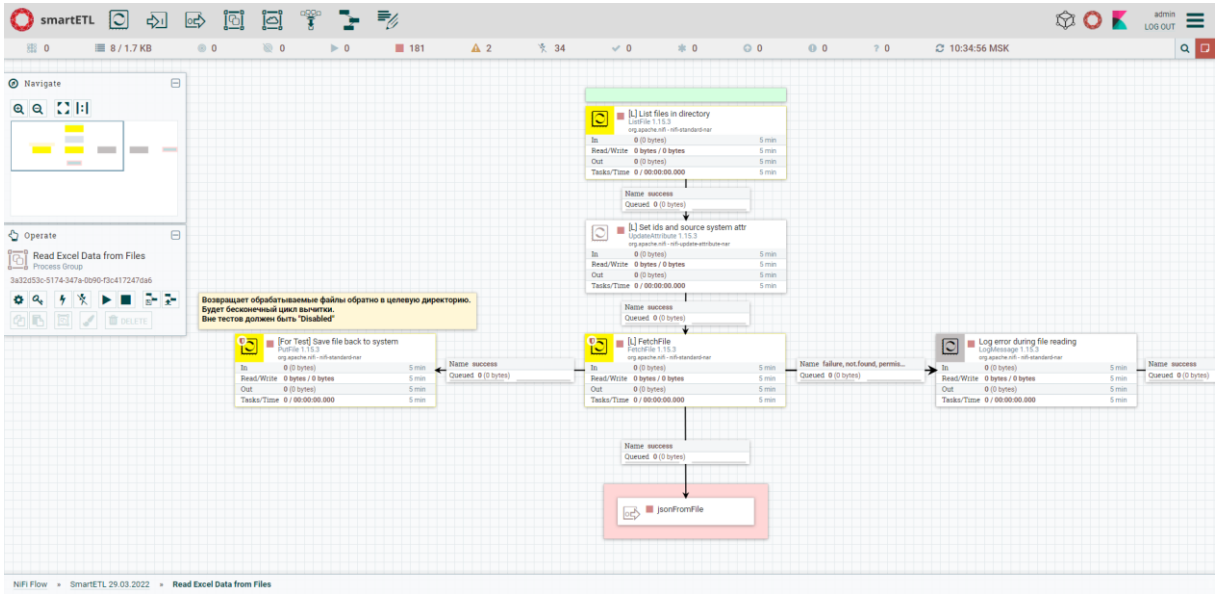

Рисунок 9 – Процессор загрузки файлов с локального компьютера

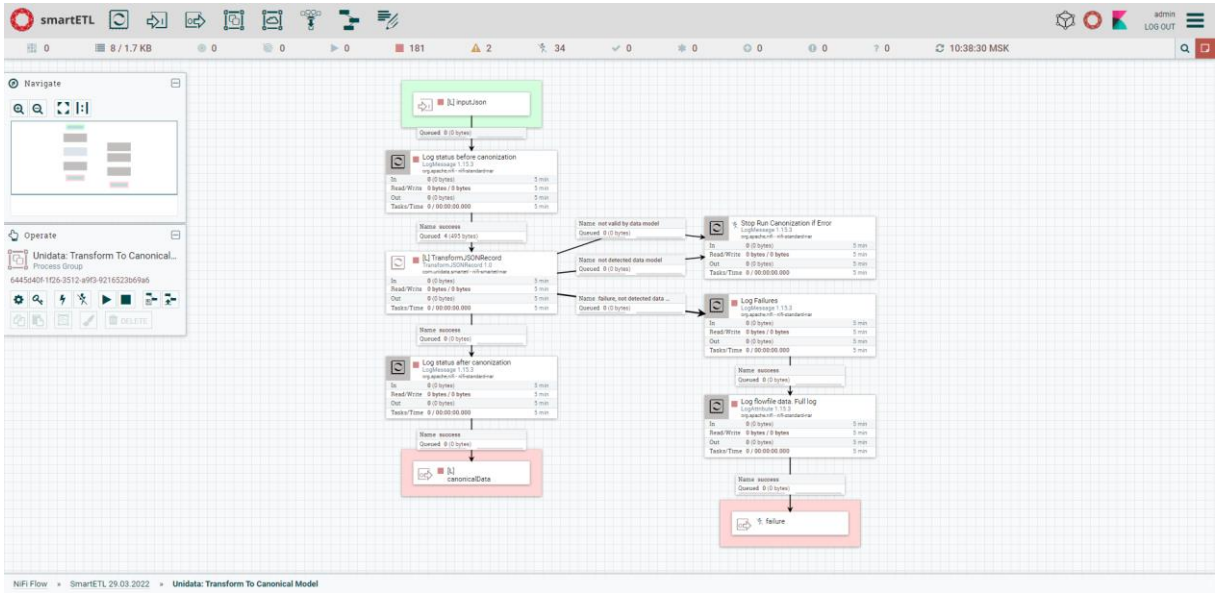

Рисунок 10 – Процессор трансформации данных к каноничному виду

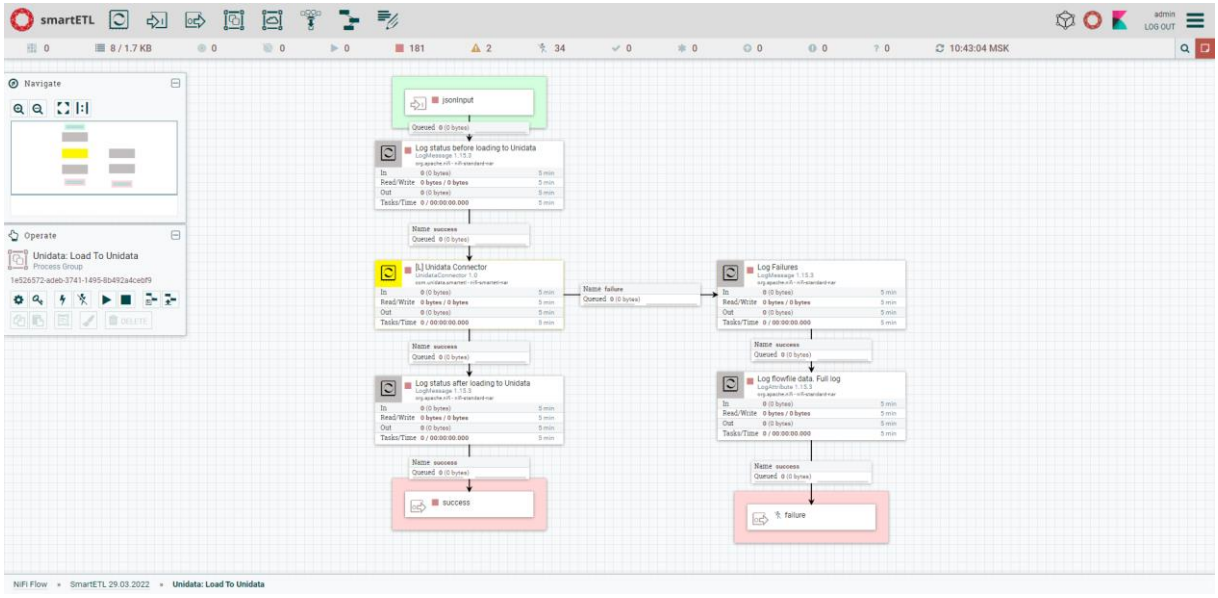

Рисунок 11 – Процессор выгрузки данных в Юниверс MDM

3.2 Управление шаблонами

Чтобы **загрузить готовый шаблон**:

- Раскройте меню Operate. Для этого нажмите кнопку , расположенную на левой границе экрана.
- В появившемся меню нажмите кнопку Upload template (последняя в списке).
- В результате откроется модальное окно.
- Нажмите Select template и выберите требуемый .xml-файл шаблона.
- Нажмите Upload.

Чтобы **использовать готовый шаблон в группе**:

- Перейдите в требуемую группу.
- Наведите курсор на иконку добавления шаблонов (Рисунок 12), расположенную в левом верхнем углу экрана.
- Зажмите ЛКМ и перетащите иконку на рабочую область.
- В результате действия откроется список добавления шаблонов.
- Выберите требуемый шаблон из списка.
- Нажмите кнопку Add.

- В результате будут добавлены блоки и настройки, записанные в шаблон (как правило, собранные в отдельную группу). В любой группе допустимо использовать несколько одинаковых групп, созданных из шаблона.
- После добавления содержимое может быть отредактировано. Это не повлияет на шаблон, из которого содержимое создано.

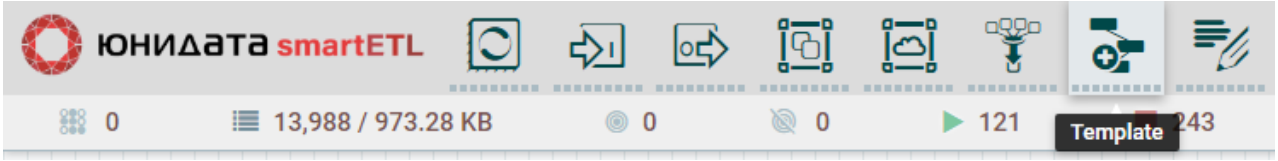

Рисунок 12 – Иконка добавления шаблонов

Для **создания шаблона** схемы потока данных:

- Нажмите ПКМ в произвольном месте рабочей области.
- В контекстном меню выберите пункт Create template.
- В результате откроется модальное окно.
- Введите имя шаблона. При необходимости добавьте описание.
- Нажмите кнопку Create.
- В результате отобразится оповещение о создании шаблона.

Чтобы **открыть список** шаблонов:

- Нажмите кнопку ☰ в правом верхнем углу экрана.
- Выберите пункт Templates.
- В результате отобразится окно со списком загруженных и созданных шаблонов (Рисунок 13).
- Чтобы удалить шаблон нажмите кнопку Delete, расположенную в правой колонке таблицы.
- Чтобы скачать шаблон в формате .xml нажмите кнопку Download, расположенную в правой колонке таблицы.

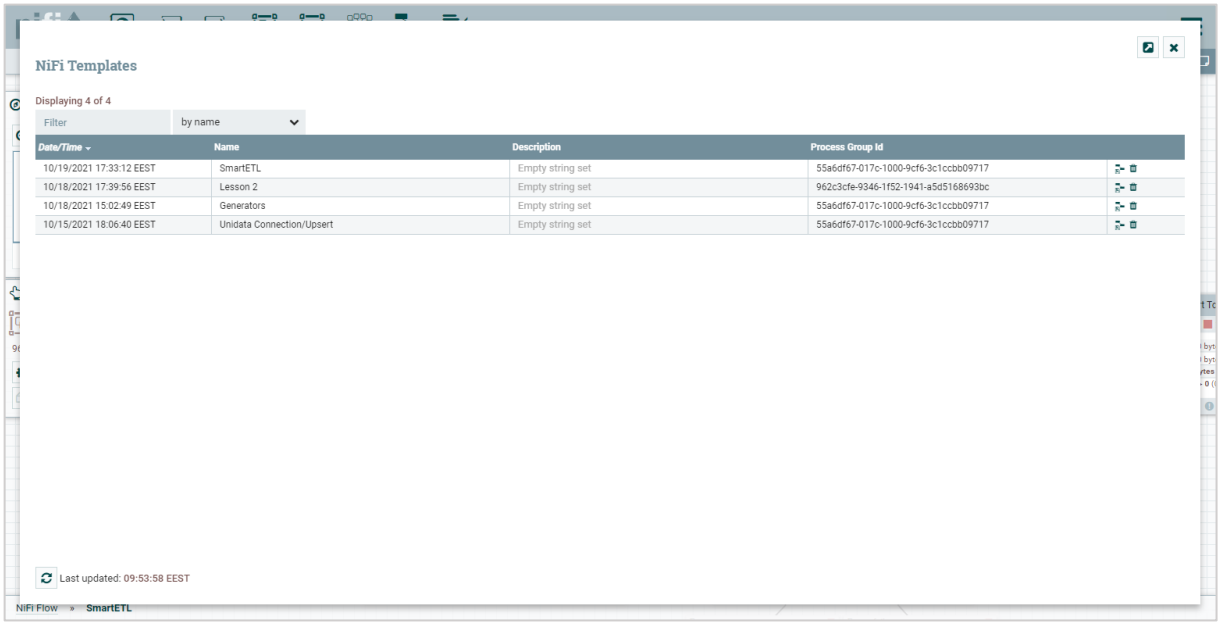

Рисунок 13 – Список шаблонов

3.3 Настройка связи процессора трансформации с реестром моделей

Предварительные условия:

- Установлено приложение трансформации данных.
- Загружен шаблон для SmartETL.
- В реестре моделей созданы необходимые модели данных и маппинги.

Для того, чтобы трансформация данных выполнялась, необходимо в процессоре трансформации указать параметры подключения к реестру моделей. Чтобы подключить процессор к реестру:

1. Откройте Apache NiFi в браузере, если это не сделано ранее. Адрес строится следующим образом:

```
https://SERVER_HOSTNAME:8443/nifi/
```

2. Откройте группу SmartETL. Для этого либо дважды кликните по блоку, либо наведите курсор на блок, вызовите контекстное меню и выберите Enter group.
3. Откройте группу Transform To Canonical Model.
4. Откройте настройки блока TransformJSONRecord. Для этого наведите курсор на блок, нажмите ПКМ и выберите Configure.
5. Переключитесь на закладку Properties в открывшемся окне (Рисунок 14).

6. Нажмите кнопку →, расположенную в крайней правой колонке строки SmartETL Model Registry Service.

7. В результате действия откроется окно настроек потоков файлов (Рисунок 15).

8. Создайте новый и/или отредактируйте существующий сервис. Подробнее см. ниже.

9. Если подключение выполнено верно, то в списке сервисов в столбце State требуемый сервис будет иметь статус Enabled.

    - Для смены сервиса выключите текущий сервис (кнопка Disable в правом столбце), обновите список (кнопка Refresh в левом нижнем углу), включите другой сервис (кнопка Enable в правом столбце).

10. После завершения настроек закройте модальное окно.

Создание нового сервиса:

- Нажмите кнопку + в правом верхнем углу таблицы.
- В открывшемся списке выберите сервис с типом RemoteStorageService 1.0. Доступна фильтрация списка.
    - Если требуется подключение к БД, например, к Mongo или PostgreSQL, то выбор типа и настройки производятся заказчиком.
- Нажмите кнопку Add.
- Отредактируйте добавленный сервис.

Редактирование сервиса:

- Нажмите кнопку ⚙, расположенную в крайнем правом столбце требуемого сервиса.
- Перейдите на закладку Properties (Рисунок 16).
- В поле «URL» укажите URL, по которому доступен реестр моделей. Если реестр размещен на том же сервере, что и Apache NiFi, то может быть указан localhost.
- В поле Username укажите имя учётной записи реестра моделей.
- В поле Password укажите пароль учётной записи реестра моделей.
- Если поле со свойством не создано, его можно создать, нажав кнопку + в верхнем правом углу таблицы.
- Примените изменения. Нажмите кнопку Apply.

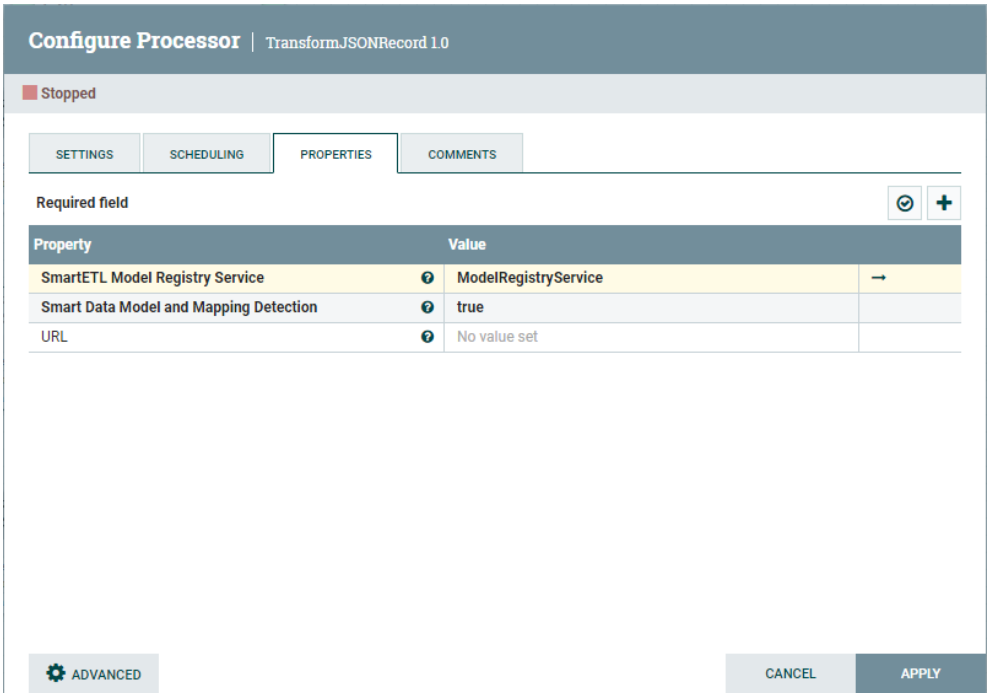

Рисунок 14 – Кастомные свойства блока TransformJSONRecord

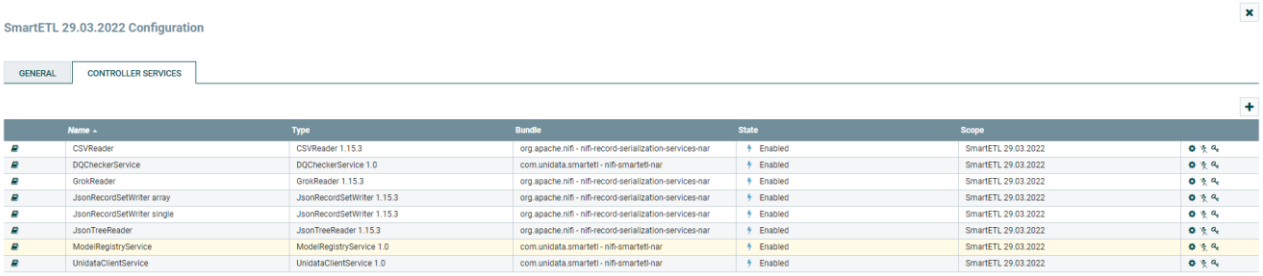

Рисунок 15 – Окно настроек потоков файлов

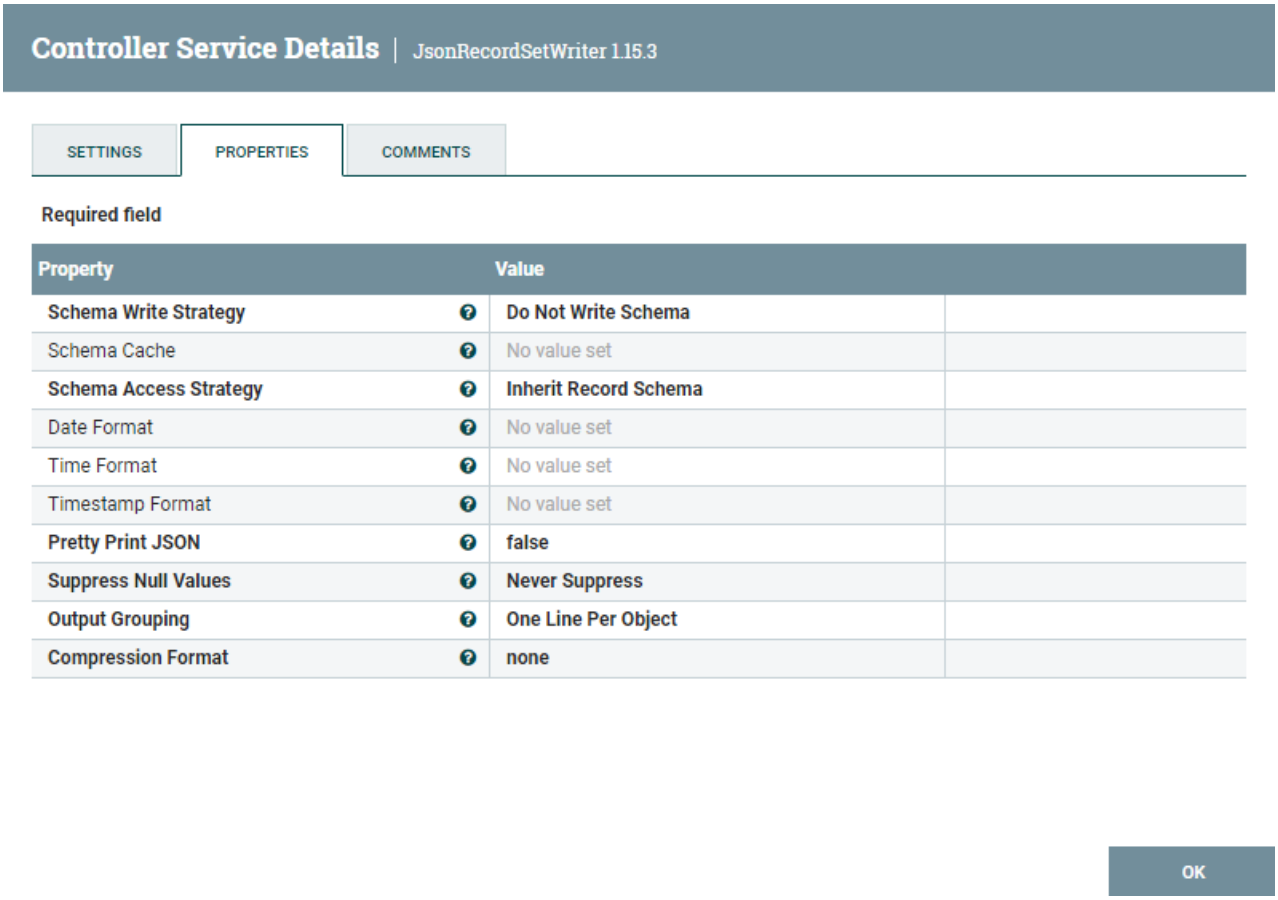

Рисунок 16 – Свойства сервиса, считывающего одиночные записи в формате JSON

3.4 Соединение процессоров

После реализации собственных процессоров загрузки и выгрузки необходимо перестроить схему потоков данных, использовав новые процессоры.

Схема с новыми процессорами может выглядеть следующим образом (Рисунок 17).

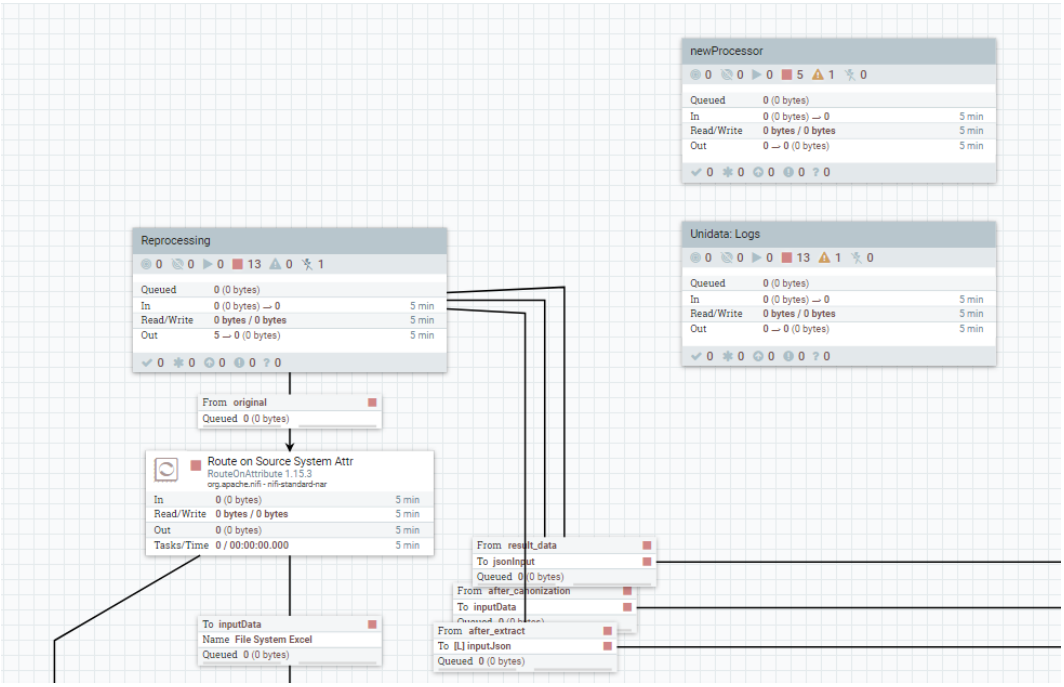

Рисунок 17 – Пример схемы расположения процессоров, включая новый пользовательский процесс

Для **перестройки связей** выполните действия:

1. Наведите курсор на группу процессора, от которой необходимо протянуть связь.
2. Зажмите ЛКМ и протяните курсор до группы процессора, к которой необходимо протянуть связь.
3. Отпустите ЛКМ.
4. В результате действия отобразится модальное окно со свойствами связи. Можно изменить входящие / исходящие порты и добавить настройки.
5. Примените изменения. Нажмите кнопку Add.
6. В результате действия на схеме отобразится новая связь. Также на линии связи будет отображаться блок виртуальной очереди, которая отображает движение данных между процессорами.
7. При необходимости повторите действия 1-6 для остальных групп процессоров.

Для **удаления связей** выполните действия:

1. Предварительно очистите очередь данных, выключите входящий и исходящий процессоры.
2. Наведите курсор на блок виртуальной очереди между теми группами процессоров, у которых необходимо удалить связь.

3. Кликните ПКМ.

4. В контекстном меню выберите пункт Delete. Связь будет удалена без подтверждения действия.

После перестройки связей поток данных может использоваться по назначению. Неиспользуемые процессоры на схеме, при необходимости, могут быть удалены. Также может быть создана резервная копия настроек процессоров при помощи шаблонов. Подробнее см. 3.2.

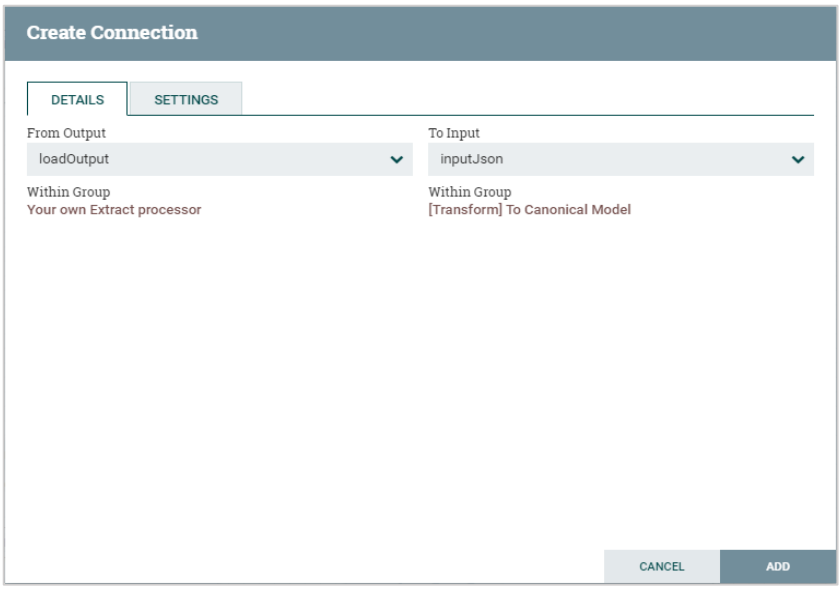

Рисунок 18 – Свойства связи

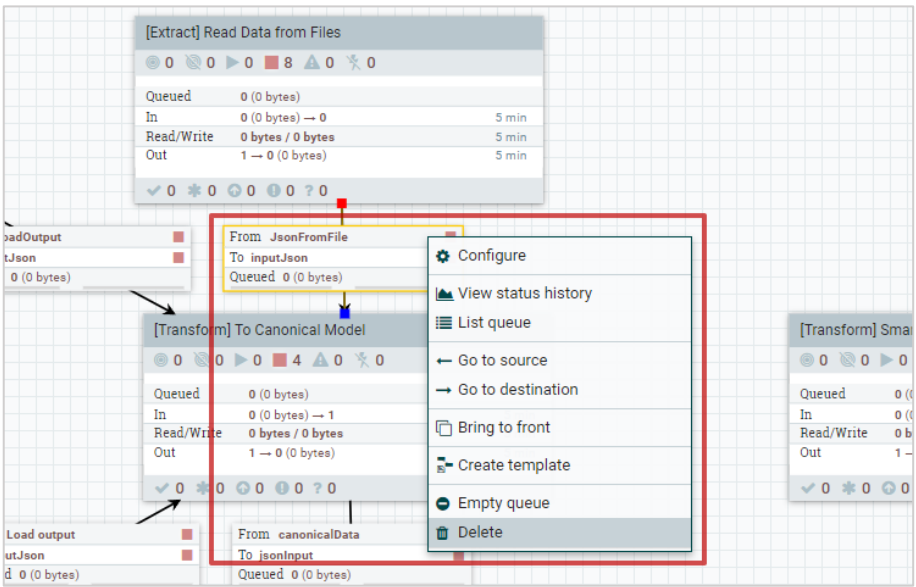

Рисунок 19 – Удаление связи

Пример схемы с перенастроенными связями (Рисунок 20).

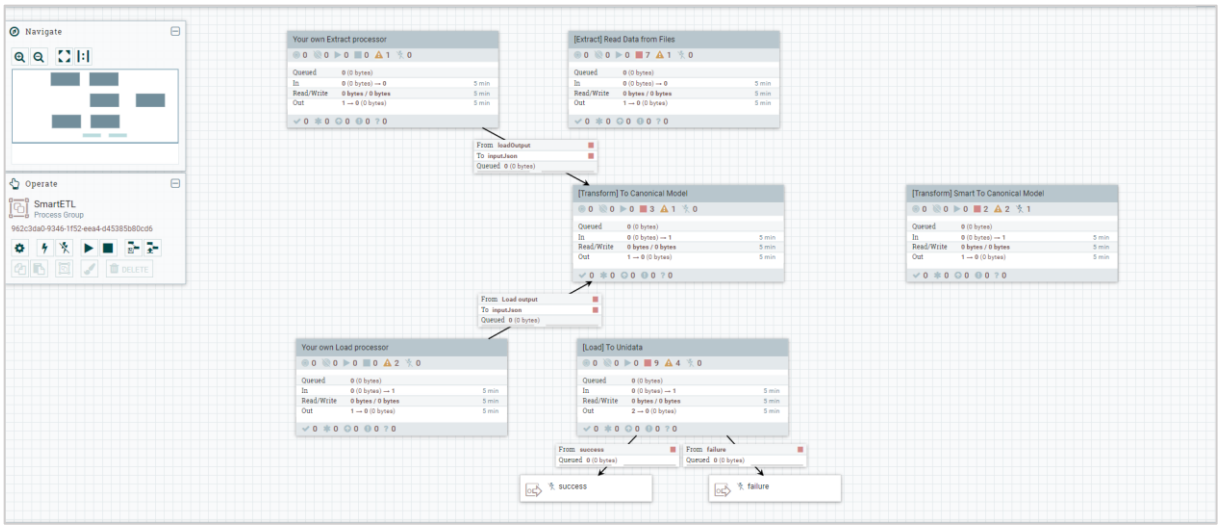

Рисунок 20 – Схема процессоров с перестроенными связями

3.5 Запуск потока данных

Перед запуском потока данных необходимо убедиться, что все предыдущие потоки данных остановлены и очереди данных очищены:

- Войдите в группу SmartETL, если это не сделано ранее.
- Нажмите ПКМ в произвольном месте рабочей области.
- В контекстном меню выберите пункт Stop.
- Повторно нажмите ПКМ в произвольном месте рабочей области.
- В контекстном меню выберите пункт Empty all queues.
- Подтвердите действие.
- Дождитесь очистки очередей.
- Нажмите Ok в окне отчета об удалении очередей.

Чтобы **запустить поток данных**:

- Войдите в группу SmartETL, если это не сделано ранее.
- Нажмите ПКМ в произвольном месте рабочей области.
- В контекстном меню выберите пункт Start.
- В результате действия будут запущены все группы и процессоры, которые содержатся внутри группы SmartETL.

Чтобы **запустить отдельный процессор**:

- Войдите в требуемую группу, если это не сделано ранее.
- Наведите курсор на процессор.
- Нажмите ПКМ на блоке процессора.
- В контекстном меню выберите пункт Start.

Ошибки работы потоков данных. В случае, если в процессе выполнения будет обнаружена ошибка, то в правом верхнем углу соответствующего блока (группы или процессора) отобразится значок ошибки. При наведении на значок можно ознакомиться с ошибкой (Рисунок 21).

Ошибка в блоке группы может отображать сразу несколько ошибок процессоров внутри. Для изучения проблемы необходимо войти внутрь группы и изучить все процессоры, на которых отображена ошибка.

Отдельные блоки могут выполнять функцию логирования, и содержать подробную информацию об ошибках процессоров.

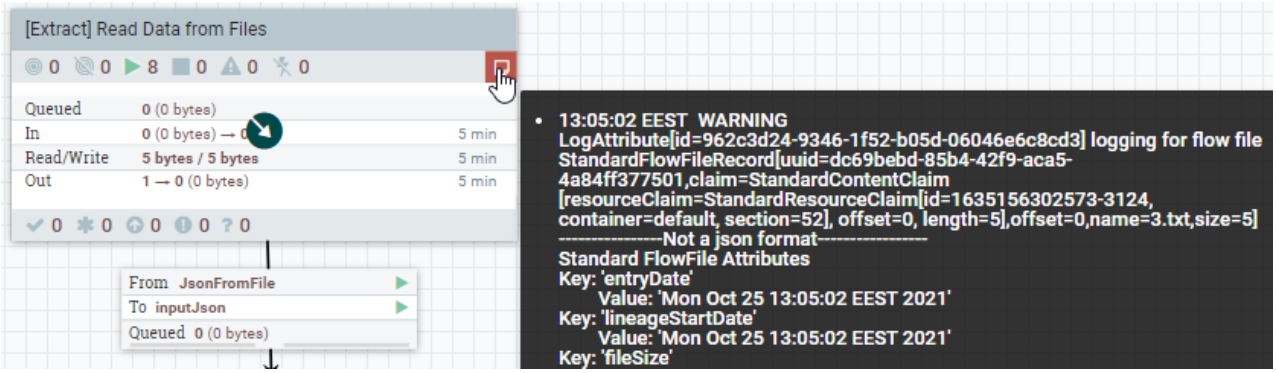

Рисунок 21 – Ошибки выполнения в группе процессоров

3.6 Просмотр файлов, переданных к другому процессору (группе)

Для каждой связи между процессорами (или группами процессоров) предусмотрена виртуальная очередь, отображаемая в виде блока (Рисунок 22). Блок содержит информацию:

- Количество данных, которые были переданы от одного процессора, и ещё не обработаны другим процессором. То есть количество данных в очереди на этот момент.
- Имена входящих и исходящих портов.

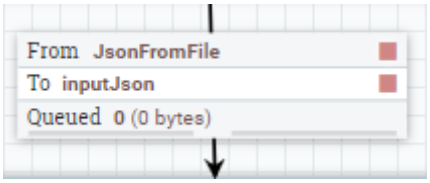

Рисунок 22 – Связь и блок виртуальной очереди

Для просмотра перечня файлов, переданных через связь:

- Наведите курсор на блок очереди.
- Нажмите ПКМ на блоке.
- В контекстном меню выберите пункт List queue.
- В результате действия отобразится модальное окно со списком переданных файлов.
    - Доступен просмотр содержимого файлов и скачивание файлов.

4 Процессоры для Модуля трансформации данных

4.1 Общие сведения

Раздел содержит описания процессоров, используемых в Apache NiFi для различных задач SmartETL.

Процессоры необходимы для работы большинства модулей системы, включая модули реестра моделей, трансформации данных и работы с версиями данных.

Настройка процессоров требуется при развертывании Юниверс SmartETL, либо при изменении потоков трансформации: например, при добавлении нового сценария или новой системы-источника.

4.2 Процессор Universe Connector

4.2.1 Описание

UniverseConnector используется для взаимодействия с Юниверс (продукты DQ и MDM). Процессор осуществляет логин в Юниверс и сохраняет записи из FlowFile в Юниверс через кастомный Rest API.

Процессор может сохранять в реестры Юниверс MDM массивы одиночных записей, связи между записями, и все связанные записи (т.е. записи, указанные в поле relations).

Принцип работы процессора

- При попытке включения экземпляра UniverseClientService производится попытка входа в Юниверс по введенным имени пользователя и паролю.
- В случае успешного входа статус экземпляра UniverseClientService становится Enabled и сервис можно использовать в UniverseConnector процессорах.
- Если введена неверная комбинация логин и пароля, тогда статус UniverseClientService будет Enabling и данный экземпляр будет пытаться залогиниться в Юниверс с некоторой периодичностью.
- Детали ошибки подключения можно будет увидеть в уведомлениях NiFi.
- Процессор UniverseConnector может быть включен, только если в атрибуте UniverseClientService указан активный экземпляр.
- На вход ожидается FlowFile с content, содержащим массив JSON-записей, либо одну JSON-запись.

- Если на вход в процессор поступил FlowFile с content, содержащим **одну** JSON-запись. В случае успешного сохранения записи в Юниверс: по всем связям типа success из процессора выйдет FlowFile с content, содержащим эту запись. В случае ошибки сохранения записи или ошибки логина: по связям типа failure выйдет FlowFile с исходным content, а в атрибутах будет указан код статуса неудачного запроса с сообщением об ошибке.
- Если на вход в процессор поступил FlowFile с content, содержащим **массив** JSON-записей. Все успешно сохраненные в Юниверс записи выйдут из процессора по success связям внутри content одного FlowFile. По failure связям выйдут по одному FlowFile на каждую JSON-запись, которую не удалось сохранить в Юниверс, и в атрибутах каждого FlowFile будет указаны status code и сообщение об ошибке неудачного запроса, либо ошибки DQ. Например, если на вход поступил FlowFile с массивом из 10 JSON-записей и UniverseClientService успешно сохранил 8 из них, а сохранение 2 оставшихся завершилось с ошибкой, то по success связям выйдет один FlowFile, содержащий массив из 8 записей, а по failure связям выйдет два FlowFile, в каждом из которых будет по одной ошибочной JSON-записи.

Процессор содержит обязательный атрибут Universe Client Service, который содержит **параметры**:

- URL (Строковый) – Юниверс URL (например, http://localhost:8080).
- Username (Строковый) – Имя пользователя Юниверс.
- Password (Строковый) – Пароль пользователя Юниверс.

Пример одиночной JSON-записи

```
{
    "entityName": "Person",
    "attributes": {
        "name": "John",
        "surname": "Smith",
        "age": 26,
        "passport_code": "123456789"
    }
}
```

Пример JSON-записи со связями

```
{
    "entityName": "citizen",
```

```
    "attributes": {
        "name": "Иван",
        "surname": "Иванов",
        "date_of_birth": "25.01.1990"
    },
    "relations": {
        "citizen_address": [{
                "record": {
                    "attributes": {
                        "flat": 21,
                        "address_text": "Тестовый адрес"
                    }
                }
            }, {
                "record": {
                    "attributes": {
                        "flat": 20,
                        "address_text": "Тестовый адрес"
                    }
                }
            }
        ]
    }
}
```

Пример массива JSON-записей

```
[{
        "entityName": "Person",
        "attributes": {
            "name": "John",
            "surname": "Smith",
            "age": 26,
            "passport_code": "123456789"
        }
    }, {
        "entityName": "User",
        "attributes": {
            "first_name": "Max",
            "last_name": "Smith",
            "age": 27,
            "cars": [{
                    "car_name": "car_name_1"
                }, {
                    "car_name": "car_name_2"
                }
            ]
        },
        "relations": {
```

```
            "user_address": [{
                    "record": {
                        "attributes": {
                            "flat": 21,
                            "address_text": "Тестовый адрес 1"
                        }
                    }
                }
            ]
        }
    }
]
```

4.2.2 Установка процессора

Чтобы использовать процессор в схеме трансформации, его необходимо добавить в Apache NiFi и настроить. Выполните действия:

1. Откройте Apache NiFi в браузере, если это не сделано ранее. Адрес строится следующим образом:

```
https://SERVER_HOSTNAME:8443/nifi/
```

2. Наведите курсор на иконку добавления процессоров (Рисунок 23), расположенную в левом верхнем углу экрана.

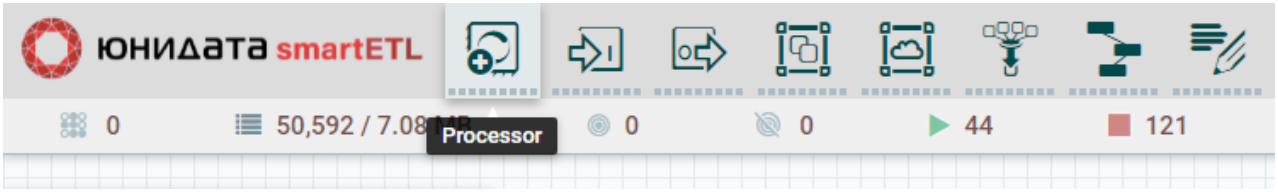

Рисунок 23 – Иконка добавления процессоров

3. Зажмите ЛКМ и перетащите иконку на рабочую область.
4. В результате действия откроется список добавления процессоров.
5. Выберите в списке UniverseConnector (можно воспользоваться поиском).
6. Нажмите кнопку Add.
7. В результате будет добавлен новый блок процессора на рабочей области.
8. Откройте настройки блока UniverseConnector. Для этого наведите курсор на блок, нажмите ПКМ и выберите Configure.
9. Перейдите во вкладку Properties.
10. Нажмите кнопку →, расположенную в крайней правой колонке свойства Universe Client Service.

- Свойство должно иметь значение UniverseClientService. Если строки с сервисом нет, то необходимо создать новое свойство и в Value выбрать сервис UniverseClientService.

11. В результате действия откроется окно настроек потоков файлов (Рисунок 15).

12. Создайте новый и/или отредактируйте требуемый сервис. Нажмите кнопку ⚙, расположенную в крайнем правом столбце.

13. В открывшемся окне перейдите на вкладку Properties.

14. Заполните параметры подключения для процессора.

- URL (Строковый) – Юниверс URL (например, http://localhost:8080).
- Username (Строковый) – Имя пользователя Юниверс.
- Password (Строковый) – Пароль пользователя Юниверс.

15. Если подключение выполнено верно, то в списке сервисов в столбце State требуемый сервис будет иметь статус Enabled.

- Для смены сервиса выключите текущий сервис (кнопка Disable в правом столбце), обновите список (кнопка Refresh в левом нижнем углу), включите другой сервис (кнопка Enable в правом столбце).

16. После завершения настроек закройте модальное окно.

4.3 Процессор Universe DQ

4.3.1 Описание

UniverseDQProcessor используется для применения правил качества данных Юниверс к записям, приведённым в канонический вид (результат работы процессора To Canonical Model). Используются правила обогащения и валидации.

Принцип работы процессора

- Принимает на вход json-схему в канонической форме.
- Отправляет json-схему на проверку в DQCheckerService.
- DQCheckerService логинится в Юниверс DQ или Юниверс MDM (через UniverseClientService) и получает токен.
- Отправляет soap-запрос в Юниверс для проверки и обогащения записей.
- Получает ответ в виде soap:

- Если в ответе не содержится ошибок, то отправляет статус success с изменённым или исходным json (если обогащения не произошло).
- Если в ответе ошибки, то отправляет статус error с информацией об ошибках валидации и статус incorrect с исходным файлом.

Исходящий файл при различных статусах

- success – json-схема в каноническом виде, с данными, обогащёнными в Юниверс.
- error – файл, содержащий данные об ошибке проверки или о неудачных валидациях.
- incorrect – json-схема, не прошедшая валидацию (в атрибутах указывается, какие правила не прошли валидацию).

Параметры процессора

Сервис DQCheckerService, для подключения к Юниверс. Содержит в себе:

- UniverseClientService – для авторизации в Юниверс.
- Universe wsdl – адресс wsdl DQ в Юниверс, для валидации и обогащения (формат <schema>://<host>:<port>/Universe-backend/api/public/dq/v5?wsdl).

Пример входящего файла

```
{
    "entityName": "citizen",
    "attributes": {
        "surname": "Иванов",
        "firstname": "Иван",
        "patronymic": "Иванович",
        "date_of_birth": "1957-01-23T00:00:00.000",
        "source_id": "d86c3432-ae16-406e-be86-a57751668423",
        "childrenNames": [
           {
               "name": "Кеша"
           },
           {
               "name": "Гаврила"
           }
       ]
    }
}
```

Пример исходящего файла

```
{
  "entityName" : "citizen",
  "attributes" : {
    "firstname" : "Иван",
    "patronymic" : "Иванович",
    "surname" : "ИВАНОВ",
    "date_of_birth" : "1957-01-23T00:00:00.000",
    "source_id" : "d86c3432-ae16-406e-be86-a57751668423",
    "childrenNames" : [ {
      "name" : "Кеша"
    }, {
      "name" : "Гаврила"
    } ]
  }
}
```

4.3.2 Установка процессора

Чтобы использовать процессор в схеме трансформации, его необходимо добавить в Apache NiFi и настроить. Выполните действия:

1. Откройте Apache NiFi в браузере, если это не сделано ранее. Адрес строится следующим образом:

```
https://SERVER_HOSTNAME:8443/nifi/
```

2. Наведите курсор на иконку добавления процессоров (см. Рисунок 23), расположенную в левом верхнем углу экрана.
3. Зажмите ЛКМ и перетащите иконку на рабочую область.
4. В результате действия откроется список добавления процессоров.
5. Выберите в списке UniverseDQProcessor (можно воспользоваться поиском).
6. Нажмите кнопку Add.
7. В результате будет добавлен новый блок процессора на рабочей области.
8. Откройте настройки блока UniverseDQProcessor. Для этого наведите курсор на блок, нажмите ПКМ и выберите Configure.
9. Перейдите во вкладку Properties.
10. Нажмите кнопку →, расположенную в крайней правой колонке строки Universe data quality checker.

- Свойство должно иметь значение DQCheckerService. Если строки с сервисом нет, то необходимо создать новое свойство и в Value выбрать сервис DQCheckerService.

11. В результате действия откроется окно настроек потоков файлов (Рисунок 15).

12. Создайте новый и/или отредактируйте требуемый сервис. Нажмите кнопку ⚙, расположенную в крайнем правом столбце.

13. В открывшемся окне перейдите на вкладку Properties.

14. Заполните параметры подключения для процессора.

    - UniverseClientService – для авторизации в Юниверс.
    - Universe wsdl – адресс wsdl DQ в Юниверс, для валидации и обогащения (формат <schema>://<host>:<port>/Universe-backend/api/public/dq/v5?wsdl).

15. Если подключение выполнено верно, то в списке сервисов в столбце State требуемый сервис будет иметь статус Enabled.

    - Для смены сервиса выключите текущий сервис (кнопка Disable в правом столбце), обновите список (кнопка Refresh в левом нижнем углу), включите другой сервис (кнопка Enable в правом столбце).

16. После завершения настроек закройте модальное окно.

4.4 Процессор Merge JSON Record

4.4.1 Описание

MergeJSONRecord – это доработанная версия стандартного процессора MergeRecord, ориентированная на работу с JSON. Используется для объединения нескольких FlowFile в один с использованием приоритетов объединения. Процессор создает кластер, где накапливаются FlowFile для объединения. Когда кластер заполняется, то всё содержимое сливается в единый FlowFile.

Внутри кластера может содержаться несколько FlowFile, имеющих минимальные различия. Для определения их реальной схожести используется атрибут, задающий приоритет.

Принцип работы процессора

- Принимает на вход FlowFile в виде json-схем.
- Проверяет входящие файлы.

- Если у файлов совпадает значение атрибута, имя которого указано в поле Correlation Attribute Name, то все такие файлы складываются в один кластер.
- После заполнения кластера все FlowFile объединяются в один. Файлы объединяются с учётом приоритета, выставленного в поле Rate Attribute Name. Правила объединения:
    - Самым приоритетным (основным) будет FlowFile с наибольшим значением атрибута Rate Attribute Name, а остальные будут дополняющими.
    - Если у основного и дополняющего JSON по одному и тому же пути заполнено значение простого типа или JSON объект: сохраняется значение основного JSON.
    - Если у основного и дополняющего JSON по одному и тому же пути JSON заполнено значение типа массив, тогда массив у основного JSON дополняется недостающими элементами массива из дополняющего.
    - Если у дополняющего JSON заполнено значение по JSON пути, а у основного нет, тогда недостающее значение добавляется к основному JSON. Верно и для случая, когда недостающим значением является JSON объект.
    - Если у двух дополняющих JSON заполнено значение по JSON пути, а у основного нет, тогда в основной попадет значение более приоритетного дополняющего JSON, менее приоритетные проигнорируются.

Алгоритм кластеризации

Алгоритм работает идентично алгоритму Bin-Packing Algorithm в стандартном процессоре MergeRecord. Кластер объявляется заполненным в случаях:

- Когда количество FlowFile в кластере достигло значения параметра Maximum Number of Records.
- Когда кластером достигнут максимальный срок жизни, который может быть указан в параметре Max Bin Age.
- Когда достигнут максимальный объем кластера, который может быть указан в параметре Maximum Bin Size.

Также есть исключение, при котором кластер также объявляется заполненным. Пусть в параметре Maximum Number of Bins указано, например, значение по умолчанию 10, в параметре Minimum Number of Records указано значение 2 и в очереди перед процессором накопилось ровно 10 FlowFile с разными значениями коррелирующего атрибута, то есть все они образуют по одному кластеру в памяти процессора. Как только в очередь попадает еще

один FlowFile с неповторяющимся значением коррелирующего атрибута, кластер, в котором находится первый в очереди FlowFile, объявляется заполненным. В память процессора попадает кластер, образованный только что прибывшим 11-ым (теперь 10-м в очереди) FlowFile.

Параметры процессора

Имя	**Тип**	**Обязательный**	**Описание**
Correlation Attribute Name	Строковый	Да	Имя атрибута, по совпадению значения которого будут объединяться FlowFile
Rate Attribute Name	Строковый	Да	Имя атрибута рейтинга FlowFile, по значению которого будет найден главный FlowFile, в который будут объединены остальные FlowFile собранного кластера
Attribute Strategy	Выпадающий список из двух опций: Keep All Unique Attributes Keep Only Common Attributes	Да	Стратегия объединения атрибутов FlowFile
Minimum Number of Records	Целое положительное число	Да	Минимально возможное число FlowFile в кластере
Maximum Number of Records	Целое положительное число	Да	Максимально возможное число FlowFile в кластере
Minimum Bin Size	Целое положительное число	Да	Минимальный размер кластера FlowFile
Maximum Bin Size	Целое положительное число	Нет	Максимальный размер кластера FlowFile
Max Bin Age	Значение должно удовлетворять маске <duration> <time unit>, где <duration> - целое	Нет	Максимальное время жизни кластера FlowFile

	положительное число, <time unit> - seconds, minutes или hours		
Maximum Number of Bins	Целое положительное число	Да	Максимальное количество одновременно собираемых кластеров FlowFile

Типы исходящих связей

- Merged – В связи этого типа отправляются успешно объединенные FlowFile.
- Failure – В случае ошибки объединения или парсинга JSON контента из кластера в связи этого типа отправляются исходные FlowFile кластера.
- Original – В связи этого типа отправляются исходные FlowFile заполненного кластера, независимо от результата объединения.

Пример объединения JSON

Пусть в кластер попали следующие FlowFile с JSON контентом, представленным в порядке приоритета:

FlowFile с Rate = 3

```
{
    "name": "John",
    "surname": "Smith",
    "cars": [
        {
            "carName": "Audi"
        }]
}
```

FlowFile с Rate = 2

```
{
    "name": "Джон",
    "age": 21,
    "contacts": {
        "e-mail": "example@gmail.com",
        "phone": "123-12-45"
    }
}
```

FlowFile с Rate = 1

```
{
```

```
    "contacts": {
        "phone": "89111234567",
        "country": "Russia"
    },
    "cars": [{
            "carName": "Audi"
        }, {
            "carName": "BMW"
        }
    ]
}
```

То есть FlowFile с Rate = 3 является основным, а остальные – дополняющими. Результатом объединения будет FlowFile с JSON контентом:

```
{
    "name": "John",
    "surname": "Smith",
    "age": 21,
    "contacts": {
        "e-mail": "example@gmail.com",
        "phone": "123-12-45",
        "country": "Russia"
    },
    "cars": [{
            "carName": "Audi"
        }, {
            "carName": "BMW"
        }
    ]
}
```

4.4.2 Установка процессора

Чтобы использовать процессор в схеме трансформации, его необходимо добавить в Apache NiFi и настроить. Выполните действия:

1. Откройте Apache NiFi в браузере, если это не сделано ранее. Адрес строится следующим образом:

```
https://SERVER_HOSTNAME:8443/nifi/
```

2. Наведите курсор на иконку добавления процессоров (см. Рисунок 23), расположенную в левом верхнем углу экрана.
3. Зажмите ЛКМ и перетащите иконку на рабочую область.
4. В результате действия откроется список добавления процессоров.
5. Выберите в списке MergeJSONRecord (можно воспользоваться поиском).

6. Нажмите кнопку Add.
7. В результате будет добавлен новый блок процессора на рабочей области.
8. Откройте настройки блока MergeJSONRecord. Для этого наведите курсор на блок, нажмите ПКМ и выберите Configure.
9. Перейдите во вкладку Properties.
10. Добавьте параметры процессора и укажите их значения. Список параметров см. выше в описании.
11. После завершения настроек нажмите кнопку Apply.

4.5 Процессор Save In Data Manager

4.5.1 Описание

SaveInDataManager используется для передачи и сохранения FlowFile из Модуля трансформации в Модуль работы с версиями данных.

Для работы процессора требуется минимум одна входящая связь к данным.

Принцип работы процессора

- Для сохранения файла передаётся json-схема и сам FlowFile.
- Для сохранения FlowFile есть 2 метода. Каждый метод используется для определенных случаев.
- При первом сохранении файла должен быть вызван метод Создания batch (первичная загрузка файла), в который обязательно должен быть передан batchId, еще не существующий в системе. В случае, если batchId не передан, или передан идентификатор, для которого уже создан batch, то будет выдана ошибка и файл не сохранится. В случае успеха создается batch и запись с сохраненным FlowFile.
- При последующих сохранениях файлов в том же batch должен вызываться метод Загрузка файла, и в него должен передаваться batchId, который уже заведен в системе. В противном будет выдана ошибка.
- Методы возвращают информацию о сохраненном в системе файле Record. По идентификатору Record можно запрашивать эту информацию с помощью метода Получить метаинформацию о файле. Также можно скачать сам файл с помощью метода Скачивание файла.

- Так же у записи и у batch есть параметр Status, который можно изменять с помощью методов Изменение статуса батча и Изменение статуса записи.
- Доступна возможность репроцессинга данных.

Параметры процессора

- Data Manager URL (строковый) – URL Модуля работы с версиями данных. Указывается значение ${dataManagerURL}. Значение переменной указывается в переменных для группы, к которой принадлежит процессор (перейдите на уровень выше, вызовите свойства группы и выберите пункт Variables).
- Create batch (логический) – Если значение параметра true, то при отправке запроса на сохранение файла, Модуля работы с версиями данных создаст batch с указанным в запросе batchId. Если значение false, то модуль будет искать уже существующий batch с указанным batchId и, если такой не будет найден, сохранение завершится неудачей.
- Id (строковый) – objectId сохраняемого файла. Указывается значение ${uuid}. Значение переменной заполняется из json-схемы, которая передается вместе с FlowFile.
- runId (строковый) – Идентификатор запуска. Указывается значение ${runId}. Значение переменной заполняется из json-схемы, которая передается вместе с FlowFile.
- BatchId (строковый) – Идентификатор batch. Указывается значение ${batchId}. Значение переменной заполняется из json-схемы, которая передается вместе с FlowFile.
- State (строковый) – Статус. Указывается значение ${state}. Значение переменной заполняется из json-схемы, которая передается вместе с FlowFile.
- Source system (строковый) – Система-источник. Значение переменной заполняется из json-схемы, которая передается вместе с FlowFile.
- Generate Download Link (логический) – Если значение параметра true, то в FlowFile, в атрибут smartetl.data.manager.<State>.link, который проходит через через связи типа original, будет записана ссылка на скачивание сохраненного файла. Ссылка доступна столько дней, сколько указано в параметре <Download Link Age>.
- Download Link Age (целочисленный) – Если значение параметра <Generate Download Link> равно true, тогда доступно заполнение параметра. Значение параметра допустимо в диапазоне от 1 до 7.

Все параметры процессора типа String поддерживают Expression Language. Типы исходящих связей описаны ниже:

- Response. В случае успешного сохранения файла во все связи данного типа будет отправлен FlowFile, в content которого записан ответ от Модуля работы с версиями данных.
- Original. В случае успешного сохранения файла во все связи данного типа будет отправлен оригинальный FlowFile.
- Failure. В случае возникновения какой-либо ошибки в процессе сохранения файла в МРВД или если какой-либо из параметров пуст после применения Expression Language, тогда во все связи данного типа будет отправлен исходный FlowFile

4.5.2 Установка процессора

Чтобы использовать процессор в схеме трансформации, его необходимо добавить в Apache NiFi и настроить. Выполните действия:

1. Откройте Apache NiFi в браузере, если это не сделано ранее. Адрес строится следующим образом:

```
https://SERVER_HOSTNAME:8443/nifi/
```

2. Наведите курсор на иконку добавления процессоров (Рисунок 23), расположенную в левом верхнем углу экрана.
3. Зажмите ЛКМ и перетащите иконку на рабочую область.
4. В результате действия откроется список добавления процессоров.
5. Выберите в списке SaveInDataManager (можно воспользоваться поиском).
6. Нажмите кнопку Add.
7. В результате будет добавлен новый блок процессора на рабочей области.
8. Откройте настройки блока SaveInDataManager. Для этого наведите курсор на блок, нажмите ПКМ и выберите Configure.
9. Перейдите во вкладку Properties.
10. Добавьте параметры процессора и укажите их значения. Список параметров см. выше в описании.
11. После завершения настроек нажмите кнопку Apply.

4.6 Процессор Schema Generator

4.6.1 Описание

SchemaGenerator принимает на вход FlowFile с content, который представляет собой либо массив JSON-записей, либо единственную JSON-запись.

Принцип работы процессора

- От процессора к другому блоку можно построить два типа связей: success и failure.
- Если генерация успешна, и JSON-схемы сохранены в Модуль реестра данных, то во все связи типа success поступит FlowFile, у которого content будет содержать сохраненную JSON схему.
- Если генерация прошла с ошибкой, или возникла проблема с сохранением, то во все связи типа failure поступит исходный FlowFile, в атрибутах которого будут указаны status code (json.schema.generation.error.status.code) и сообщение об ошибке (json.schema.generation.error.message). В случае ошибки на стороне Модуля реестра данных, то сообщения поступают из модуля. В случае ошибки на стороне процессора, то ошибка попадет в logger, либо также попадет в атрибуты (json.schema.generation.error.message).

Параметры процессора

- JSON schema generator service: Кастомный Contoller Service – RemoteStorageService, в котором заложена вся логика взаимодействия с Модулем реестра моделей. Необходимо создать новый, или выбрать один из уже созданных экземпляров Controller Service. Внутри параметра указываются свойства: URL, Username и Password, необходимые для входа в Модуль реестра моделей.
- JSON schema name – Имя генерируемой JSON схемы (не должно совпадать с уже созданными моделями данных в Модуле реестра моделей). Поддерживает Expression Language.

4.6.2 Установка процессора

Чтобы использовать процессор в схеме трансформации, его необходимо добавить в Apache NiFi и настроить. Выполните действия:

1. Откройте Apache NiFi в браузере, если это не сделано ранее. Адрес строится следующим образом:

```
https://SERVER_HOSTNAME:8443/nifi/
```

2. Наведите курсор на иконку добавления процессоров (Рисунок 23), расположенную в левом верхнем углу экрана.
3. Зажмите ЛКМ и перетащите иконку на рабочую область.
4. В результате действия откроется список добавления процессоров.
5. Выберите в списке SchemaGenerator (можно воспользоваться поиском).
6. Нажмите кнопку Add.
7. В результате будет добавлен новый блок процессора на рабочей области.
8. Откройте настройки блока SchemaGenerator. Для этого наведите курсор на блок, нажмите ПКМ и выберите Configure.
9. Перейдите во вкладку Properties.
10. Добавьте параметры процессора и укажите их значения. Список параметров см. выше в описании.
11. После завершения настроек нажмите кнопку Apply.

5 Модуль работы с версиями данных

Модуль хранит все сохраненные версии FlowFile с данными. Для того, чтобы данные поступали в Модуль работы с версиями данных, необходимо, чтобы в поток трансформации был встроен процессор сохранения (SaveInDataManager, см. пп. 4.5).

Перед первым запуском требуется настройка Модуля и процессора SaveInDataManager. Далее Модуль работает самостоятельно, являясь частью процесса трансформации данных.

При необходимости, можно взаимодействовать с Модулем через API.

5.1 Описание методов сохранения данных

5.1.1 NewRecordRequest

Предназначен для сохранения метаинформации при создании новой записи.

Описание

```
data class NewRecordRequest(
    val runId: String?,
    val batchId: String?,
    val objectId: String?,
    val sourceSystem: String?,
    val createdBy: String?,
    val updatedBy: String?,
    val state: String?,
    val customInfo: Map<String, *>? = mapOf<String, Any>()
)
```

Пример json

```
{"runId": "test",
"batchId": "6e9cd572-3917-4e8b-bd40-d0e9214c4e46",
"objectId": "test",
"sourceSystem": "test",
"createdBy": null,
"updatedBy": null,
"state": "primordial",
"customInfo": {}
}
```

5.1.2 Record

Запись и данные о ней, которые хранятся в Модуле работы с версиями данных.

Описание

```
data class Record(
    val id: String,
    val runId: String,
    val objectId: String?,
    val sourceSystem: String,
    val createdBy: String?,
    @JsonFormat(pattern = "dd-MM-yyyy HH:mm:ss")
    val createTime: LocalDateTime,
    val updatedBy: String? = createdBy,
    @JsonFormat(pattern = "dd-MM-yyyy HH:mm:ss")
    val updateTime: LocalDateTime = createTime,
    val contentType: String?,
    val contentLength: Long,
    val checksum: String,
    val fileName: String?,
    val reprocessing: Boolean,
    val state: String,
    val batch: Batch,
    val storageLink: StorageLink,
    val status: Status
): Serializable
```

Пример json

```
{
  "id": "cc7ccc3a-8f55-41af-93f2-0925b07442db",
  "runId": "test",
  "objectId": "test",
  "sourceSystem": "test",
  "createdBy": null,
  "createTime": "22-03-2022 17:46:08",
  "updatedBy": null,
  "updateTime": "22-03-2022 17:46:08",
  "contentType": "application/octet-stream",
  "contentLength": 20,
  "checksum": "lu9wvvsxSU93crl5SUo2dA==",
  "fileName": "test_file.tst",
  "reprocessing": false,
  "state": "primordial",
  "batch": {
    "id": "a859e4fe-6160-4da1-85cc-85b8a0670b4b",
    "sourceFile": {
      "id": "2022-03-22/20112be1-9d50-450d-b8a7-4affefe1ee74",
      "bucket": "mrvd",
      "storageKey": "s3"
    },
    "status": {
      "code": "NEW",
      "title": "Новое"
```

```
    }
  },
  "storageLink": {
    "id": "2022-03-22/20112be1-9d50-450d-b8a7-4affefe1ee74",
    "bucket": "mrvd",
    "storageKey": "s3"
  },
  "status": {
    "code": "NEW",
    "title": "Новое"
  }
}
```

5.1.3 StorageLink (S3)

Файл, который хранится в Minio.

Описание

```
data class StorageLink(
    val id: String,
    val bucket: String,
    val storageKey: String,
): Serializable
```

Пример json

```
{
    "id": "2022-03-22/20112be1-9d50-450d-b8a7-4affefe1ee74",
    "bucket": "mrvd",
    "storageKey": "s3"
}
```

5.1.4 Batch

Описание Batch

Описание

```
data class Batch(
    val id: String,
    val sourceFile: StorageLink,
    val status: Status
): Serializable
```

Пример json

```
{
    "id": "a859e4fe-6160-4da1-85cc-85b8a0670b4b",
    "sourceFile": {
      "id": "2022-03-22/20112be1-9d50-450d-b8a7-4affefe1ee74",
      "bucket": "mrvd",
      "storageKey": "s3"
    },
    "status": {
      "code": "NEW",
      "title": "Новое"
    }
}
```

5.1.5 Status

Возможные статусы сохранения данных.

Описание

```
data class Status(val code: String, val title: String): Serializable  {
    companion object {
        val NEW_STATUS = Status("NEW", "Новое")
        val COMPLETED_STATUS = Status("COMPLETED", "Обработано")
        val ERROR_STATUS = Status("ERROR", "Ошибка")
        val TERMINATE_STATUS = Status("TERMINATE", "Прервано")

        val availableStatuses = listOf(NEW_STATUS, COMPLETED_STATUS,
ERROR_STATUS, TERMINATE_STATUS)
    }
}
```

Пример json

```
{
      "code": "NEW",
      "title": "Новое"
}
```

5.1.6 ResponseError

Ошибка на выходе REST-сервиса.

Описание

```
class ResponseError constructor(
    val errorCode: Int,
```

```
    val errorGuid: String,
    @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")
    val errorDateTime: LocalDateTime,
    val errorText: String,
    val stackTrace: String
)
```

Пример json

```
{
  "errorCode": 101,
  "errorGuid": "7fd747b9-e392-4f38-bd39-f7323a188646",
  "errorDateTime": "2022-03-22 17:49:51",
  "errorText": "Batch with id = f8b1e88f-b901-49b1-99ca-3abd2f2dc209 not
found",
  "stackTrace":
"com.Universe.smartetl.datamanager.controller.error.NotFoundException:
..."
}
```

5.2 API модуля

5.2.1 Создание batch (первичная загрузка)

Запрос

```
POST /upload/batch
Content-type = multipart/form-data
```

Таблица 1 – Параметры запроса Создания batch

Параметр	*Обязательный*	*Комментарий*
data	+	Метаданные в формате json newRecordRequest (передаётся строкой)
file	+	FlowFile
generatePresignedUrl	-	Генерировать ли ссылку на файл (true/false)
presignedUrlExpirationInDays	-	Сколько времени будет доступна ссылка (дни)

Ответ

Таблица 2 – Параметры ответа Создания batch

Параметр	*Тип данных*
batch*	Batch
checksum*	string
contentLength*	integer($int64)

contentType	string
createTime*	string($date-time)
createdBy	string
fileName	string
id*	string
objectId	string
reprocessing*	boolean
runId*	string
sourceSystem*	string
status*	Status
storageLink*	SourceFile
updateTime*	string($date-time)
updatedBy	string

5.2.2 Загрузка файла

Запрос

```
POST /upload
Content-type = multipart/form-data
```

Таблица 3 – Параметры запроса Загрузки файла

Параметр	*Обязательный*	*Комментарий*
data	+	Метаданные в формате json newRecordRequest (передаётся строкой)
file	+	FlowFile
generatePresignedUrl	-	Генерировать ли ссылку на файл (true/false)
presignedUrlExpirationInDays	-	Сколько времени будет доступна ссылка (дни)

Ответ

Таблица 4 – Параметры ответа Загрузки файла

Параметр	*Тип данных*
batch*	Batch
checksum*	string
contentLength*	integer($int64)
contentType	string
createTime*	string($date-time)
createdBy	string
fileName	string
id*	string
objectId	string
reprocessing*	boolean
runId*	string
sourceSystem*	string

status*	Status
storageLink*	SourceFile
updateTime*	string($date-time)
updatedBy	string

5.2.3 Скачивание файла

Запрос

```
GET /download/record/{id}
```

где id – идентификатор записи, который возвращается в поле id в ответе метода Загрузка файла.

Ответ

В качестве ответа начинается загрузка файла.

5.2.4 Получить метаинформацию о файле

Запрос

```
GET /info/record/{id}
```

где id – идентификатор записи, который возвращается в поле id в ответе метода Загрузка файла.

Ответ

Таблица 5 – Параметры ответа Загрузки файла

Параметр	*Тип данных*
batch*	Batch
checksum*	string
contentLength*	integer($int64)
contentType	string
createTime*	string($date-time)
createdBy	string
fileName	string
id*	string
objectId	string
reprocessing*	boolean
runId*	string
sourceSystem*	string
status*	Status
storageLink*	SourceFile

updateTime*	string($date-time)
updatedBy	string

5.2.5 Изменение статуса batch

Запрос

```
PUT /change/status/batch/{id}/{newStatus}
```

Параметры:

id – идентификатор batch, который возвращается в поле batch.id ответа метода Загрузка файла.

newStatus – новый статус. Варианты: NEW, COMPLETED, ERROR, TERMINATE

Ответ

При успешном изменении в ответ передается статус 200.

5.2.6 Изменение статуса записи

Запрос

```
PUT /change/status/record/{id}/{newStatus}
```

Параметры:

id – идентификатор записи, который возвращается в поле id ответа метода Загрузка файла.

newStatus – новый статус. Варианты: NEW, COMPLETED, ERROR, TERMINATE

Ответ

При успешном изменении в ответ передается статус 200.

5.3 Репроцессинг

Доступна возможность пометить данные для повторной обработки в модуле трансформации данных (Apache NiFi).

Порядок действий:

- В БД, в таблице record проставляется флаг reprocessing.

- Далее периодически запускается задача, которая находит все записи для репроцессинга, и складывает их в очередь rabbitMQ. После успешной отправки флаг снимается. Периодичность запуска задачи задается в настройках, по умолчанию 60 минут.
- Затем Apache NiFi забирает записи из этой очереди, и повторно обрабатывает (проходит схему трансформации).

Альтернативный способ: принудительный запуск репроцессинга через REST-запрос `GET /reprocessing`.

5.4 Конфигурация модуля

Конфигурация расположена в файле `/service/src/main/resources/application.yaml`. Изменение параметров возможно только перед сборкой jar-файла модуля.

Перечень доступных параметров для конфигурации:

```
datamanager:
  postgres:
    url: "jdbc:postgresql://postgres:5432"
    user: "postgres"
    password: "password"
    database: "datamanager"
  s3:
    accessKey: "AKIAIOSFODNN7EXAMPLE"
    secretKey: "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
    serverHost: "minio"
    serverPort: 9000
  rabbit:
    host: "rabbit"
    port: 5672
    user: "user"
    password: "plvv9FHI93s="
  jobs:
    reprocessing:
      cronExpression: "0 0 * * * ?"
```

Если конфигурирование производится через системные переменные, то все имена переменных необходимо привести к верхнему регистру, а точки заменить на подчеркивание. Например, для того, чтобы поменять настройку пользователя для RabbitMQ, надо задать переменную DATAMANAGER_RABBIT_USER.

6 Устранение неисправностей

Проблемы Apache NiFi

Проблема	*Решение*
Недостаточно прав. Например, недоступно редактирование процессоров	В учетной записи пользователя не добавлена одна или несколько групп, расширяющих права доступа. Необходимо обратиться к администратору системы.
При попытке импорта модели данных из Юниверс MDM возникает ошибка: неверный пароль	Пароль Юниверс MDM изменился, но изменения не были отражены в конфигурации SmartETL. Необходимо перейти в настройки, выбрать нужную систему-источник. В свойствах системы изменить конфигурацию (логин/пароль).
Блок связи между элементами процессора имеет статус no detected schema. Не удается запустить передачу данных	Нет связи с требуемой схемой. Необходимо либо убрать эту ветку потока данных, либо соединить ветку с необходимыми блоками
Блок связи между элементами процессора имеет статус no valid schema. Не удается запустить передачу данных	Указанная схема содержит ошибки. Необходимо исправить схему модели данных
Блок связи между элементами процессора имеет статус failure. Не удается запустить передачу данных	Ошибка в потоке данных. Необходимо убедиться, что соединены верные блоки и корректны настройки и т.д.
У пользовательского процессора не отображается иконка создания связи с другими процессорами (нельзя создать связь)	Убедиться, что в группе extract есть output-порт. Также желательно убедиться, что в группе, куда должна прийти связь, есть input-порт
Поток данных остановился	Проверить состояние виртуальных очередей между блоками внутри процессора. Максимальный объем очереди 1 ед., так как для безопасности данных остановка должна производиться моментально. Если очередь заполнена, то необходимо вручную проверить данные, схему данных и маппинг
Нельзя удалить группы с процессорами	Удаление группы возможно только после удаления шаблона, из которых группа загружена. Рекомендуется перед удалением скачать шаблон (если нет резервной копии)

Проблемы PostgreSQL

Проблема	*Решение*
Низкая скорость выгрузки из БД	Требуется оптимизация ресурсов и конфигурации PostgreSQL. Необходимо проанализировать данные – выяснить средний

	объем, сложность данных, количество связей между данными. Основываясь на результате анализа данных, следует изменить конфигурацию PostgreSQL. При необходимости, выделить больше вычислительной мощности. Описание конфигурации см. в официальной документации PostgreSQL.

Проблемы других компонентов системы

Проблема	*Решение*

Не удается войти в Minio. Система была развернута через Docker	Перезапустите контейнер:
	`docker-compose stop minio`
	`docker-compose up -d`

Список сокращений и условные обозначения

Принятые термины и определения

ETL	аббревиатура Extract, Transform, Load. Вид процессов управления данными, который состоит из этапов: извлечение, трансформация и выгрузка данных.
FlowFile (файл потока)	базовый объект обработки в Apache NiFi. Он включает в себя данные и атрибуты данных, которые используются процессорами NiFi для обработки. Файл потока обычно содержит данные, извлеченные из исходных систем.
Авторизация	предоставление учетной записи прав на выполнение определенных действий в процессе входа в платформу.
Аутентификация	проверка подлинности логина/пароля или идентификатора учетной записи.
БД (база данных)	совокупность данных, хранимых в соответствии со схемой данных.
ЛКМ	левая кнопка мыши
Маппинг	определение соответствия данных между различными семантиками одного объекта. В контексте SmartETL под маппингом следует понимать описание соответствия между двумя моделями данных
ПКМ	Правая кнопка мыши

Условные обозначения

Важное уточнение

Примечание

Фрагменты программного кода, либо содержимого файла:

```
Universe.search.nodes.addresses=localhost:9300
Universe.search.cluster.name=elasticsearch-cluster
Universe.search.index.prefix = default
```