# Конфликты

## Что такое конфликты?

**Конфликты** - проблемы, которые возникли при импорте структуры. Они могут (и должны) быть решены пользователем.

Система не применяет изменения (4-ый этап), если есть хотя бы один активный конфликт.

Конфликт считается активным, если он не был решен (resolved) пользователем.

## Что такое сурс (источник) конфликта?

Сурс (источник) конфликта - это место создания конфликта. Сурс конфликта нужен для того, чтобы при решении конфликтов дать пользователю подсказку, где система нашла эти конфликты

## Какие классы (регистры) отвечают за конфликты?

StructConflictRegister

StructConflictStrategy

StructConflictSourceStrategy

Существуют следующие виды конфликтов:

Выбор инстанции

## Выбор инстанции

Если система нашла два и более подходящих кандидатов при нахождения статичной ссылки, она создает данный вид конфликта.

Чтобы решить данный конфликт, пользователь должен выбрать **одну** инстанцию из предложенного списка

# Файлы

## Как экспортируются файлы?

Файлы экспортируется путем их регистрации в **ExportStructRefManager**

При экспорте записываются метаданные файла (название и расширение) и сам контент в виде B64.

## Как импортируются файлы?

Файлы не требуют анализа, поэтому третий этап для файлов пропускается.

В четвертом этапе файлы сохраняются в новой системе.

ВАЖНО! Надо, чтобы файлы применились ПЕРВЕЕ структур, которые зависит от файлов. Например, если сама структура зависит от файлов (печатные формы), то надо, чтобы файлы применились перед структурой.

Почему?

Причина: при импорте частей, которые зависит от файлов, они берут fileId с **ImportStructRefManager**. А **ImportStructRefManager** в свою очередь берет эти fileId с ImportStructDto, а значения туда записываются после применения файлов.

Почему файлы не переносятся 1 к 1?

Файлы не переносятся 1 к 1 (то есть у файлов ID будут одинаковыми и в экспортируемых и в импортируемых системах) по причине того, что экспортируемые и импортируемые системы могут быть одной системой, но в разных компаниях. И в таком случае надо будет учитывать вариант, когда файл уже есть в системе с такой же ID, соответственно его не надо применять.

Еще одна причина, почему файлы не переносятся 1 к 1 - полная независимость разных компании (даже в рамках одной системы)

# Статичные ссылки

## Что такое статичные ссылки?

**Статичные ссылки** - ссылки в разных частях системы, которые ссылаются на конкретные инстанции.

Примеры статичных ссылок:

Значение по умолчанию у полей с типом БО и СО

Пользователь, департамент или рабочая группа в правах

Ссылки в скриптах

## Какие классы (регистры) отвечают за логику статичных ссылок?

StructStaticRefRegister

StaticRefImportAnalyzer

## Экспорт статичных ссылок

Если вышеописанные части системы имеют статичную ссылку, значит надо экспортировать и данную инстанцию (чтобы сохранить целостность структуры)

Экспорт статичных ссылок подразумевает собой регистрацию данных ссылок в **ExportStructRefManager**

При экспорте статичных ссылок скачивается вся полезная информация об инстанции - какие у нее есть поля и какие параметры у этих полей

## Импорт статичных ссылок

При импорте статичных ссылок система применяет следующую стратегию:

Вначале проверяется, имеются ли уникальные поля у инстанции. Если есть хотя бы одно уникальное поле, система выбирает **первый способ** нахождения подходящего кандидата для статичной ссылки.

Если нет ни одного уникального поля, система выбирает **второй способ** нахождения подходящего кандидата.

**Первый способ**:

Система собирает все инстанции, у которых есть совпадение (матч) по уникальным полям и сортирует по убыванию.

Если инстанция с наибольшим матчем не имеет конкурента (то есть две и более инстанции не имеют n совпадений, где n - количество максимального матча по уникальным полям), то выбирается данная инстанция.

Если у две и более инстанций имеются одинаковое количество матчов по уникальным полям, то создается конфликт.

Если система не смогла найти ни одну инстанцию по уникальным полям, то она создает инстанцию с этими же значениями.

**Второй способ**:

Система делает фильтрованный запрос (в эластик) на получение таблицы с инстанциями.

Фильтр ставится по первым n полям, а значение для фильтра берутся с импортируемой инстанции.

n настраивается в конфиге **StructImportConfig**

Если система нашла две и более инстанции, то создается конфликт.

Если система нашла одну инстанцию, то выбирается данная инстанция.

Если система не нашла ни одну инстанцию, то создается инстанция со значениями импортируемой инстанции.

ВАЖНО УЧЕСТЬ:

Процесс создания инстанции будет происходить на 4-ом этапе, также система НЕ создает конфликт в таком случае

# Введение

## Общая часть

Экспорт/импорт структуры (ЭИС) - один из модулей myBpm, предназначенный для полного или частичного переноса структуры компании между инстанциями myBpm (как между компаниями, так и между системами). Сам процесс переноса выглядит следующим образом: в компании-доноре (компания, структуру которой необходимо перенести) в окне экспорта происходит выгрузка файл с расширением" .mybpm". В этом файле лежит структура компании-донора в json-формате. Далее, этот файл загружается в компанию-реципиент (компания, куда необходимо перенести структуру), где в случае отсутствия ошибок и конфликтов, можно применить эту структуру. Вся логика строится на кодах (не идентификаторы).

Логически ЭИС разделен на 4 этапа

1. Подготовка экспорта
2. Экспорт
3. Подготовка импорта (анализ импорта)
4. Импорт

На данный момент используя ЭИС можно перенести следующие части системы:

Структура БО

Поддерживаются следующие типы БО: БО, CO, Панель, Справочник, Процессы. (17.07.23)
Поддерживаются следующие сущности: динамические поля, нативные поля, виджеты, контекстное меню, вкладки (табы), печатные формы, статичные ссылки (17.07.23)
Поддерживается перенос процессов (фигуры, блоки, выражения) (17.07.23)

Права доступа

Поддерживаются права на следующие сущности: сам БО, нативные поля, динамические поля, виджеты, табы (17.07.23)

Скрипты

Переносить можно лишь скрипты, имеющие рабочую версию, в противном случае выгрузка скриптов будет недоступна (17.07.23)

## Подготовка экспорта

Основные классы

ExportStructRegister
AbstractDependencyWorker
ExportValidator

Основная коллекция

CompanyExportStruct

### Основная логика

После добавления БО или изменения выгружаемых частей, для экспорта, происходит следующее

* Изменения сохраняются в основную коллекцию
* Вызывается метод loadExportData(), в котором происходит валидация выбранных БО и расчет зависимостей от других БО

В случае возникновения ошибок во время валидации - экспорт невозможен.

### Дополнительно

Максимальное количество ошибок, сгенерированных во время валидации прописано в конфиг-файле **StructExportConfig**

## Экспорт

Основные классы

ExportStructRegister
StructureRegister
StructAccessRightsRegister
ScriptsStructureRegister
StructStaticRefRegister
ProcessStructureRegister

После нажатия кнопки "Выгрузить" начинается процесс экспорта структуры. С коллекции CompanyExportStruct скачиваются все выбранные в 1 этапе БО и выбранные для экспорта части (структура, права, скрипты). Далее создаются структуры (java-классы, как правило, в названии имеют "Struct") хранящие в себе всю необходимую информацию для импорта. Основной моделью является CompanyStruct, которая переводится в json-формат посредством библиотеки jackson (через ObjectMapper). По итогу экспорта скачивается mybpm-файл.

## Подготовка импорт (анализ импорта)

Основные классы

ImportStructRegisterImpl
AbstractImportAnalyzer
StructureRegister
StructAccessRightsRegister
ScriptsStructureRegister
StructStaticRefRegister
ProcessStructureRegister

Основная коллекция

ImportStructDto

После загрузки файла полученного во 2 этапе в другую компанию, начинается 3 этап - анализ импорта.

Во время анализа происходит следующее:

1. Определяется возможность импортировать структуру из файла (валидация).
2. Сбор необходимых данных для 4 этапа (идентификаторы для сохранения)
3. Сбор информации о результате импорте для отображения пользователю (что создастся, что обновится и так далее)

## Импорт

Основные классы

ImportStructRegisterImpl
StructFileRegister
StructureRegister
StructStaticRefRegister
ScriptsStructureRegister
StructAccessRightsRegister
ImportStructureStrategy

Если же во время анализа не было обнаружено ошибок, то файл можно применить, другими словами начинается 4 этап - импорт.

В рамках этого этапа начинается создание DTO через кафку, идентификаторы для сохранения, как правило, определяются уже на 3 этапе.

# Права доступа

Этапы:

1) Пропускается, так как у прав нет превалидации

2) На данном этапе происходит процесс экспорта прав.

Основная логика в **StructAccessRightsRegister.export()**

Процесс конвертации основных моделек (*AccessGroupDto*, *AccessSetDto*) в вспомогательные модельки (*AccessGroupStruct*, *AccessSetStruct*) происходит в **AccessRightsImportUtil**

Экспортируются:

a) Права выбранного БО

б) Права всех полей (динамические, нативные и виджеты)

с) Права вкладок

Вышеописанные части БО экспортируются используя коды, эти коды потом (на следующем этапе) маппятся (mapping) с кодами другой системы

3) На данном этапе система анализирует импортируемые права

Анализом занимается **AccessRightsImportAnalyzer**, который создается и вызывается в **StructAccessRightsRegister.analyze()**

Анализатор попытается найти все БО и записать новые права в соответствующий ImportRecordDto. Если БО не было найдено, запишется ошибка в коллекцию ошибок

После успешного нахождения начинается процесс обратной конвертации: вспомогательные модельки конвертируются в основные и запишутся в ImportRecordDto

4) На данном этапе система применяет изменения по правам

Важно отметить, что права ПЕРЕЗАПИСЫВАЮТСЯ, даже если экспортируемые права - дефолтные (по умолчанию)

# Бизнес-процессы

### Общая часть

Используя ЭИС можно перенести структуру Бизнес-процессы

### 1 этап - подготовка экспорта

#### Валидация

Для БП отсутствует валидация

#### Зависимости

На данный момент существуют следующие виды зависимостей в БП

1. Из блока типа "BlockSetField"
2. Из выражения типа "ExprValue"
3. Из выражения типа "ExprBoiFind"
4. Из выражения типа "ExprNewBoi"
5. Из выражения типа "ExprConst"

### 2 этап - экспорт

#### Основной класс

ProcessStructureRegister

#### Основной метод

createBoProcessVersionStruct()

#### Основные модели

BoProcessVersionsStruct
BoProcessStruct

Во время 2 этапа происходит формирование структур, в качестве идентификаторов используются коды. Важно отметить, что выгружаются лишь рабочая и последняя тестовая версии.

### 3 этап - подготовка импорта (анализ импорта)

#### Основные классы

StructureImportAnalyzer

#### Потенциальные ошибки анализа импорта

\_Отсутствие БО на который ссылается БП\_

#### Идентификаторы

Во время 3 этапа собираются идентификаторы для сохранения в 4 этапе. Так например, если импортируемый БО не был найден в системе то он будет сохранен с идентификатором с поля id класса BoStruct. Если же БО был найден в системе по коду, то произойдет слияние структуры из импорта и системы. Также, в случае обновления, рабочей версией становится, созданная из импорта.

# Скрипты

### Общая часть

Используя ЭИС можно перенести структуру скриптов.

### 1 этап - подготовка экспорта

#### Валидация

Для экспорта скриптов отсутствует валидация

#### Зависимости

На данный момент существуют следующие виды скриптовых зависимостей

1. Из блока типа "BlockSetField"
2. Из выражения типа "ExprValue"
3. Из выражения типа "ExprBoiFind"
4. Из выражения типа "ExprNewBoi"
5. Из выражения типа "ExprConst"

### 2 этап - экспорт

#### Основной класс

ScriptsStructureRegister

#### Основной метод

BoScriptVersionsStruct exportScriptStructure(String boId, ExportStructRefManager refManager, AuthUserInfo authInfo);

#### Основные модели

BoScriptVersionsStruct
BoScriptsStruct

Во время 2 этапа происходит формирование структур, в качестве идентификаторов используются коды. Важно, экспортируются лишь рабочая и последняя тестовая версии скриптов.

### 3 этап - подготовка импорта (анализ импорта)

#### Основные классы

StructureImportAnalyzer

#### Потенциальные ошибки анализа импорта

Не найден БО-владелец для скрипта

Если на 2 этапе были выгружены только скрипты без структуры, а в системе-рецепиенте не будет найден БО для которого они были выгружены (поиск идет по коду БО), то возникнет ошибка

Не найден БО на который ссылается блок или выражение внутри скриптов

К примеру, если внутри скриптов есть выражение тип "ExprNewBoi" ссылающийся на БО с кодом "city", но в системе-реципиенте и в импорте такого БО нет, то произойдет ошибка.

В ходе анализа помимо валидации собираются данные для отображения на клиенте. Для скриптов это

* Параметры запуска
* Рабочая версия или нет
* Информация о полях, на которых стоит триггер

#### Идентификаторы

Во время 3 этапа собираются идентификаторы для сохранения в 4 этапе.

# Структура

### Общая часть

Используя ЭИС можно перенести структуру БО, СО, Панелей, Справочников и Процессов.

### 1 этап - подготовка экспорта

#### Валидация

При подготовке экспорта происходит валидации выбранных БО, на возможность экспорта. Для структуры происходит проверка на наличие кодов в:

* самом БО
* печатных формах
* динамических полях
* виджетах
* табах
* в опциях выпадающего списка (dropdown single) и одиночного выбора (radio button group)
* в пунктах опросника

#### Зависимости

На данный момент существуют 3 виде структурных зависимостей

1. БО являющийся частью Составного объекта (СО)
2. Вложенные БО (2 уровня)
3. Справочник, на который ссылается выпадающий список

### 2 этап - экспорт

#### Основной класс

StructureRegister

#### Основной метод

BoStruct exportBoStructure(String boId, ExportStructRefManager refManager, AuthUserInfo authInfo)

#### Основные модели

BoStruct
BoFieldStruct

Во время 2 этапа происходит формирование структур, в качестве идентификаторов используются коды.

### 3 этап - подготовка импорта (анализ импорта)

#### Основные классы

StructureImportAnalyzer

#### Потенциальные ошибки анализа импорта

\_Несоответствие типов БО\_

Если во время анализа был найден БО по коду, но с другим типом, то возникает эта ошибка.
Пример возникновения, из тестового сервера был выгружен СО с кодом "Client", а на прод-среде был найден Справочник с таким же кодом.

\_Несоответствие типов бизнес-полей\_

Если во время анализа был найден БО по коду с идентичным типом, происходит проверка совместимости бизнес-полей, если было найдено поле по коду, но с иным типом то выйдет эта ошибка
Пример возникновения: из тестового сервера был выгружен БО с полем Текстовое поле с кодом "name", а на прод среде по этому коду был найдено поле с типом Почта

\_Изменения уникальности поля\_

Если во время анализа был найден БО по коду с идентичным типом, происходит проверка совместимости бизнес-полей, если было найдено поле по коду c таким же типом, но с разными настройками уникальности, то возникает эта ошибка.

\_Отсутствие БО для СО\_

Эта ошибка возникает, если для импортируемого СО не был найден составной БО ни в системе, ни в импорте.

\_Отсутствие нужного БО для бизнес-поля с типом "Вложенный объект"\_

Эта ошибка возникает, если для бизнес-поля импортируемого БO не был найден БО ни в системе, ни в импорте.

В ходе анализа помимо валидации собираются данные для отображения на клиенте. Для структуры это

* Новое название БО
* Новые поля
* Измененные поля
* Новые вкладки в контекстном меню
* Новые виджеты
* Измененные виджеты

#### Идентификаторы

Во время 3 этапа собираются идентификаторы для сохранения в 4 этапе. Так например, если импортируемый БО не был найден в системе то он будет сохранен с идентификатором с поля id класса BoStruct. Если же БО был найден в системе по коду, то произойдет слияние структуры из импорта и системы.