JEXL_Guide

Выпуск 0.0.1

мая 27, 2026

Содержание

1 Введение и основы JEXL

1.1 Введение в языки выражений

Языки выражений — специализированные языки, которые вычисляют выражения во время работы программы. Они позволяют динамически выполнять математические, логические и строковые операции, обращаться к свойствам объектов и вызывать методы без компиляции кода.

Языки выражений выполняются во время работы программы и позволяют гибко менять логику — например, проверять условия `user.age > 18` или вычислять формулы `price * quantity` без перезапуска приложения.

Зачем нужны языки выражений:

- гибкость — изменение логики без перекомпиляции приложения;
- безопасность — ограниченный набор операций снижает риски выполнения вредоносного кода;
- удобство — простой синтаксис для вычислений и проверок условий;
- динамическое вычисление — выражения могут загружаться из внешних источников.

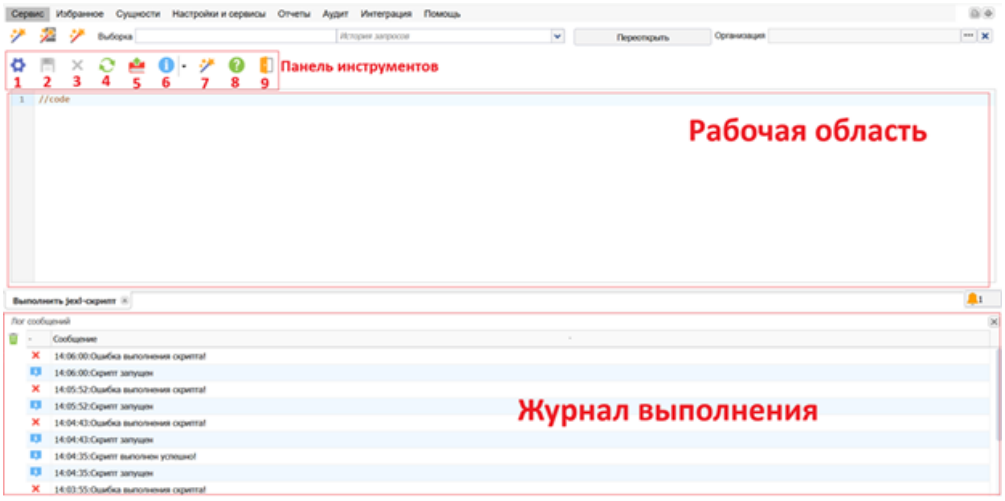
JEXL — библиотека, реализующая язык выражений для вычисления и выполнения скриптов в Java-приложениях.

1.2 Редактор JEXL

Как открыть редактор JEXL:

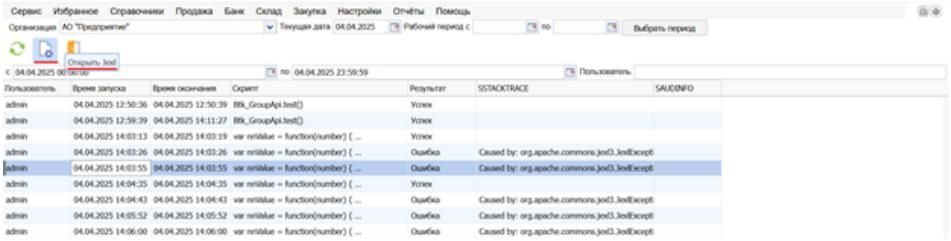
Откройте главное меню системы и перейдите в `Сервисы > Инструменты > Выполнить JEXL-скрипт`

Редактор состоит из трёх областей:



Панель инструментов — набор кнопок для управления скриптами:

1. **Выполнить** — запуск текущего скрипта.
2. **Сохранить**.
3. **Откатить**.
4. **Обновить**.
5. **Открыть скрипт из библиотеки JEXL** — выбор скрипта из общего/личного хранилища.
6. **Информация + Аудит JEXL** — просмотр метаданных и истории изменений



7. **Проверить правильность JEXL-скрипта** — проверка синтаксиса написанного кода.
8. **Помощь** — краткая справка по существующим методам:
    - Помощь — вкладка с методами и их кратким описанием.
    - Список объектов — список объектов API всей системы.
    - Контекст — фильтр отображения методов по контексту, API, математическим функциям.
    - К оглавлению — возврат к редактору кода.
    - Обновить.
    - Проверить правильность JEXL-скрипта.
    - Официальная документация.

9. **Выход**.

Рабочая область — написание и редактирование JEXL-кода.

Журнал выполнения — хронология выполнения скриптов, ошибки и предупреждения, системные сообщения.

- Очистить логи — удаление всей текущей истории сообщений из журнала.

1.3 Основы синтаксиса выражений

Простые выражения

JEXL поддерживает стандартные операторы, аналогичные Java.

Арифметика:

```
var sum = 5 + 3;        // 8
var total = 8 * 10;       // 80
var remainder = 10 % 3; // 1
```

Логические операции:

```
var isActive = true;
var isEligible = (age >= 18) && isActive;
var isAdmin = (role == "admin") || (role == "superuser");
```

Сравнения:

```
price == 100
name != 'admin'
score > 90
```

Поддерживаемые типы данных

JEXL работает с основными типами:

- числа — целые (42), дробные (3.14);
- строки — в одинарных („text“), двойных («text») или обратных кавычках (`test`);
- булевы значения — true, false;
- null — отсутствие значения;
- коллекции — списки [1, 2, 3], мапы {„key“: „value“}, множества {1, 2, 3}.

Обертки для безопасной передачи данных

В JEXL для работы со Scala-методами, требующими строгой типизации и поддержки null, используются специальные обертки.

Scala — статически типизированный язык, где:

- null для примитивов (Long, Int) запрещен.

При вызове Scala-методов из JEXL:

- JEXL передает «сырые» значения (String, Number, null);
- Scala ожидает строгие типы (`NLong`, `NGid`).

Null-типы и их назначения:

Null-тип	Назначение
`NLong`	Работа с идентификаторами (ID)
`NNumber`	Работа с целыми/дробными числами
`NGid`	Работа с глобальными идентификаторами (GUID)
`NDate`	Работа с датами
`NString`	Работа со строками (с поддержкой null)
`NBigDecimal`	Точные расчеты (деньги, финансы)
`NDuration`	Арифметика временных промежутков

Создание Null-типа на примере `NLong`:

```
var NLong = function(number) {
    return new ("ru.bitec.app.gtk.lang.NLong", number);
};
var nlValue = NLong(21);
```

Конкретное применение null-типов:

```
// Создаем Map с параметрами для заказа
Map1.put(
    new ("ru.bitec.app.gtk.lang.NLong", ropStageDet.id),
    new ("ru.bitec.app.gtk.lang.NNumber", ropStageDet.nQty)
);

// Создаем заказ
idvOrder = Stm_OrderOutApi.createOrderByContractStage(
    ropStage.id,
    idvDepOwner,
    idvOrderType,
    asScala(Map1), // Основные параметры
    asScala(EmptyMap), // Доп. параметры 1
    asScala(EmptyMap), // Доп. параметры 2
    null
);
```

1.4 Работа с переменными

Способы объявления переменных

JEXL предоставляет три способа объявления переменных: `let`, `const`и `var`, каждый с своей областью видимости и правилами изменяемости.

`let` — **Локальная переменная**

Ключевое слово `let` создаёт переменную, доступную только внутри блока { }, где она объявлена, включая вложенные блоки. Переменная не может быть переопределена в той же области.

Пример:

```
if (56 < 100) {
    let sText = "Example text";
}
dialogs.showMessage(sText) // ошибка: переменная объявлена в другом блоке
```

`const`— **Неизменяемая переменная**

Переменная, объявленная через const, имеет блочную область видимости, но её значение нельзя изменить после инициализации.

Пример:

```
if (56 < 100) {
    `const`sText = "Example text";
    sText = "Changed text"; // Ошибка: изменение константы
}
```

Исключение: если `const`ссылается на объект, его внутренние свойства могут изменяться — неизменной остаётся только ссылка.

Пример:

```
`const`arrayEven = [2, 4, 6, 8];
arrayEven[0] = 10;
dialogs.showMessage(arrayEven[0]); // Вывод: 10
```

var — Глобальная переменная

По умолчанию `var` создаёт переменную, видимую во всём скрипте, и позволяет её переопределять:

```
if (56 < 100) {
    var sText = "Example text";
    sText = "Changed text";
}
sText = "Outer text"; // Изменение разрешено
dialogs.showMessage(sText); // Вывод: Outer text
```

Когда что использовать:

- `const`— для значений, которые не должны меняться;
- `let` — для обычных переменных внутри блоков;
- `var` — для глобальных переменных (с осторожностью).

Ленивое вычисление значения

В JEXL нет отдельной конструкции `lazy val`. Если значение нужно вычислить только при первом обращении и затем переиспользовать, следует применять явную ленивую инициализацию:

- значение-кэш хранить в `var`;
- доступ к нему выполнять через функцию;
- при первом вызове функция вычисляет значение и сохраняет его в кэш;
- при последующих вызовах возвращает уже сохранённый результат.

Такой подход удобно использовать для дорогих вычислений, повторных SQL-запросов и получения данных, которые в рамках одного выполнения скрипта не меняются. `var` в текущем руководстве описан как переменная, видимая во всём скрипте, а `const`— как неизменяемая ссылка, поэтому для такого шаблона обычно используют `var` для кэша и `const`или function для accessor-функции. (Global ERP)

Пример

```
var vSettingsCache = null;

`const`getSettings = function() {
    if (isNull(vSettingsCache)) {
        vSettingsCache = sql(`
            select code, name
            from cfg_settings
            where is_active = true
        `).asList();
    }
    return vSettingsCache;
};

var settings1 = getSettings(); // первое обращение: выполняется запрос
var settings2 = getSettings(); // повторное обращение: используется кэш
```

Рекомендации

- не выполнять тяжёлое выражение напрямую в нескольких местах;
- выносить ленивую инициализацию в отдельную функцию с понятным именем;
- использовать такой приём только в пределах одного JEXL-скрипта;
- не применять этот шаблон там, где значение должно пересчитываться при каждом обращении.

Объявления переменных поддерживаемых типов

Числовые типы:

```
var nValue = 42; // Целое число
var fValue = 42.0f; // Число с плавающей точкой
var lValue = 42L; // Длинное целое
var dValue = 42.0d; // Число двойной точности
var bigIntValue = 42H; // Большое целое
var bigDecValue = 42.0B; // Число высокой точности
var hexValue = 0x2A; // Шестнадцатеричное число
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
var octValue = 052; // Восьмеричное число
var sciValue = 4.2E+1D; // Число в научной нотации
```

Строковые типы:

```
var sStr1 = "Hello"; // Строка (двойные кавычки)
var sStr2 = 'World'; // Строка (одинарные кавычки)
var sStr3 = `line1`; `line2`; `line3`; // Многострочный комментарий;
var intValue = 10
var interpolatedStr = `Value: ${intValue}`; // Строка с подстановкой
var regex = ~/pattern\d+/; // Регулярное выражение
```

Специальные значения:

```
var bVal = true; // Логическое значение
var nVal = null; // Пустое значение
var tuple2 = (v1, v2) -> { return new(`scala.Tuple2`, v1, v2); }; // Кортеж
```

Коллекции:

```
var array = [1, 2, 3]; // Массив
var objArray = [1, "two", 3.0]; // Массив объектов
var list = [1, 2, 3, ...]; // Список
var set = {"a", "b", "c"}; // Множество
var map = {"key1": 1, "key2": 2}; // Карта
var range = 1..10; // Диапазон чисел
var rop = Stm_OrderOutApi.load(id); // Строка роп соответствующая объекту БД
```

Пустые коллекции:

```
var emptyArray = []; // Пустой массив
var emptyList = [...]; // Пустой список
var emptySet = {}; // Пустое множество
var emptyMap = {:}; // Пустая карта
```

Доступ к свойствам объектов

Два варианта доступа к объекту:

```
// Через точку (автоматически вызывает getter)
var sClientName = order.client.name;

// Через метод
var nTotal = order.getTotal();
```

Импорт статических обектов

Синтаксис:

```
#pragma import <Путь до класса>
#pragma import "<Путь до класса> as алиас"
```

Импортированный объект доступен будет доступен по наименованию класса `<Имя_класса>:` или по алиасу, при его указинии `<алиас>:`

Быстрый ввод: введите `import` и вызовите подсказки сочетанием `Ctrl+Enter`.` Варианты подсказок:

```
import named    ->  #pragma import "class as alias"
import default  ->  #pragma import class
```

При вводе полного имени класса (`ru.bitec.app.gtk.lang.NLong`) и просто имени класса (`NLong`) работают подсказки полного пути

Пример:

```
#pragma import ru.bitec.app.gtk.lang.NLong
#pragma import "ru.bitec.app.gtk.lang.NNumber as nr"
// Вызов статического метода
let dec = nr:fromAny("10.01");
let long = NLong:fromAny("10");
// Создание класса
new nr(dec);
new NLong(long);
```

2 Запись JEXL-макросов

Сервис записи JEXL-макросов преобразует действия пользователя в готовые JEXL-скрипты. Инструмент создает сценарии автоматизации без ручного написания кода и глубокого изучения JEXL.

Основные сценарии использования:

- Массовое создание объектов — добавление позиций номенклатуры, контрагентов, других сущностей.
- Пакетное обновление данных — изменение реквизитов у группы объектов.
- Формирование отчетных форм — создание печатных форм с настройкой шаблонов.
- Импорт данных из файлов — обработка XLSX-файлов для переноса информации.

Примечание

Записываются только вызовы `Api` и `Pkg`. Макросы не содержат вызовы интерактивной логики.

2.1 Создание макросов

Начало записи

1. Перейдите в **Сервис** -> **Инструменты** -> **Начать запись JEXL-скрипта**.
2. В главном меню появится красный флаг — индикатор записи.

> **Примечание**
>
> Перед записью сохраните или откатите все несохраненные данные во всех формах.
>
> При несохраненных данных система выдаст ошибку о невозможности изменения активности перехвата вызовов методов.

Запись действий

После начала записи система отслеживает все действия. Выполните операции с данными, которые нужно записать.

Завершение и сохранение

1. Остановите запись через:
   - Красный флаг в главном меню.
   - **Сервис** -> **Инструменты** -> **Закончить запись JEXL-скрипта**.
2. Просмотрите записанный скрипт в появившемся окне.
3. При закрытии окна подтвердите сохранение.
4. Выберите папку (по умолчанию — **Личные**).
5. Скрипт сохранится в Библиотеке скриптов.

2.2 Просмотр и редактирование

Для хранения, просмотра, редактирования и запуска сохраненных скриптов используется библиотека JEXL.

Путь: `Настройка системы > Настройки и сервисы > Сервисы JEXL > Библиотека JEXL`.

В библиотеке отображается дерево каталогов и скриптов. По умолчанию доступны два каталога:

- **Личные** — содержит скрипты, доступные только текущему пользователю.
- **Общего назначения** — содержит скрипты, доступные всем пользователям.

С помощью операций на панели инструментов можно создавать и удалять каталоги и скрипты, формировать иерархию каталогов и перемещать скрипты между каталогами.

Чтобы открыть скрипт в редакторе, дважды нажмите на нужный пункт дерева. Открытый скрипт можно просмотреть, изменить и выполнить.

Аудит запусков скрипта

На вкладке **Аудит** отображаются результаты запуска выбранного скрипта.

Данные можно отфильтровать:

- по дате выполнения;
- по пользователю, который выполнил запуск.

Если запуск завершился с ошибкой, подробности отображаются в окне **Стек ошибок**.

В окне **Дополнительная информация** отображаются данные, которые автор скрипта добавил в лог с помощью команды `audInfo`.

При каждом выполнении скрипта из библиотеки система создает новую запись аудита.

2.3 Примеры скриптов

Создание печатной формы:

```
var rop_Rpt_Report1 = Rpt_ReportApi.insert();
Btk_ObjectGroupApi.register(rop_Rpt_Report1, 27851L, 1B, 1B);
Rpt_ReportApi.setsSystemName(rop_Rpt_Report1, 'SomeReportName');
Rpt_ReportApi.setsCaption(rop_Rpt_Report1, 'Отчет');
Rpt_ReportApi.setidModule(rop_Rpt_Report1, 901L);
var rop_Rpt_ReportVersion1 = Rpt_ReportVersionApi.insertByParent(rop_Rpt_Report1);
Rpt_ReportVersionApi.setidReportType(rop_Rpt_ReportVersion1, 451L);
```

В сформированном скрипте все идентификаторы имеют исходный вид. Если этот скрипт планируется использовать на других базах данных, где идентификаторы будут отличаться, необходимо заменить их получение в скрипте - например, на метод `findByMnemoCode`.

Видоизмененный скрипт:

```
var rop_Rpt_Report1 = Rpt_ReportApi.insert();
Btk_ObjectGroupApi.register(rop_Rpt_Report1, 27851L, 1B, 1B);
Rpt_ReportApi.setsSystemName(rop_Rpt_Report1, 'SomeReportName');
Rpt_ReportApi.setsCaption(rop_Rpt_Report1, 'Отчет');
Rpt_ReportApi.setidModule(rop_Rpt_Report1, Btk_ModuleApi.findByMnemoCode('btk'));
var rop_Rpt_ReportVersion1 = Rpt_ReportVersionApi.insertByParent(rop_Rpt_Report1);
Rpt_ReportVersionApi.setidReportType(rop_Rpt_ReportVersion1, Rpt_ReportTypeApi.
↪findByMnemoCode('jasper'));
```

2.4 Обработка данных Excel

Для работы с `xlsx` в `jexl` используется библиотека poi.apache.org.

Для удобства загрузки файлов написана *функциональная библиотека* `Bts_XlsxPkg`. Метод `uploadParseFiles` открывает диалог выбора файла и парсит его, предоставляя разработчику возможность его обработки. Создается объект класса `org.apache.poi.ss.usermodel.Workbook`.

Для выполнения массовой обработки данных из файлов Excel, вы можете использовать следующий метод:

```
lib("Btk_XlsxLib").uploadParseFiles(fun);
```

Где `fun` - это функция Jexl, которая будет выполняться для обработки данных из Excel.

Пример обработки xlsx:

```
var fun = x -> {
  var sheet = x.getSheet("Материалы");
  var lastRowNum = sheet.getLastRowNum();
  var i = 1;
  while (i <= lastRowNum ){
    var svNomName = sheet.getRow(i).getCell(3).getStringCellValue(); //  Условное␣
↪наименование
    var idvMSRItem = sheet.getRow(i).getCell(4).getNumberCellValue().toBigDecimal(); //␣
↪ЕИ
    var idvGost = sheet.getRow(i).getCell(5).getStringCellValue(); // ГОСТ
    var idvSortamentType = sheet.getRow(i).getCell(6).getNumberCellValue().
↪toBigDecimal(); // Тип сортамента

   var rop_Bs_Goods1 = Bs_GoodsApi.insert();
   Bs_GoodsApi.setsNomName(rop_Bs_Goods1, svNomName); // установка условного наименования
   Bs_GoodsApi.setidMeasureItem(rop_Bs_Goods1, idvMSRItem); // установка ЕИ
   Bs_GoodsApi.setsGost(rop_Bs_Goods1, svNomName); // установка условного наименования
   Bs_GoodsApi.setsNomName(rop_Bs_Goods1, svNomName); // установка условного наименования

   commit();
   i = i+1;
  }
  true;
}

lib("Btk_XlsxLib").uploadParseFiles(fun);
```

Приведенный ниже пример функции `fun` иллюстрирует, как обработать данные из файла Excel и создать объекты в системе на их основе.

Расширенная обработка с разными типами данных:

```
var fun = x -> {
// Получение листа "Лист1" из файла Excel
var sheet = x.getSheet("Лист1");

// Определение количества заполненных строк в файле
var lastRowNum = sheet.getLastRowNum();
var i = 0;

while (i <= lastRowNum) {
// Класс для которого мы создаем объекты
var rop_RplTst_AllDbTypes1 = RplTst_AllDbTypesApi.insert();

    // Получение значений из ячеек Excel и преобразование их к необходимым типам
    // Для получения значения типа Long нужно дополнительно использовать метод toLong
    var vjson = sheet.getRow(i).getCell(0).getStringCellValue();
    var vsystemname = sheet.getRow(i).getCell(1).getStringCellValue();
    var vcaption = sheet.getRow(i).getCell(2).getStringCellValue();
    var vgidrefclassany = sheet.getRow(i).getCell(3).getStringCellValue();
    var vidobjecttype = sheet.getRow(i).getCell(4).getNumericCellValue().toLong();
    var vdate = toDate(sheet.getRow(i).getCell(5).getStringCellValue());
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    // Преобразование для NNumber
    var vNumber = sheet.getRow(i).getCell(6).getNumericCellValue().toBigDecimal();

    // Установка значений для созданного объекта
    RplTst_AllDbTypesApi.setjJson(rop_RplTst_AllDbTypes1, vjson);
    RplTst_AllDbTypesApi.setsSystemName(rop_RplTst_AllDbTypes1, vsystemname);
    RplTst_AllDbTypesApi.setsCaption(rop_RplTst_AllDbTypes1, vcaption);
    RplTst_AllDbTypesApi.setgidRefAny(rop_RplTst_AllDbTypes1, vgidrefclassany);
    RplTst_AllDbTypesApi.setNumber(rop_RplTst_AllDbTypes1, vNumber);

    // Проверка и установка idObjectType только если он не равен 0
    if (vidobjecttype != 0) {
      RplTst_AllDbTypesApi.setidObjectType(rop_RplTst_AllDbTypes1, vidobjecttype);
    }

    // Установка даты
    RplTst_AllDbTypesApi.setdDate(rop_RplTst_AllDbTypes1, vdate);

    // Сохранение изменений
    commit();

    // Увеличение счетчика строк
    i = i + 1;
}

// Возврат булевого значения, необходимого для выполнения скрипта
true;
}
```

Запуск скрипта

После написания функции `fun`, выполните скрипт, и появится модальное окно для выбора файлов Excel. Выберите необходимые файлы, и функция начнет обработку данных и создание объектов в системе на основе данных из Excel.

С помощью этой библиотеки вы можете эффективно работать с данными Excel и использовать записанные Jexl-скрипты для массовой обработки и создания объектов по сложным сценариям.

Полный тестовый скрипт обработки Excel:

```
var fun = x -> {
  var sheet = x.getSheet("Лист1");
  var lastRowNum = sheet.getLastRowNum();
  var i = 0;
  while (i <= lastRowNum ){
    var rop_RplTst_AllDbTypes1 = RplTst_AllDbTypesApi.insert();
    var vjson = sheet.getRow(i).getCell(0).getStringCellValue();
    var vsystemname = sheet.getRow(i).getCell(1).getStringCellValue();
    var vcaption = sheet.getRow(i).getCell(2).getStringCellValue();
    var vgidrefclassany = sheet.getRow(i).getCell(3).getStringCellValue();
    var vidobjecttype = sheet.getRow(i).getCell(4).getNumericCellValue().toLong();
    var vdate = toDate(sheet.getRow(i).getCell(5).getStringCellValue());

    RplTst_AllDbTypesApi.setjJson(rop_RplTst_AllDbTypes1, vjson);
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    RplTst_AllDbTypesApi.setsSystemName(rop_RplTst_AllDbTypes1, vsystemname);
    RplTst_AllDbTypesApi.setsCaption(rop_RplTst_AllDbTypes1, vcaption);
    RplTst_AllDbTypesApi.setgidRefAny(rop_RplTst_AllDbTypes1, vgidrefclassany);
    if (vidobjecttype != 0) RplTst_AllDbTypesApi.setidObjectType(rop_RplTst_AllDbTypes1,␣
↪vidobjecttype);
    RplTst_AllDbTypesApi.setdDate(rop_RplTst_AllDbTypes1, vdate);

    commit();
    i = i+1;
  }
true;
}

lib("Btk_XlsxLib").uploadParseFiles(fun);
```

Внимание

- **Пустые ячейки**: могут возвращать 0 вместо null — добавляйте проверки.
- **Типы данных**: следите за соответствием типов Excel и системы. В приведенном примере, используется метод getStringCellValue() для получения текстовых значений, и getNumericCellValue().toLong() для числовых значений.
- **Возвращаемое булево значение**: функция должна возвращать true/false.

2.5 Работа с файлами

Загрузка во временный каталог

```
files.uploadFileToTemp(res -> {
  var file = res.file(); //java.io.File
  //чтение содержимого файла
  var utils = new('org.apache.commons.io.FileUtils');
  var content = utils.readFileToString(file, "UTF-8");

  dialogs.showMessage(content);
  true;
});
```

Примечание

Работает только для jexl-скриптов в контексте выборки.

Выгрузка строки как файла

```
var text = "Привет!"
files.downloadBytes(text.getBytes("UTF-8"), "Имя_Файла.txt", true, false);
```

Примечание

Работает только для jexl-скриптов в контексте выборки.

2.6 Необходимые права

Для использования сервиса:

1. Стандартные права на приложение и доступ к **Сервис** -> **Инструменты** -> **Закончить запись JEXL-скрипта**.
2. Привилегия **Доступ к инструментам меню Сервиса** для объекта `Btk_ManagementPkg`.

3 Коллекции и структуры данных

В этом разделе описаны основные типы коллекций в JEXL: списки, массивы, множества, карты и кортежи. Вы узнаете, как объявлять коллекции, добавлять и удалять элементы, а также выполнять основные операции для работы с данными. Материал содержит практические примеры для каждого типа коллекций.

3.1 Список

Список является изменяемым типом данных (mutable). Это значит, что можно добавлять и удалять элементы, а также изменять существующие.

Исходный массив для примера:

```
var mvStock = ["P-1001", "P-1002", "P-1003", ...];
```

Основные операции

Объявление списков:

```
var mvStockApproachOne = [...];
```

Добавление элементов (только для mutable):

```
mvStock.add("P-1004");
```

Получение элементов:

```
mvStock[0]; // способ 1
mvStock.get(0); // способ 2
```

Проверка наличия элемента:

```
mvStock.contains("P-1002"); // возвращает true/false
```

Удаление элементов (только для mutable):

```
mvStock.remove("P-1002");
```

Размер списка:

```
mvStock.size(); // возвращает количество элементов
```

Проверка на пустоту:

```
mvStock.isEmpty(); // возвращает true/false
```

Поиск индекса элемента:

```
mvStock.indexOf("P-1001"); // возвращает индекс или -1
```

Замена элемента (только для mutable):

```
mvStock.set(0, "newElement"); // способ 1
mvStock[0] = "newElement"; // способ 2
```

Объединение списков (только для mutable):

```
var listUnion = [1,2,3, ...];
mvStockApproachOne.addAll(listUnion);
```

Очистка списка (только для mutable):

```
mvStock.clear(); // удаляет все элементы
```

3.2 Массив

Массив — это изменяемая (mutable) структура данных с фиксированным размером. Вы не можете добавлять и удалять элементы, но можно изменять значения существующих элементов (если они сами по себе изменяемы).

Исходный массив для примера:

```
var mvStock = ["P-1001", "P-1002", "P-1003"];
```

Основные операции

Объявление массива:

```
var mvStockApproachOne = [];
var mvStockApproachTwo = [1, 2, 3];
```

Получение элементов:

```
mvStock[0]; // способ 1
mvStock.get(0); // способ 2
```

Обновление значений:

```
mvStock[0] = 100;
```

Проверка наличия элемента:

```
mvStock.contains("P-1002"); // возвращает true/false
```

Размер списка:

```
mvStock.size(); // количество элементов
```

Поиск индекса элемента:

```
mvStock.indexOf("P-1001"); // возвращает индекс или -1
```

3.3 Множество

Доступны только mutable (изменяемые) множества. **Основные операции**

Создание множества:

```
var setData = {"P-1001", "P-1002", "P-1003"}; // Множество с элементами
var setEmptyData = {}; // Пустое множество
```

Добавление элементов:

```
setData.add("P-1004"); // Добавление одного элемента
setData.addAll({"P-1005", "P-1006"}); // Добавление другого множества
setData.addAll(["P-1007", "P-1008"]); // Добавление элементов из списка
```

При добавлении списка порядок элементов множества сбивается.

Проверка и получение элементов:

```
var exists = setData.contains("P-1001"); // Проверка наличия → true/false
```

Удаление элементов:

```
setData.remove("P-1001"); // Удаление по значению
setData.clear(); // Полная очистка множества
```

Работа с размерами:

```
var size = setData.size(); // Количество элементов
var isEmpty = setData.isEmpty(); // Проверка на пустоту → true/false
```

Операции с множествами:

```
// Пересечение (оставляет только общие элементы)
setData.retainAll({"P-1005", "P-1006"});

// Разность (удаляет элементы другого множества)
setData.removeAll({"P-1002", "P-1003"});
```

3.4 Карта

Далее в примерах будем работать с этой картой:

```
var mapData = {"bwhasError" : 11, "svKeyTransaction" : 21}; // объявление словаря
```

Внимание

Если в карте добавляется несколько элементов с одинаковыми ключами, то последнее значение перезапишет предыдущее. В словаре все ключи уникальны, и каждому ключу соответствует только одно значение.

```
var mapData = {
    "bwhasError": 11,
    "svKeyTransaction": 21, // Это значение будет перезаписано
    "svKeyTransaction": 22 // Останется только это
};
// Итоговая карта: {"bwhasError": 11, "svKeyTransaction": 22}
```

Основные операции со словарями

Получение и добавление элемента:

```
mapData.e = 5; // добавление нового ключа (способ 1)
mapData.put("c", 3); // добавление нового ключа (способ 2)
mapData["d"] = 4; // добавление нового ключа (способ 3)

var approachTwo = mapData.bwhasError; // получение значения по ключу (способ 1)
var approachOne = mapData.get("bwhasError"); // получение значения по ключу (способ 2)
var approachThree = mapData["bwhasError"]; // получение значения по ключу (способ 3)
```

Получить и добавить значению по ключу можно тремя способами: через точку, квадратные скобки, метод get.

- Точечная нотация работает только с ключами, которые являются валидными идентификаторами (без пробелов, дефисов, начинающиеся с буквы).
- Для ключей со спецсимволами (тире, пробелы, точки) используйте квадратные скобки.
- Методы put()/get() работают всегда, но требуют больше кода.

Пример с ограничениями:

```
var mvStock = {"P-1001" : 1, "P-1002" : 2, "P-1003" : 3};
var nCode = mvStock.P-1001; // Дефис воспримется как оператор
```

Проверка наличия ключа:

```
mapData.containsKey("bwhasError"); // возвращает true/false
```

Удаление элемента:

```
mapData.remove("bwhasError"); // удаляет пару "ключ-значение" по ключу
```

Размер словаря:

```
mapData.size(); // количество пар "ключ-значение"
```

Получение всех ключей:

```
mapData.keySet(); // возвращает массив ключей (например,["bwhasError", "svKeyTransaction
↪"])
```

Получение всех значений:

```
mapData.values(); // возвращает массив значений (например, [11, 21])
```

Проверка на пустоту:

```
mapData.isEmpty(); // true, если карта пустая
```

Изменение значения по ключу:

```
mapData["bwhasError"] = 10; // присваивает новое значение ключу
```

Добавление новой пары «ключ-значение»:

```
mapData["newKey"] = 42; // добавляет новый ключ с значением
```

Очистка словаря:

```
mapData.clear(); // удаляет все элементы
```

Объединение словарей:

```
var newMap = {"key1": 1, "key2": 2};
mapData.addAll(newMap); // добавляет все пары из newMap в mapData (существующие ключи␣
↪перезаписываются)
```

Операция entrySet:

```
var mapData = {"bwhasError" : 11, "svKeyTransaction" : 21};
var setMap = mapData.entrySet(); // Получаем множество пар ключ-значение

for (entry : setMap) { // Итерация по всем элементам (парам ключ-значение)
    var key = entry.getKey(); // Получить ключ текущей пары
    var value = entry.getValue(); // Получить значение текущей пары
}
```

Возвращает множество(Set) пар ключ-значение - объектов Map.Entry.

Каждый Entry содержит:

- getKey() - уникальный ключ;
- getValue() - соответствующее значение.

Перебор значений идет в обратном порядке.

Когда использовать entrySet():

- Когда нужны и ключи, и значения в цикле.
- Для модификации значений через entry.setValue().

- Для получения «снимка» всех данных Map

Получение ключа по значению

Нет встроенной функции для получения ключа по значению, но можно написать свою локальную функцию для этой цели:

```
// Получения ключа по значению
var findKeysByValue = function(map, targetValue) {
    var matchedKeys = [...]; // Сюда будем складывать подходящие ключи
    for (let key : map.keySet()) { // Перебираем каждый ключ
        if (map[key] == targetValue) { // Если значение по ключу совпадает с искомым
            return key; // Добавляем ключ в результат
        }
    }
    return matchedKeys; // Возвращаем найденные ключи в виде списка
};

var keyListForValue = findKeysByValue(mapData, 11);
dialogs.showMessage(toString(keyListForValue));
```

По такому же принципу можно реализовать удаление ключа по значению.

3.5 Конвертации коллекций

Из `Array[?]` / `scala.collection.IterableOnce[?]` / `java.lang.Iterable[?]`

Доступны методы:

- `toArray`, `toIntArray`, `toFloatArray`, `toDoubleArray`, `toLongArray`, `toCharArray`, `toByteArray`, `toShortArray`, `toBooleanArray`
  — конвертация в `Array` и методы, гарантирующие конвертацию в примитивные массивы.
- `toJArrayList`
  — конвертация в `java.util.ArrayList[?]`
- `toSeq`
  — конвертация в `immutable.Seq[?]`
- `toMSeq`
  — конвертация в `mutable.Seq[?]`
- `toList`
  — конвертация в `immutable.List[?]`
- `toSet`
  — конвертация в `immutable.Set[?]`
- `toMSet`
  — конвертация в `mutable.Set[?]`
- `toJHashSet`
  — конвертация в `java.util.HashSet[?]`
- `toTuple`
  — конвертация в `TupleN`
- `toOption`
  — конвертация в `Option[?]`

Если ? — это Tuple2[K, V]:

- toJHashMap
  — конвертация в java.util.HashMap[K, V]
- toMap
  — конвертация в immutable.Map[K, V]
- toMMap
  — конвертация в mutable.Map[K, V]

Из java.util.Map[?, ?] / immutable.Map[?, ?] / mutable.Map[?, ?]

- toJHashMap
  — конвертация в java.util.HashMap[K, V]
- toMap
  — конвертация в immutable.Map[K, V]
- toMMap
  — конвертация в mutable.Map[K, V]
- toSeq
  — конвертация в immutable.Seq[(K, V)]
- toMSeq
  — конвертация в mutable.Seq[(K, V)]

Примеры использования

```
//Пример создания Seq (часто используется в Api/Pkg)
let ropa = [rop1, rop2].toSeq;
let ida = ropa.map(r => r.id);
//Создание карты через Tuple
let map = [["one", 1].toTuple, ["two", 2].toTuple].toMMap;
//создание Option
let opt = [null].toOption // None
```

3.6 Кортеж

Кортежи — это неизменяемые (immutable) структуры данных, которые позволяют хранить фиксированное количество элементов разных типов. Они полезны, когда нужно временно сгруппировать данные без создания отдельного класса.

В JEXL отсутствует встроенный тип данных Кортеж (Tuple), но его можно эмулировать с помощью интеграции с Scala. Кортежи Tuple2 — это естественный способ передать такие данные из JEXL в Scala-код.

```
var tuple2 = (v1, v2) -> { return new(`scala.Tuple2`, v1, v2); };
var value = tuple2("param1", "param2");
// Или
var value = ["param1", "param2"].toTuple;
```

- New(scala.Tuple2, . . .) — создает экземпляр Scala-кортежа, используя JVM-интеграцию.

- Функция принимает 2 аргумента и возвращает неизменяемую пару (v1, v2).

В Scala стандартные кортежи (Tuple) ограничены 22 элементами (до Tuple22):

```
var tuple22 = (v1, v2, v3,v4, v5, v6,v7, v8, v9,v10, v11, v12, v13, v14, v15,v16, v17,␣
↪v18,v19, v20, v21,v22) -> {
    return new(`scala.Tuple22`, v1, v2, v3,v4, v5, v6,v7, v8, v9,v10, v11, v12, v13, v14,
↪ v15,v16, v17, v18,v19, v20, v21,v22);
};

dialogs.showMessage(toString(tuple22(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,␣
↪16, 17, 18, 19, 20, 21, 22)));
```

Чтобы передавать параметры в Scala-коллекции, используем asScala() или toSeq:

```
// Массив пар [Имя_параметра, Значение]
var tuple2Params = asScala([
    tuple2("npInParam1", 10l), // Параметр 1 (Long)
    tuple2("npInParam2", 125l), // Параметр 2 (Long)
    tuple2("npInParam3", 1000l), // Параметр 3 (Long)
    ... // Остальные элементы (если есть)
]);
//Или
var tuple2Params = [
    ["npInParam1", 10l].toTuple, // Параметр 1 (Long)
    ["npInParam2", 125l].toTuple, // Параметр 2 (Long)
    ["npInParam3", 1000l].toTuple, // Параметр 3 (Long)
    ... // Остальные элементы (если есть)
].toSeq;
```

Особенности:

- asScala() — преобразует JEXL-массив в Scala-коллекцию;
- toSeq - преобразует JEXL-массив в неизменяемый Seq
- каждый элемент коллекции — кортеж («имя», значение).

Кортежи неизменяемы, что гарантирует:

- параметры не поменяются случайно во время выполнения процедуры;
- нет риска коллизий в многопоточных сценариях.

4 Контроль выполнения: условия, функции и обработка ошибок

В этом разделе описаны основные конструкции управления выполнением кода в JEXL: условные операторы, циклы, методы и функции, обработка ошибок. Материал содержит синтаксис и практические примеры для каждого элемента.

4.1 Условные конструкции и циклы

Условия if-else

Синтаксис:

```
if (условие) {
    // блок кода при true
} else {
    // блок кода при false
}
```

Пример:

```
if (order.total > 10000) {
    discount = 0.1;
} else if (order.total > 5000) {
    discount = 0.05;
} else {
    discount = 0;
}
```

Тернарный оператор

Синтаксис:

```
условие ? значение_если_true : значение_если_false
```

Пример:

```
// Простой пример
var access = (id >= 18) ? "Разрешено" : "Запрещено";
```

Циклы

While

Синтаксис:

```
while (условие) {
    // действие
}
```

Пример:

```
// Простой пример
var i = 0;
while (i < 10) {
    total++;
    i++;
}
```

For

При использовании циклов важно понимать область видимости переменных:

- Если итерируемая переменная объявлена с `let` или `const` - она доступна только внутри блока цикла.
- Если переменная объявлена без ключевого слова (неявно) - она ведёт себя как var:
  - Доступна после выполнения цикла.
  - Сохраняет последнее присвоенное значение.
  - Если переменная с таким именем уже существовала, то переменная созданная в цикле затрет значение ранее объявленной переменной.

Способ 1:
Классический C-подобный цикл с индексом.

Синтаксис:

```
for (инициализация; условие; шаг) {
    // тело цикла
}
```

Пример:

```
let x = 0;
for (let i = 0; i < 10; ++i) {
    x += i
}
dialogs.showMessage(toString(x));
```

Подходит также для массивоподобных структур, где есть доступ по индексу `(array[i])`:

```
let x = 0;
let array = [100, 200, 300];
for (let i = 0; i < size(array); ++i) {
    x += array[i];
}
```

Позволяет гибко управлять индексом коллекции (например, менять шаг i += 2).

Способ 2:

Синтаксис:

```
for (let элемент : коллекция) {
    // тело цикла
    // элемент доступен только здесь
}
```

Используется для перебора элементов итерируемой структуры (массива, коллекции и т. д.). Итерируемая переменная получает значение каждого элемента на каждой итерации.

```
let x = 0;
let array = [1, 2, 3];
for (let item : array) {
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    x += item;
}
```

Переменная item доступна только внутри цикла (если используется `let`).

Способ 3:

Синтаксис:

```
for (элемент : коллекция) {
    // тело цикла
    // элемент доступен здесь
}
// элемент ДОСТУПЕН и здесь (ведёт себя как var)
```

Если итерируемую переменную не объявлять никаким ключевым словом(`let`, `const`, `var`), то она будет объявлена как `var`, и будет иметь последнее присвоенное значение.

Пример коллекции:

```
let x = 0;
let array = [1, 2, 3];
for (item : array) {
    x += item;
}
dialogs.showMessage(toString(item)); // Будет выведено значение '3'
```

- item не объявляется явно — она создаётся неявно и «живёт» после цикла.
- Может привести к неожиданным эффектам, если item уже была определена ранее.
- Обход результатов объектного запроса с преобразованием в список.

  ```
  var l= toJRops(Btk_SomeEntityApi.byParent(rop)).asList()
  for(r:l){
      logInfo(r.id);
      logInfo(r.sSystemName)
  }
  ```

Пример множества:

```
var mvStock = {"P-1001", "P-1002", "P-1003"};
for(item : mvStock) {
    dialogs.showMessage(item);
}
```

Пример карты:

```
var mvStock = {"P-1001" : 1, "P-1002" : 2, "P-1003" : 3};
for(item : mvStock) {
    dialogs.showMessage(item); // Здесь item это значения словаря по ключу
}
```

Foreach

Синтаксис:

```
коллекция.foreach(элемент -> {
    // действия с элементом
});
```

В JEXL оператор `foreach` используется исключительно для функционального подхода, когда данные поступают в виде потока (например, из SQL-запроса или коллекции).

Ключевые особенности:

- потоковая обработка;
- нет возвращаемого значения.

Пример:

```
sql("SELECT name FROM users").foreach(row -> {
    logInfo("User: " + row.name);
});
```

4.2 Вызов методов и функций в JEXL

Базовый вызов методов

Синтаксис:

```
объект.метод(параметры)
```

Пример:

```
dialogs.showMessage(ropvInvoice);
dialogs.showMessage(toString(ropvInvoice.nsum));
```

Поддерживается цепочка вызовов:

```
sheet.getRow(0).getCell(1)
```

Определение пользовательской функции

Синтаксис:

```
function имяФункции(параметры) {
    // тело функции
    результат; // опционально
}
```

Пример:

```
function calculateSum(a, b) {
    // Тело функции
    a + b; // Возвращаемый результат (неявный return)
}

dialogs.showMessage(toString(calculateSum(2, 3)));
```

Объявление локальных функций

Синтаксис:

```
var fun_LoadInvoiceTemplate = function(x) {
    // Тело функции
    return true;
};
```

Пример:

```
function test(params) {
    dialogs.showMessage("hello world");
}

const FX_DeleteInvisibleSymbolsF = function(sText) {
    var querry = sql(`Select regexp_replace('${sText}', '[\\s\\u200B\\u200C\\u200D\\
↪uFEFF]', '', 'g') as ShortText;`).asSingle();
    return querry.shorttext;
}

//-----------------------------------------------------------------------------
//main code block
//-----------------------------------------------------------------------------
{
    var sCaptionLong = 'A 694 EM 147_2024 12 105'
    var sCaptionFunc = FX_DeleteInvisibleSymbolsF(sCaptionLong);
    dialogs.showMessage('FUNC: '+ sCaptionFunc);
}

const FX_DeleteInvisibleSymbolsF = function(sText) {
    var querry = sql(`
        SELECT regexp_replace('${sText}', '[\\s\\u200B\\u200C\\u200D\\uFEFF]', '', 'g')␣
↪AS ShortText;
    `).asSingle();
    return querry.shorttext;
}

//-----------------------------------------------------------------------------
// Main code block
//-----------------------------------------------------------------------------
{
    var sCaptionLong = 'A 694 EM 147_2024 12 105';
    var sCaptionFunc = FX_DeleteInvisibleSymbolsF(sCaptionLong);
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    dialogs.showMessage('FUNC: ' + sCaptionFunc);
}
```

4.3 Обработка ошибок

Jexl в ядре не поддерживает исключения, механизм исключений реализован через аннотации.

Синтаксис:

```
@begin{
    //code
}@exception function(e){
    // code
}
```

Для бросания исключения можно воспользоваться командой raise: `raise("Текст исключения")`

Внимание

Конструкция `try - catch` не работает. Вместо нее обязательно необходимо использовать `@begin - @exception`.

Пример:

```
const ErrorHandler = new java.util.concurrent.atomic.AtomicReference(null); // Объявим␣
↪обработчик ошибок

... // Произвольный блок кода

var svLogList = ''; // Переменная для лога успешности и ошибок
ErrorHandler.set(null); // Сбросим обработчик ошибок перед попыткой поймать ошибку

@begin { // Блок кода, в котором попытаемся поймать ошибку
    ... // Произвольный блок кода
    raise("Текст ошибки"); // Событие, которое вызывает ошибку
}
@exception function(vEx1) { // Обработка исключения
    ErrorHandler.set(vEx1); // Сохраняем ошибку в наш обработчик
}; // Обязательно ставим точку с запятой

const ErrorCatch = ErrorHandler.get(); // Запросим значение из обработчика

if (isNotNull(ErrorCatch)) { // Проверяем, была ли ошибка
    svLogList += `\r\n - Ошибка! ${ErrorCatch.getCause().getMessage()}`;
} else {
    svLogList += `\r\n - Успешно!`;
}

dialogs.showMessage(svLogList); // Отображение логов на экране
```

Ограничения именованных параметров

В JEXL синтаксис именованных параметров `method(paramName = value)` **не работает**.

Неверно: `Act_TransDocApi.createBySrc(gidpSrc = aroDoc.gid())`

Верно: `Act_TransDocApi.createBySrc(aroDoc.gid())` — только позиционная передача аргументов в порядке сигнатуры.

5 Особенности работы в системе Global

Система Global предоставляет расширенные возможности для работы с данными через JEXL-скрипты. В статье описаны основные методы работы с переменными, выборками, данными и внешними сервисами.

5.1 Наименование переменных для nullable типов и атрибутов

Правила именования переменных: `[Variable][a][t][Scope][Name][Suffix]`.

Компоненты именования:

- **[Variable]** — определяет тип данных:
  - n — число;
  - s — строка;
  - j — строка в формате JSON;
  - d — дата;
  - r, x — запись;
  - u, cur — курсор;
  - l, blob — бинарные данные;
  - c — символьные данные;
  - b — булево значение;
  - id — идентификатор;
  - gid — глобальный идентификатор.
- **[a]** — определяет, является ли переменная последовательностью;
- **[t]** — определяет пользовательский тип;
- **[Scope]** — область действия:
  - v — переменная процедуры;
  - p — параметр процедуры;
- **[Name]** — имя переменной;
- **[Suffix]** — суффикс:
  - _dz — системные атрибуты;
  - _z — проектные атрибуты.

Примеры именования параметров:

- dStart — дата начала в таблице;
- dvStart — дата начала переменной процедуры;
- dgMaxDate — максимальная дата переменной пакета;
- tvdaDate — тип коллекции дат в процедуре;
- davDate — коллекция дат в процедуре;
- uvStudents — курсор в процедуре.

5.2 Основные методы и принципы работы

Поиск методов класса

Для просмотра доступных методов любого класса выполните следующие шаги:

1. Перейдите в **Настройки системы** → **Обозреватель проектов**.
2. Введите имя класса (например, `Stm_OrderOut`).
3. Откройте файл с суффиксом `Api` (например, `Stm_OrderOut_Api`).
4. Перейдите на вкладку **Методы**.

В этом разделе отображаются все доступные методы класса. Большинство классов содержат стандартный набор методов для работы.

Работа с текущей выборкой

При запуске JEXL-скрипта часто происходит работа внутри существующей выборки (например, при выделении строки в таблице).

С помощью `selection.getVar` можно получить атрибуты текущей выборки:

```
var id = selection.getVar("атрибут");
var idvOrder = selection.getVar("id"); // "id" - название атрибута
```

Загрузка строки через rop

rop — специальный тип данных для переменных, соответствующий строке объекта в таблице базы данных. Внутри данной строки можно обратиться к любому физическому столбцу. По сути это проводник к строке в базе данных.

Принцип работы:

1. Сначала проверяет наличие данных в кэше.
2. Если данных нет — загружает их из базы.
3. Предоставляет удобный доступ к полям строки.

Получить строку можно с помощью метода load. Полученное значение будет типа rop:

```
ropOrder = Stm_OrderOutApi.load(idvOrder);
```

Доступ к атрибутам строки

Обычные атрибуты (колонки таблицы) доступны напрямую через точку:

```
var svNumDoc = ropOrder.sNumDoc; // Номер документа
var idvClient = ropOrder.idClient; // ID клиента
```

Работа с JSON-атрибутами на примере Stm_OrderOut

Получение значений атрибутов:

1. Получение всего JSON объекта:

```
var jvParams = ropOrder.jParams_dz; // Получаем полный JSON-объект параметров
```

2. Получение конкретного значения по имени атрибута:

```
var svStation = Stm_OrderOutApi.getObjAttrValue(ropOrder, 'station'); // Получаем␣
↪значение атрибута 'station'
```

3. Получение значения по ID атрибута:

```
// Получаем ID класса из текущей выборки
var idvClass = selection.getVar('idClass'); // Получим id класса
// Находим ID атрибута по мнемокоду 'station' для указанного класса
var idvAttrStation = Btk_AttributeApi.findByMnemoCode(idvClass, 'station');
// Получаем значение атрибута для указанного заказа
var svStationById = Stm_OrderOutApi.getObjAttrValue(ropOrder, idvAttrStation);
```

Установка значений атрибутов:

1. Установка по имени атрибута — передать название класса и название атрибута:

```
Stm_OrderOutApi.setObjAttrValue(ropOrder, 'station', "Новая станция");
```

2. Установка по ID атрибута:

```
// Получаем ID класса из текущей выборки
var idvClass = selection.getVar('idClass');
// Находим ID атрибута по мнемокоду 'station' для указанного класса
var idvAttrStation = Btk_AttributeApi.findByMnemoCode(idvClass, 'station');
// Получаем значение атрибута для указанного заказа
var svStationById = Stm_OrderOutApi.getObjAttrValue(ropOrder, svStationById);
Stm_OrderOutApi.setObjAttrValue(ropOrder, svStationById, "Новое значение"); // Установка␣
↪значения для атрибута с указанным ID
```

Изменение значений

Чтобы изменить значение в строке, используйте специальные методы-сеттеры. У каждого класса свои сеттеры (смотрите в методах API):

```
// Меняем номер документа
Stm_OrderOutApi.setsNumDoc(ropOrder, "НОВЫЙ-001");

// Меняем клиента
Stm_OrderOutApi.setidClient(ropOrder, 12345L);
```

Поиск по мнемокоду

Часто нужно получить ID по коду (например, код способа доставки):

```
// Получаем ID способа доставки "ЖД"
var idvZHD = Bs_DeliveryMethodApi.findByMnemoCode('ZHD');
```

Внимание

код чувствителен к регистру! „ZHD“ и „zhd“ — разные коды.

Пример полного скрипта:

```
// 1. Получаем ID из текущей выборки
var idvOrder = selection.getVar("id");

// 2. Загружаем строку заказа
var ropOrder = Stm_OrderOutApi.load(idvOrder);

// 3. Получаем атрибуты
var svNumDoc = ropOrder.sNumDoc; // Номер
var idvStationEndJson = Stm_OrderOutApi.getObjAttrValue(ropOrder, 'idStationEndJson');

// 4. Меняем номер
Stm_OrderOutApi.setsNumDoc(ropOrder, "НОВЫЙ-002");

// 5. Ищем ID способа доставки
var idvDelivery = Bs_DeliveryMethodApi.findByMnemoCode('ZHD');
```

Доступ к универсальным характеристикам

Помимо объектных характеристик, к классам могут быть подключены универсальные характеристики (UC). Доступ к ним осуществляется через `ClassApi`:

- `ClassApi.getUniCharValue(rop, "CharName")` — получение по системному имени.
- `ClassApi.getUniCharValue(rop, idAttr)` — получение по ID атрибута.

Примечание

Значения UC также могут быть `null`. Перед использованием применяйте проверку `isNotNull()` или `.nvl()`.

5.3 Преобразование типов данных

Метод	Пример использования
`.asJLong()`	`"123".asJLong()`
`.toLong()`	`"456".toLong()`
`.toInteger()`	`"789".toInteger()`
`.toDouble()`	`"12.34".toDouble()`
`.toBigDecimal()`	`"123.45".toBigDecimal()`
`.asDate()`	`"2024-01-01".asDate()`
`.toDate("формат")`	`toDate("01.01.24", "dd.MM.yy")`
`.toString()`	`123.toString()`
`.asString()`	`321.asString()`
`.asList()`	`sql("SELECT...").asList()`
`.asSingle()`	`sql("SELECT...").asSingle()`
`.asScala()`	`[1,2,3].asScala()`
`.asJava()`	`scalaSeq.asJava()`
`.toJObject()`	`toJObject('{"id":1}')`

5.4 Работа с датами

Метод	Описание	Пример
getFirstDay	Возвращает дату начала месяца от текущей даты	`var a = getFirstDayOfMonth(); var b = getFirstDayOfMonth('27.01.2025 12:42:10');`
getFirstDay	Возвращает первый день указанного периода	`var a = getFirstDayOfPeriod(sysDate(), 'M'); var b = getFirstDayOfPeriod('27.01.2025 12:42:10', 'Q');`
getLastDayO	Возвращает дату конца месяца от текущей даты	`var a = getLastDayOfMonth(); var b = getLastDayOfMonth('27.01.2025 12:42:10');`
getLastDayO	Возвращает последний день указанного периода	`var a = getLastDayOfPeriod(sysDate(), 'M'); var b = getLastDayOfPeriod('27.01.2025 12:42:10', 'Q');`
nanoTime	Получить значение System.nanoTime()	`var a = nanoTime();`
sysDate	Возвращает текущую дату	`var a = sysDate();`
toDate	Переводит строку в дату по указанному формату	`var a = toDate("27.02.2020 12:42:10"); var b = toDate("27.02.2020 12:42:10", "dd.MM.yyyy HH:mm:ss");`
truncDate	Получение даты на начало дня от переданной	`var a = truncDate(sysDate());`
truncYear	Получение даты на начало года от переданной	`var a = truncYear(sysDate());`
-	Вычитает из даты указанное количество дней	`var dYesterday = sysDate() - 1;`
+	Добавляет к дате указанное количество дней	`var dTomorrow = sysDate() + 1;`

5.5 Работа в контексте выборки

Контекст выборки (JexlSelScript) расширяет контекст бизнес-логики возможностью работы с выборками и пользовательским интерфейсом. Используется в тех случаях, когда необходимо получить данные из полей для пользовательского ввода.

Метод	Описание	Пример использования
`varExists(name)`	Проверяет существование переменной в текущей и мастер-выборках	`if (varExists("client_id")) { ... }`
`selfVarExists(name)`	Проверяет существование переменной только в текущей выборке	`if (selection.selfVarExists("temp_value")) { ... }`
`getVar(name)`	Получает значение переменной из текущей или мастер-выборки	`var id = selection.getVar("doc_id").asLong();`
`getSelfVar(name)`	Получает значение только из текущей выборки	`var code = selection.getSelfVar("product_code");`
`setVar(name, value)`	Устанавливает значение переменной (ищет в иерархии выборок)	`selection.setVar("status", "approved");`
`setSelfVar(name, value)`	Устанавливает значение только в текущей выборке	`selection.setSelfVar("comment", "Не заполнено");`
`addVar(name, value, FieldType)`	Добавляет новую переменную в текущую выборку	`selection.addVar("id", 1, FieldType.ftFloat());`
`newForm()`	Создает новую форму	`Act_DocEdit.newForm().params(...).open();`
`.form()`	Возвращает форму текущего контекста	`selection.form();`
`.opers('Операция')`	Возвращает указанную операцию для дальнейшего выполнения	`selection.opers("CREATESTMORDERIN");`
`.execute()`	Выполняет ранее полученную операцию	`selection.opers("CREATESTMORDERIN").execute();`

Работа с формой выборки

Создание и настройка формы:

```
var res = Btk_ClassAvi // Класс, предоставляющий форму выбора
.list() // Инициализация формы списка (табличное отображение)
.newForm() // Создание новой формы
.params({ // Передача параметров в форму в виде мапы
    "one": 1,
    "two": 2,
    "three": 3,
    "more": "many more"
})
.locates({"id": 146800}) // Выбор строки
.results(["id", "idClass"]) // Какие поля вернуть после выбора
.openLookup(); // Открытие окна
```

Метод	Описание
.list()	Форма будет отображаться как таблица (список записей). Возможные отображения можно посмотреть в обозревателе проекта в классе с расширением Avi
.newForm()	Создает новую формы
.params()	Передает параметры в форму (могут использоваться для фильтрации/логики)
.locates()	Установка строки на активную
.results()	Задает поля, которые вернутся после выбора
.openLookup()	Открывает окно выбора
.findSelection('O	Ищет и возвращает **выборку**, содержащую указанное отображение
.baseRep()	Возвращает ссылку на объект, к которому был вызван метод

Пример: Поиск и выполнение операций в связанной выборке

Демонстрирует, как найти выборку, связанную с конкретным документом, и выполнить в ней операцию.

```
var sel = selection.form() // получает форму по выделенному элементу
.findSelection(Stm_OrderOutAvi.Card() // Ищет и возвращает **выборку**, содержащую␣
↪указанное отображение
.baseRep()); // возвращает базовое представление объекта (ссылку)

if (sel != null) { // если выборка найдена, то выполняет действие
    sel.opers("CREATESTMORDERIN").execute(); // выполняет переданную операцию
}
```

Для работы с текущей выборкой можно использовать сокращенный синтаксис:

```
selection.opers("CREATESTMORDERIN").execute();
```

Обработка результата:

```
if (res.getLookupResult() == LookupResult.ok()) {
    // Пользователь выбрал значение и нажал "OK"
    for(var i: 1 .. res.size()) { // Цикл по всем выбранным записям
        dialogs.showMessage("id " + res.getData(i, 0) + " idClass " + res.getData(i, 1));
    }
} else {
    // Пользователь закрыл форму или нажал "Отмена"
    dialogs.showMessage("Значение не было выбрано");
}
```

- res.getLookupResult() — проверяет, как закрыли форму:
  - LookupResult.ok() — нажали «Выбрать»;
  - LookupResult.cancel() — нажали «Отмена» или закрыли окно.
- res.size() — количество выбранных записей (если форма поддерживает множественный выбор).
- res.getData(i, 0) — доступ к данным:
  - i — номер строки (начинается с 1);
  - 0 — индекс поля (соответствует порядку в .results([«id», «idClass»])).

Пример работы с мастер-выборками:

```
// Получаем значение из атрибута "id" выборки, приводим его к типу NString и пишем в␣
↪переменную idTree
var idTree = getVar("id").asNString();

// Получаем атрибут DGLOBALENDDATE из мастер-выборки
var dEndDate = getVar("super$DGLOBALENDDATE").asNDate();

// Вызываем метод из пакета с передачей параметров
// Обратите внимание, для обращения к Api или пакетам используются их короткие имена␣
↪(без скобок)
var fltCond = Act_UniversalReportPkg.getUniFilterCondByIdTree(idTree, dEndDate);

// Вызывается ещё один метод, возвращающий объект scala-класса immutable.Map[NString,␣
↪Any]
var filters = Act_UniversalReportPkg.getFilterValues(idTree);

/* Определение Map-ы внутри jexl-скрипта. Отметим, что Map внутри jexl и объект scala-
↪класса
Map (неважно mutable или immutable) - это разные объекты. Наиболее важным
фактом является то, что передать scala-объект Map, полученный в предыдущей
строке,
в пакетный метод, принимающий scala-объект Map, в jexl-скрипте напрямую нельзя,
именно поэтому приходится получать значения из переменной filters,
перезаписывать
их в jexl-Map param и потом передавать param в scala-метод. */

var param = {
    "flt_idDepOwner" : filters['flt_idDepOwner'],
    "flt_idAcc" : filters['flt_idAcc'],
    "flt_dFrom" : filters['flt_dFrom'],
    "flt_dTo" : filters['flt_dTo'],
    "flt_idAdjustMethod" : filters['flt_idAdjustMethod'],
    "flt_idAccCor" : filters['flt_idAccCor'],
    "uniFilterCondition_dz": fltCond
};

// Открываем новую форму с переданными параметрами
Act_TransAvi.defList().newForm().params(param).open();
```

Мастер-выборки:

- Доступ через префикс `super$`.
- Пример: `getVar("super$PARENT_ID")`.

Типизация:

- Используйте `.asString()`, `.asDate()` для явного преобразования.
- Для чисел: `.asBigDecimal()`, `.asJLong()`.

Дополнительные методы работы с формой выборки

Открывает мастер пользовательского ввода по указанному идентификатору формы:

```
Btk_WizardLib().launch(getSelfVar("id").asNLong); // В качестве параметра передается id␣
↪формы.
```

5.6 Работа с SQL

Для работы с SQL используются команды:

1. Для запросов на чтение:

```
sql(sqlText:String)
```

2. Для запросов на запись:

```
tsql(sqlText:String).execute()
```

Методы обработки для sql:

Метод	Описание	Пример
.asList()	Вернуть список записей	sql("SELECT * FROM users").asList()
.asSingle()	Вернуть одну запись	sql("SELECT * FROM users WHERE id=1").asSingle()

Методы обработки для tSql:

Метод	Описание	Пример
.execute()	Выполнить запрос	tsql("DELETE FROM temp").execute()

Примеры:

Простые запросы:

```
// Чтение
var activeUsers = sql("SELECT name FROM users WHERE is_active=true").asList();

// Запись
tsql("UPDATE orders SET status='done' WHERE id=42").execute();
```

Обработка результатов:

```
// Вывод имен пользователей
sql("SELECT name FROM users").foreach(row -> {
    logInfo("User: " + row.name);
});

// Подсчет суммы
var total = 0;
sql("SELECT amount FROM payments").foreach(p -> {
    total = total + p.amount;
});
```

Параметризация:

```
var productId = selection.getVar('id');
var product = sql("SELECT * FROM products WHERE id=${productId}").asSingle();

var productId = selection.getVar('id');
var sSqlText = ''; // объявляем переменную, в которой будем собирать sql-выражение
sSqlText = "SELECT * FROM products WHERE id="; //
sSqlText += toString(productId);
sSqlText += "\r\n order by id";
var product = sql(sSqlText).asSingle();

var productId = selection.getVar('id');
var product = sql(`
    SELECT *
    FROM products
    WHERE id=${productId}`)
.asSingle();
```

Особенности работы с JexlSqlRow

Результат `sql().asList()` возвращает список объектов типа `JexlSqlRow`.

Доступ к данным осуществляется следующими способами:

- `row.fieldName` — по имени колонки из `SELECT`;
- `row.getValue("fieldName")` — безопасный доступ (рекомендуется при динамических именах);
- `row.containsKey("fieldName")` — проверка наличия поля перед обращением;

Примечание

Поля в `JexlSqlRow` формируются строго из запроса. Обращение к несуществующему полю или к полю со значением `NULL` через точечную нотацию вызовет ошибку выполнения скрипта.

Поведение NULL при интерполяции

JEXL-интерполяция `${variable}` преобразует значение `null` в пустую строку `""`, а не в SQL-`NULL`. Выражение `WHERE id=${nullField}` сгенерирует `WHERE id=`, что вызовет синтаксическую ошибку PostgreSQL.

Перед подстановкой в SQL всегда проверяйте значение на `null`:

```
if (row.fieldName.isNotNull) {
sql("SELECT * FROM table WHERE col=${row.fieldName}").asList()
```

Либо используйте параметризованный запрос через `on(Symbol("param") -> value)`.

```
## Дополнительные методы

| Метод | Описание | Пример использования |
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
|-------|----------|---------------------|
| `lpad(str, len, ch)` | Дополнение строки слева | `lpad("5", 3, "0") → "005"` |
| `rpad(str, len, ch)` | Дополнение строки справа | `rpad("5", 3, "0") → "500"` |
| `flush()` | Синхронизация с БД | `flush()` |
| `commit()` | Подтверждение транзакции | `commit()` |
| `nvl(a, b)` | Возвращает a, если не null, иначе b | `nvl(var, "default")` |
| `isNull(x)` | Проверка на null | `isNull(obj)` |
| `isNotNull(x)` | Проверяет, что переданное значение не null | `isNotNull(obj)` |
| `pgArrayToNLongList` | Конвертация массива в список | `pgArrayToNLongList(arr)` |
| `parseId(gid: Object): Long` | Получение идентификатора объекта из NGid-а Работает для␣
↪строкового или NGid-значения | `var sGid = "14323/44334"; var id = parseId(sGid);` |
| `parseIdClass(gid: Object): Long` | Получение идентификатора класса из NGid-а Работает␣
↪для строкового или NGid-значения | `var sGid = "14323/44334"; var idClass =␣
↪parseIdClass(sGid);` |
| `toJRops(rops: Traversable[Rop]): JexlToJRops` | Обход записей, возвращенных объектным␣
↪запросом, например byParent Выполняет запрос на чтение к базе данных и позволяет␣
↪обойти записи | `var l = toJRops(Btk_SomeEntityApi.byParent(rop)).asList(); for(r:l){␣
↪logInfo(r.id); logInfo(r.sSystemName); }` |
| `toJObject(json: String): JexlToJObject` | Парсинг json-объекта | `var jData =␣
↪toJObject('{"id": 1}'); logInfo(jData.getLong("id"));` |

## Работа с BTS процедурами

**Переход в раздел процедур:**

Для того чтобы открыть перечень BTS-процедур, перейдите в **Настройки системы** →␣
↪**Настройки и сервисы** → **Процедуры**. Здесь можно просмотреть различные варианты␣
↪существующих процедур. Можно перейти в карточку любой процедуры и посмотреть, какие␣
↪действия там описаны.

Выберите в фильтре **Тип: Процедура для сбыта** и создайте новую тестовую процедуру,␣
↪нажав кнопку **Создать**. Введите наименование и код процедуры.

**Написание JEXL-кода:**

Предположим, мы хотим рассчитать некоторую формулу на основе входящих параметров. Чтобы␣
↪процедура вернула значение, объявим раздел с результатом и вернем соответствующее␣
↪значение. Объявим входные параметры и расчетную формулу:
```

var nvParam1 = npInParam1; var nvParam2 = npInParam2; var nvParam3 = npInParam3; var nvSum = (npInParam1 + npInParam2) * npInParam3;

// Для того чтобы процедура вернула значение, добавим возврат результата: return nvSum;

```
Сохраните код.

**Вызов процедуры через JEXL-редактор:**

Перейдите в отдельный редактор JEXL. Объявите служебную функцию для типизации данных.␣
↪Данная функция будет возвращать тип данных scala.Tuple2, состоящий из двух переменных.
```

var tuple2 = (v1, v2) -> { return new(scala.Tuple2, v1, v2); };

```
Заведите массив параметров, в котором запишите входящие значения. В нем будет объявлен␣
↪массив Scala (это делается за счет указания последнего элемента с троеточием). Первые␣
↪три элемента будут объявлены с помощью функции tuple2. В каждом таком элементе массива␣
↪будет указано название параметра и его значение.
```

var tuple2Params = asScala([tuple2(«npInParam1», 10l), tuple2(«npInParam2», 125l), tuple2(«npInParam3», 1000l), ...]);

```
Объявите переменную, в которую при помощи API-функции findByMnemoCode поместите␣
↪идентификатор ранее созданной процедуры:
```

val idvBtsProcedure = Bts_ProcedureApi.findByMnemoCode(Sale_Test1);

```
Далее вызовите конструкцию для расчета значения BTS-процедуры и запишите результат в␣
↪переменную. Для этого используйте API-метод Bts_ProcedureLib, передав в него:
- идентификатор процедуры;
- массив параметров;
- дополнительный контекст выборки для расчета.
```

var nvSumCalc = Bts_ProcedureLib.execById(idvBtsProcedure, tuple2Params, selection.coreRep());

```
Метод `Bts_ProcedureLib.execById` ожидает параметры в формате, который понимает Scala,␣
↪поэтому мы используем кортеж. Кортежи здесь - это "мост" между JEXL и Scala,␣
↪обеспечивающий четкую, безопасную и удобную передачу параметров.

Верните результат процедуры и выведите его на экран.
```

dialogs.showMessage(toString(nvSumCalc));

```
Аналогичным образом можно создавать и более сложные BTS-процедуры.

## Работа с мониторингом сессий сервера приложений

Система автоматически ведет метаинформацию о сессиях всех пользователей. Доступ к этой␣
↪информации организован следующим образом:

**Настройки системы** → **Сервис** → **Инструменты** → **Мониторинг сессий сервера␣
↪приложений**.

В списке отображаются все активные сессии с указанием последнего выполненного действия␣
↪для каждого пользователя.

Для работы с сессионными данными текущего пользователя предусмотрены следующие методы:

Получение текущего действия:
```

session.applicationInfo().action();

```
Обновление информации о действии:
```

session.applicationInfo().action_$eq(«Значение текущего действия»);

```
Пример использования:
```

for (ropWW: ropaIntWar) { session.applicationInfo().action_$eq(Phosagro_SB0205_NEW - ВП - ${toString(ropWW.gid)}); Stk_InternalWarrantApi.setdExecDateOut(ropWW, cDate); session.applicationInfo().action_$eq(Phosagro_SB0205_NEW - выполняем ВП - ${toString(ropWW.gid)}); Stk_InternalWarrantApi.setidStateOut(ropWW, idvStateIntWar); session.applicationInfo().action_$eq(Phosagro_SB0205_NEW - Создаем PCO); var ropWarOut = Stk_WarrantOutApi.createWarrantOutByInternalWarrant(ropWW); // Заполняем даты исполнения Stk_WarrantOutApi.setdDocDate(ropWarOut, cDate); Stk_WarrantOutApi.setdExecDate(ropWarOut, cDate); // Выполняем PCO session.applicationInfo().action_$eq(Phosagro_SB0205_NEW - выполняем ВП - ${toString(ropWarOut.gid)}); Stk_WarrantOutApi.setidState(ropWarOut, idvStateWarr); };

```
**Метод setUniCharValue** используется для установки значения универсальной␣
↪характеристики (универсального атрибута) объекта в системе.
```

setUniCharValue(rop: ApiRop, idpUniChar: Long, pValue: Any): Unit

```
| Параметр | Тип (Scala) | Описание |
|----------|-------------|----------|
| rop | Rop | Контекст выполнения операции (объект, к которому применяется изменение) |
| idpUniChar | Long | Идентификатор универсальной характеристики |
| pValue | Any | Новое значение характеристики (может быть Long, String и др) |

Пример использования:
```

// 1. Получаем GID значения «.» из справочника val gidChr = sql(«» SELECT r.gid FROM Btk_UniversalReference r JOIN btk_objecttype b ON r.idobjecttype = b.id WHERE r.sSystemname = „“ AND b.scode = „ur_O2C_sort_type“ «»).asSingle();

Stm_OrderOutApi.setUniCharValue(rop, „O2C_sort_type“, gidChr.gid);

```
## Работа с печатью

Пользователь может:
- Сформировать отчёт по его системному имени.
- Запустить JEXL-скрипт печатной формы, привязанной к типу объекта.

Создание отчета происходит с помощью следующего метода:
```

Rpt_Lib.createReportExJexl(reportName: String, reportVersionDate: Date, postBuildAction: PostBuildAction, propertyMap: Map[String, Any], idpObjectTypePrintForm: NLong = None.nl): Unit

```
| Параметр | Тип (Scala) | Описание |
|----------|-------------|----------|
| idpObjectTypePrintForm | Long | id ПФ на типе объекта. Может быть null |
| reportName | String | Имя отчета |
| reportVersionDate | Date | Дата |
| postBuildAction | PostBuildAction | Действие, которое необходимо произвести после␣
↪заполнения отчёта |
| propertyMap | Map[String, Any] | Карта входящих параметров |

Пример использования:
```

var idvOrderClass = selection.getVar(«idClass»); var idvOrder = selection.getVar(«id»);

var NLong = function(number) { // локальная функция для преобразования Long в NLong return new («ru.bitec.app.gtk.lang.NLong», number); };

var spReportName = «Stm_InvoiceOut»; var dpReportVersionDate = sysDate(); var vPostBuildAction = session .sbtClassLoader() .loadClass(„ru.bitec.app.gtk.gl.postbuildaction.PostBuildAction“) .design();

var propertyMap = { «idSrcObject» : NLong(idvOrder), «idSrcClass» : NLong(idvOrderClass) };

var idpObjectTypePrintForm = null;

Rpt_Lib.createReportExJexl(spReportName, dpReportVersionDate, vPostBuildAction, asScala(propertyMap), idpObjectTypePrintForm);

```
**Тип PostBuildAction:**

PostBuildAction - это перечисление (enum) из класса ru.bitec.app.gtk.gl.postbuildaction.
↪PostBuildAction, определяющее действия после построения отчета.

**Доступные значения:**
- `.show()` - загружает отчет на клиент;
- `.save()` - сохраняет отчет;
- `.print()` - загружает отчет на клиент и, если в конфигурации системы включена опция␣
↪печати (Configuration.Printing.enableNetworkPrinting), печатает отчет на локальном␣
↪(для сервера) принтере.

**Примечание:**
Для `.print()` поведение зависит от конфигурации сервера. Если печать не настроена,␣
↪отчет просто загружается на клиент.

**Особенности:**
- Для работы с ID используется тип NLong (внутренний класс системы).
- asScala() преобразует Java-коллекции в Scala-совместимые.
- Если idpObjectTypePrintForm = null - отчёт строится по reportName.
```

6 Практические примеры

В этом разделе рассмотрены два практических сценария: автоматическое создание пунктов маршрута для складов и обработка перехода состояний документов с валидацией бизнес-правил.

6.1 Создание пункта маршрута для каждого склада

1. SQL-запрос для получения списка складов. Получим ID и коды складов, у которых тип объекта «Склад» (stock) или «Склад в пути» (StockInWay). Полученные записи преобразуем в список с помощью метода `asList()`.

```
var sqlStock = sql(`
    SELECT stk.id as id, stk.sCode as sCode FROM stk_stock stk
    JOIN btk_objecttype ot ON ot.id = stk.idObjectType
    where ot.sCode = 'stock' OR ot.sCode = 'StockInWay'
`).asList()
```

2. Поиск пункта маршрута по коду склада. С помощью цикла `for` сделаем преобразование над полученными записями. По коду нашего склада получим id пункта маршрута.

```
var sqlStock = sql(`
    SELECT stk.id as id, stk.sCode as sCode FROM stk_stock stk
    JOIN btk_objecttype ot ON ot.id = stk.idObjectType
    where ot.sCode = 'stock' OR ot.sCode = 'StockInWay'
`).asList()

for(r: sqlStock) {
    var idvPoint = Tms_PointApi.findByMnemoCode(r.scode);
}
```

3. Создание нового пункта маршрута (если не найден). Если вернувшееся значение пункта маршрута равно null, то нужно создать его, заполнив код, название и привязку текущего склада.

```
var sqlStock = sql(`
    SELECT stk.id as id, stk.sCode as sCode FROM stk_stock stk
    JOIN btk_objecttype ot ON ot.id = stk.idObjectType
    where ot.sCode = 'stock' OR ot.sCode = 'StockInWay'
`).asList()

for(r: sqlStock) {
    var idvPoint = Tms_PointApi.findByMnemoCode(r.scode);
    if (isNull(idvPoint)) {
        var ropPoint = Tms_PointApi.insert();
        Tms_PointApi.setsCode(ropPoint, r.scode);
        Tms_PointApi.setsCaption(ropPoint, r.scode);
        Tms_PointApi.setidStock(ropPoint, r.id);
    }
}
```

4. Обновление существующего пункта маршрута (если не привязан к складу). Если полученный пункт маршрута существует, то загружаем его по полученному ранее id. Далее проверяем у полученного пункта маршрута привязку к складу, если значение равно null, то устанавливаем значение текущего склада.

```
var sqlStock = sql(`
    SELECT stk.id as id, stk.sCode as sCode FROM stk_stock stk
    JOIN btk_objecttype ot ON ot.id = stk.idObjectType
    where ot.sCode = 'stock' OR ot.sCode = 'StockInWay'
`).asList()

for(r: sqlStock) {
    var idvPoint = Tms_PointApi.findByMnemoCode(r.scode);
    if (isNull(idvPoint)) {
        var ropPoint = Tms_PointApi.insert();
        Tms_PointApi.setsCode(ropPoint, r.scode);
        Tms_PointApi.setsCaption(ropPoint, r.scode);
        Tms_PointApi.setidStock(ropPoint, r.id);
    } else {
        var ropPointW = Tms_PointApi.load(idvPoint);
        if (isNull(ropPointW.idStock)) {
            Tms_PointApi.setidStock(ropPointW, r.id);
        }
    }
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
}
```

6.2 Пример расширения обработки состояния документа

1. Проверка на пустое старое значение:

```
if(isNull(oldValue)) return 0;
```

Если текущее состояние документа `oldValue` не задано, обработка прекращается (возвращается 0).

2. Получение порядковых номеров состояний:

```
var idStateFromOrder = Btk_ClassStateApi.getOrder(oldValue);
var idStateToOrder = Btk_ClassStateApi.getOrder(value);
```

- idStateFromOrder - порядковый номер старого состояния документа.
- idStateToOrder - порядковый номер нового состояния.

Получим текущее состояние документа, воспользовавшись API-методом класса состояния.

3. Проверка условий для обработки:

```
if(idStateFromOrder <= 100 && idStateToOrder > 100 && rop.idObjectType == Btk_
↪ObjectTypeApi.findByMnemoCode("Tms_DeliveryAct"))
```

- `idStateFromOrder <= 100` - проверяет, что текущее состояние документа имеет порядковый номер $<= 100$. Обычно это соответствует статусам «Аннулировано» или «Формируется».
- `idStateToOrder > idStateFromOrder` - гарантирует, что новое состояние действительно является переходом ВВЕРХ по workflow. Исключает случаи, когда статус меняется на равный или более ранний.
- `rop.idObjectType == Btk_ObjectTypeApi.findByMnemoCode("Tms_DeliveryAct")` - убеждается, что работаем именно с документом типа «Рейс» (Tms_DeliveryAct).

4. Получение заказа (Stm_OrderOut):

```
var ropOO = Stm_OrderOutApi.loadByGid(rop.gidSrc);
if(ropOO.idObjectType != Btk_ObjectTypeApi.findByMnemoCode("OrderOut_Service"))
```

Получаем ропу заказа с помощью Api-метода loadByGid по gidSrc(родитель) текущего документа. Если тип заказа **НЕ** «OrderOut_Service» (то есть это не сервисный заказ), выполняется дальнейшая проверка.

5. Определение договора:

```
var ropContract = null;
if(isNotNull(rop.idAddContract))
    ropContract = Cnt_ContractApi.load(rop.idAddContract);
else
    ropContract = Cnt_ContractApi.load(rop.idContract);
```

Если у заказа `rop` есть доп. договор `idAddContract`, он загружается. Иначе загружается основной договор (idContract).

6. Проверка отложенного ценообразования:

```
var bDeferredPricing = Cnt_ContractApi.getObjAttrValue(ropContract, "bDeferredPricing");
if(bDeferredPricing == true)
```

Получаем значение атрибута bDeferredPricing (флаг «Отложенное ценообразование»). Если флаг true, выполняется проверка даты.

7. Проверка даты отложенного ценообразования:

```
var dDeferredPricing = Stm_ActOutApi.getObjAttrValue(rop, "dDeferredPricing");
if(isNull(dDeferredPricing) || dDeferredPricing < sysDate())
    raise("Требуется выполнить корректировку цены");
```

Получаем дату dDeferredPricing (когда должна быть выполнена корректировка цены). Если дата **не задана** или **уже прошла**, выводится ошибка.

Общий код:

```
if(isNull(oldValue))return 0;
var idStateFromOrder = Btk_ClassStateApi.getOrder(oldValue)
var idStateToOrder = Btk_ClassStateApi.getOrder(value);

if(idStateFromOrder <= 100 && idStateToOrder > 100 && rop.idObjectType == Btk_
↪ObjectTypeApi.findByMnemoCode("Tms_DeliveryAct")){
    var ropOO = Stm_OrderOutApi.loadByGid(rop.gidSrc)
    if(ropOO.idObjectType != Btk_ObjectTypeApi.findByMnemoCode("OrderOut_Service")){
        var ropContract = null
        if(isNotNull(rop.idAddContract))
            ropContract = Cnt_ContractApi.load(rop.idAddContract)
        else
            ropContract = Cnt_ContractApi.load(rop.idContract)
        var bDeferredPricing = Cnt_ContractApi.getObjAttrValue(ropContract,
↪"bDeferredPricing")
        if(bDeferredPricing == true){
            var dDeferredPricing = Stm_ActOutApi.getObjAttrValue(rop, "dDeferredPricing")
            if(isNull(dDeferredPricing) || dDeferredPricing < sysDate()){
                raise("Требуется выполнить корректировку цены")
            }
        }
    }
}
```

7 Справочник Jexl скриптов

7.1 Введение

Этот справочник содержит образцы JEXL-скриптов, применяемые для автоматизации, настройки и диагностики в системе.
Скрипты сгруппированы по тематическим разделам: работа с документами, печатными формами, маршрутами согласования, управление потребностями, распределение затрат, настройка администрируемых объектов и другие области.

Примеры предоставляются как **шаблоны для адаптации** под конкретные задачи. Поскольку система развивается, а скрипты актуализируются вручную и по мере необходимости, их логика может

требовать корректировки при использовании в новых версиях платформы.
Рекомендуется рассматривать представленные решения как отправную точку для разработки собственных реализаций.

7.2 Как добавлять JEXL-скрипты в справочник

В статье описано, как добавлять новые JEXL-скрипты в справочник: как выбрать раздел, как оформить страницу раздела и как правильно оформить сам скрипт. Инструкция нужна, чтобы все страницы справочника выглядели одинаково, пополнялись по одному правилу и оставались удобными для поиска и использования.

Добавление и изменение скриптов выполняется через **merge request**.

Если у вас нет доступа к репозиторию, нет опыта работы с Git или вы не уверены, как правильно оформить изменения, обратитесь к **Гребневу Алексею**.

Общий принцип

Справочник JEXL-скриптов состоит из тематических разделов. Раздел — это отдельная страница, внутри которой собраны скрипты одной тематики.

Раздел обычно включает название, краткое описание и список скриптов. Скрипты внутри раздела идут последовательно одним списком.

Разделение на разделы нужно, чтобы не смешивать на одной странице разные сценарии, упростить поиск нужного примера и поддерживать единый понятный формат справочника.

Порядок добавления

1. Определите, подходит ли скрипт в один из текущих разделов.
2. Если подходит, добавьте его в этот раздел.
3. Если скрипт явно не подходит ни в один текущий раздел, создайте новый раздел.
4. Подготовьте описание скрипта.
5. Вставьте код скрипта в текст страницы.
6. При необходимости добавьте вспомогательные файлы.
7. Проверьте оформление.
8. Создайте merge request.

Разделы справочника

Как выбрать раздел

Перед добавлением скрипта определите, подходит ли он в один из текущих разделов.

Ориентируйтесь прежде всего на сценарий использования скрипта, а не на используемые классы, методы или API. Скрипт нужно размещать там, где его будет искать пользователь.

Если скрипт можно логично отнести к нескольким разделам, выберите тот, в котором его ожидаемо искать по задаче.

Что делать, если неясно, в какой раздел добавить скрипт

Если вы сомневаетесь, не создавайте новый раздел сразу. Укажите это в merge request и при необходимости предложите наиболее подходящий, по вашему мнению, вариант.

Когда создавать новый раздел

Новый раздел создавайте только тогда, когда вы однозначно понимаете, что скрипт не подходит ни в один текущий раздел.

Не создавайте новый раздел на всякий случай или просто потому, что сомневаетесь. Если скрипт можно логично разместить в существующем разделе, лучше добавить его туда.

Название раздела должно отражать задачу или сценарий использования, а не название класса, таблицы или API.

Пример корректного раздела

Для оформления используйте существующие страницы справочника как шаблон. Пример: *Администрирование*

Структура скрипта

Каждый скрипт размещается внутри раздела как отдельный подпункт с заголовком третьего уровня.

В составе скрипта указываются название, описание, место применения, тип и тело скрипта.

Описание должно кратко и понятно объяснять, что делает скрипт, где он применяется и, если это полезно, чем он может служить как пример. Не нужно пересказывать код построчно, писать слишком общие формулировки или добавлять лишнюю теорию.

Место применения указывается в формате пути: `Раздел > Подраздел > Действие`

Тип всегда указывается одинаково: `JEXL-скрипт`

Тело скрипта вставляется непосредственно в текст страницы и не прикладывается отдельным файлом.

Внутри самого скрипта желательны комментарии, поясняющие основные действия. Они помогают быстрее понять логику примера и снижают риск ошибок при адаптации скрипта.

Добавление вспомогательных файлов

Если для работы скрипта требуется дополнительный файл, например Excel-шаблон, его необходимо добавить в документацию.

Такие файлы не заменяют тело скрипта. Скрипт всегда вставляется в текст страницы, а вспомогательный файл добавляется отдельно.

Чтобы добавить файл, поместите его в папку `attach`, назовите латиницей без пробелов и добавьте ссылку в текст страницы.

Инструкция по добавлению файлов: Форматирование текста.

Итог

Каждый скрипт должен находиться в подходящем разделе, быть оформлен по единому формату, содержать код в тексте страницы и быть понятен без дополнительного контекста.

7.3 Автонумерация

Применение счетчика автонумерации

Используется для условного применения стандартного счетчика автонумерации. Скрипт позволяет отключить стандартную автоматическую нумерацию для отдельного типа документов и оставить ее для остальных типов.

Например, для типа **Переоценка кассы** используется собственная логика генерации номера, поэтому системный счетчик отключается. Для остальных типов документов счетчик применяется стандартно.

Место применения: `Настройки системы > Сущности > Классы > [Класс] > Атрибуты > [Атрибут типа AutoNum] > Настройки автонумерации > Поле "Маска"`

Тип: JEXL-скрипт

```
if (self.idObjectType() == Btk_ObjectTypeApi.findByMnemoCode('Pm_SumDiffCalcSheet_PayDesk
↪')) null else counter
```

7.4 Администрирование

Коррекция параметра правила дискретного доступа

Используется чтобы удалить некорректное значение из JSON-массива параметра правила дискретного доступа. Применяется при постобработке после миграции или ошибок настройки.

Место применения: `Приложение "Администратор" > Настройки > Администрируемые объекты > [Администрируемый объект] > Редактировать > Дискретные ограничения доступа`

Внимание

В текущем виде содержит захардкоженные мнемокоды и ID — перед использованием необходимо заменить на актуальные значения.

Тип: JEXL-скрипт

```
var idvAcObject = Btk_AcObjectApi.findByMnemoCode('Pm_AdvRep'); // Адм.объект
var idvAcRole = Btk_AcRoleApi.findByMnemoCode('tstsud8.Documents'); // Роль
var idvAcRoleObjRule = 306521; // Правило роли
var idvAcRoleObjRuleParam = 350521; // Параметры позиции правила роли

if (!idvAcObject) {
    dialogs.showMessage("Не найден адм.объект с мнемокодом 'Pm_AdvRep'");
}
if (!idvAcRole) {
    dialogs.showMessage("Не найдена роль с мнемокодом 'tstsud8.Documents'");
}
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
if (!idvAcRoleObjRule) {
    dialogs.showMessage("Не найдено правило роли с мнемокодом 'Pm_AdvRep34'");
}

var idvParams = sql(`
select
arorp.id as id
from btk_acroleobjrule aror
join btk_acroleobjruledet arord on arord.idacroleobjrule = aror.id
join btk_acobjectrule aor on arord.idacobjrule = aor.id
join btk_acroleobjruleparam arorp on arorp.idacroleobjruledet = arord.id
where
aor.idacobject = `+ idvAcObject +` -- id адм.объекта
and aror.idacrole = `+ idvAcRole +` -- id роли, для которой задан параметр правила
and aror.id = `+ idvAcRoleObjRule +` -- id правила для конкретной роли
and arorp.id = `+ idvAcRoleObjRuleParam +` -- id позиции правила для конкретной роли
;`).asSingle().id;

if(isNotNull(idvParams)) {
  const ropAcRoleObjRuleParam = Btk_AcRoleObjRuleParamApi.load(idvParams);
  var jPropertiesNew = [];
  var jProperties = Btk_AcRoleObjRuleParamApi.parsejParamValue(ropAcRoleObjRuleParam).
↪asJObject();

    var jsonBuilder = "";
    var first = true;

    for (element : jProperties) {
      value = element.asLong();
        if (value != 429) {
            if (!first) {
                jsonBuilder = jsonBuilder + ",";
            }
            jsonBuilder = jsonBuilder + value;
            first = false;
        }
    }

    var jPropertiesNewString = "[" + jsonBuilder + "]";

Btk_AcRoleObjRuleParamApi.setjParamValue(ropAcRoleObjRuleParam,␣
↪toString(jPropertiesNewString));
}
```

Удаление привилегий на уровне административного объекта

Используется чтобы массово удалить привилегии на указанный административный объект для всех ролей, а также пересчитать права пользователей. Применяется при отмене доступа к устаревшему функционалу.

Место применения: `Приложение "Администратор" > Настройки > Администрируемые объекты > [Администрируемый объект] > Редактировать > Дискретные ограничения доступа`

Внимание

В текущем виде содержит незавершённый массив `arr = [...]` и захардкоженный код объекта `Act_MainMenuAvi` — перед использованием необходимо завершить логику и указать корректные значения.

Тип: JEXL-скрипт

```
var arr = [...];

var ia = sql(`
  select opg.id, opg.idacrole  from Btk_AcRoleObjPrivTypeGrant opg
  join Btk_AcObject o on o.id = opg.idacobject
                    where opg.bhasaccess = 1 and o.scode = 'Act_MainMenuAvi'
`).asList();
for(rv:ia) {
if (!arr.contains(rv.idacrole)){
        arr.add(rv.idacrole);
}
                  var rr = Btk_AcRoleApi.load(rv.idacrole);
                  Btk_AcRoleApi.unSync(rr);
                  var rop = Btk_AcRoleObjPrivTypeGrantApi.load(rv.id);
                  Btk_AcRoleObjPrivTypeGrantApi.delete(rop);
}

commit();

Btk_AcRoleGrantRegApi.lazyUpdateAll();
commit();
for(id:arr) {
Btk_UserApi.recalcPrivsMulti(Btk_AcRoleApi.getUsersByRole(id));
}


commit();
```

Пример реализации скрипта дискретного доступа

Используется как шаблон для настройки сложных правил дискретного доступа по нескольким параметрам. Позволяет протестировать логику фильтрации и отладить условия доступа к записям.

Место применения: `Приложение "Администратор" > Настройки > Администрируемые объекты > [Администрируемый объект] > Редактировать > Дискретные ограничения доступа`

Тип: JEXL-скрипт

```
if (parseIdClass(row.gid) != Pm_AdvRepApi.idClass()) {
  return true;
}

// Инициализация переменных с значениями по умолчанию
var idvDoc = row.id;
var ropDoc = null;
var idvBisObj = null;
var idvObjectType = null;
var idvAdvRepItem = [];
var bvExpAlloc = null;
var idvExpAllocBisObj = null;

// Получение значений от проверяемого объекта для проверки на соответствие параметрам
// Загружаем данные только если idvDoc не null
if (idvDoc != null) {
      ropDoc = Pm_AdvRepApi.load(idvDoc);
      // Бизнес единица авансового отчета
      idvBisObj = Pm_AdvRepApi.getAttrValue(ropDoc, 'idBisObj');
      // Тип объекта авансового отчета
      idvObjectType = Pm_AdvRepApi.getAttrValue(ropDoc, 'idObjectType');
      // Элементы позиций авансового отчета
      idvAdvRepItem = sql(`
        select det.idadvrepitem as detitem
        from pm_advrep doc
        left join pm_advrepdet det on det.idadvrep = doc.id
        where doc.id = ${idvDoc}
      `).asList();
      // Признак передачи затрат
      bvExpAlloc = sql(`
        select
        coalesce(cast(doc.jobjattrs_dz ->> 'bExpAllocationSng' as int8), 0) as bexpalloc
        from pm_advrep doc
        where doc.id = ${idvDoc}
      `).asSingle();
      // Бизнес единица дебитора
      idvExpAllocBisObj = Pm_AdvRepApi.getObjAttrValue(ropDoc, 'idPassExpBisObj');
}

// Параметры дискретного доступа
// Значения параметров для отладки (без получения от системной настройки)
//var params = sql(`
//        select
//         cast(string_to_array(translate(jObj ->> 'idBisObj', '[]', ''), ',') as␣
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
↪bigint[]) as idBisObj
//        ,cast(string_to_array(translate(jObj ->> 'idObjectType', '[]', ''), ',') as␣
↪bigint[]) as idObjectType
//        ,cast(string_to_array(translate(jObj ->> 'idAdvRepItem', '[]', ''), ',') as␣
↪bigint[]) as idAdvRepItem
//        ,cast(string_to_array(translate(jObj ->> 'idExpAllocBisObj', '[]', ''), ',')␣
↪as bigint[]) as idExpAllocBisObj
//        from json_object('{"idBisObj", "2", "idObjectType", "24302", "idAdvRepItem", "
↪", "idExpAllocBisObj", "2"}') as jObj
//      ;`).asSingle();
for (p: params) {
  var jObj = toJObject(p);
  // Инициализация переменных для хранения значений параметров дискретного доступа
  var paramBisObj = null;
  var paramObjectType = null;
  var paramAdvRepItem = null;
  var paramExpAllocBisObj = null;

  // Значения результата проверки для случая, когда параметр дискретного доступа не задан
  var resBisObj = true;
  var resObjectType = true;
  var resAdvRepItem = false;
  var resExpAlloc = false;
  var resExpAllocBisObj = true;

// Проверки параметров дискретного доступа
// Проверка бизнес единицы
    if ((idvBisObj == null || jObj.contains("idBisObj"))) {
           paramBisObj = jObj.childJArray("idBisObj");
           if (paramBisObj.size() > 0) {
               resBisObj = false;
               var i = 0;
               while (i < paramBisObj.size() && !resBisObj) {
                      var value = paramBisObj.getLong(i);
                      if (value != null) {
                          if (idvBisObj == null || idvBisObj.toString() == value.
↪toString()) {
                          resBisObj = true;
                            }
                          }
                      i = i + 1;
                      }
            }
        }

// Проверка типа объекта
    if ((idvObjectType == null || jObj.contains("idObjectType"))) {
           paramObjectType = jObj.childJArray("idObjectType");
           if (paramObjectType.size() > 0) {
               resObjectType = false;
               var i = 0;
               while (i < paramObjectType.size() && !resObjectType) {
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                    var value = paramObjectType.getLong(i);
                    if (value != null) {
                        if (idvObjectType == null || idvObjectType.toString() == value.
↪toString()) {
                        resObjectType = true;
                          }
                        }
                    i = i + 1;
                    }
            }
          }

// Проверка элемента
    if ((idvAdvRepItem == null || jObj.contains("idAdvRepItem"))) {
           paramAdvRepItem = jObj.childJArray("idAdvRepItem");
           if (paramAdvRepItem.size() > 0) {
              resAdvRepItem = false;
              var i = 0;
              while (i < paramAdvRepItem.size() && !resAdvRepItem) {
                  var value = paramAdvRepItem.getLong(i);
                  if (value != null) {
                    for(r:idvAdvRepItem){
                          if (r.detitem != null and r.detitem.toString() == value.
↪toString()) {
                            resAdvRepItem = true;
                          }
                    }
                  }
                  i = i + 1;
              }
            }
    }

// Проверка признака передачи затрат
    if (bvExpAlloc.bexpalloc != null && bvExpAlloc.bexpalloc != 0) {
        resExpAlloc = true;
     }

// Проверка бизнес единицы дебитора
  if ((idvExpAllocBisObj != null && jObj.contains("idExpAllocBisObj"))) {
           paramExpAllocBisObj = jObj.childJArray("idExpAllocBisObj");
           if (paramExpAllocBisObj.size() > 0) {
               resExpAllocBisObj = false;
               var i = 0;
               while (i < paramExpAllocBisObj.size() && !resExpAllocBisObj) {
                      var value = paramExpAllocBisObj.getLong(i);
                      if (value != null) {
                          if (idvExpAllocBisObj.toString() == value.toString()) {
                             resExpAllocBisObj = true;
                            }
                          }
                      i = i + 1;
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                    }
                }
            }
// Итоговая проверка параметров
// raise('resBisObj: ' + toString(resBisObj) + ' ' + 'resObjectType: ' +␣
↪toString(resObjectType) + ' ' + 'resAdvRepItem: ' + toString(resAdvRepItem) + ' ' +
↪'resExpAlloc: ' + toString(resExpAlloc) + ' ' + 'resExpAllocBisObj: ' +␣
↪toString(resExpAllocBisObj));
  if (resBisObj == true and resObjectType == true and resAdvRepItem == false) {
    return true;
  }
  else if (resExpAllocBisObj == true and resExpAlloc == true and resObjectType == true␣
↪and resAdvRepItem == false) {
    return true;
  }
  else {
    return false;
  }
}
```

Копирование переходов состояний с одного объекта на другой

Копирует переходы состояний с одного типа объекта на другой. Может использоваться при настройке или тиражировании конфигурации.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
var l = sql(`select s1.id idstatefrom, s2.id idstateto, s3.id idstate
from btk_statechange sc
join btk_state s1 on s1.id = sc.idstatefrom
join btk_state s2 on s2.id = sc.idstateto
join btk_state s3 on s3.id = sc.idstate
where s1.idobjecttype = {id_источника}
  and s2.idobjecttype = {id_источника}
  and s3.idobjecttype = {id_источника}`).asList();

for (w : l) {
  Btk_StateChangeApi.register({id_приемника}, w.idstatefrom, w.idstateto, w.idstate);
}
```

Создание записи о гражданстве физического лица

Создает запись о гражданстве для физического лица. Показывает пример создания дочерней записи через `insertByParent`, заполнения ссылки на страну и работы с датой через `toDate`.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
// Преобразование строки в дату
const dvBirthDay = toDate('20.05.1954', 'dd.MM.yyyy');

// Загрузка физического лица
var ropPerson = Bs_PersonApi.load(toLong(10903));

// Создание записи о гражданстве
var ropCit = Bs_PersonCitizenshipApi.insertByParent(ropPerson);

// Вывод gid созданной записи
dialogs.showMessage(toString(ropCit.gid));

// Заполнение даты начала действия гражданства
Bs_PersonCitizenshipApi.setdFrom(ropCit, dvBirthDay);

// Заполнение страны
Bs_PersonCitizenshipApi.setidCountry(ropCit, toLong(74));
```

Преобразование Scala-коллекции к Java-типу в JEXL

Показывает пример преобразования Scala-коллекции для корректной работы в JEXL. Используется, когда метод ожидает Scala-тип или при передаче коллекции между Scala-кодом и JEXL.

Место применения: `В теле JEXL-скрипта`

Тип: JEXL-скрипт

```
// Создание scala.collection.immutable.List из java.util.ArrayList.
// Методы в scala foo(idapSome: Seq[NLong]) с java коллекциями = ошибка

ZSngAsf_RsrvCalcDocApi.calcJexl(ropDoc, asScala(idavData));
```

Включение аудита для класса

Включает аудит для указанного класса системы. Может использоваться при диагностике, отладке или временном включении отслеживания изменений для объектов определенного типа.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
// id класса, для которого необходимо включить аудит
var idClass = 12345L;

// загрузка класса
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
var ropClass = Btk_ClassApi.load(idClass);

// включение аудита
Btk_ClassApi.setbAudit(ropClass, 1B);

dialogs.showMessage("Аудит включен");
```

Получение id состояния класса по системному имени

Показывает, как получить `id` состояния класса по его системному имени. Шаблон можно использовать в любом JEXL-скрипте, где нужно получить идентификатор состояния через API-класс объекта.

Место применения: `В теле JEXL-скрипта`

Тип: JEXL-скрипт

```
const idvNewDocState = Stk_WarrantOutApi.idv<тут системное имя состояния>();

// Пример: const idvNewDocState = Stk_WarrantOutApi.idvDone();
```

Проверка заполнения характеристик объекта

Проверяет, что для объекта заполнены все характеристики, настроенные на его типе объекта. Если хотя бы одна характеристика не заполнена, смена состояния прерывается с сообщением об ошибке.

Место применения: `Bts-процедура при переводе состояния`

Тип: JEXL-скрипт

```
var ropaObjAttr = Btk_ObjectAttrByObjectTypeApi.idxObjectType().byKey(new ru.bitec.app.
↪gtk.lang.NLong(rop.idObjectType));

for (ropObjAttr : ropaObjAttr) {
  // Проверка универсальной характеристики
  if (isNotNull(ropObjAttr.idUniversalCharacteristic) && Bs_OrgCheckSessionApi.
↪getUniCharValue(rop, ropObjAttr.idUniversalCharacteristic) == null) {
    raise("Запрещена смена состояния. Заполнены не все атрибуты характеристик");
  }

  // Проверка обычного атрибута характеристики
  if (isNotNull(ropObjAttr.idAttribute) && Bs_OrgCheckSessionApi.getAttrValue(rop,␣
↪ropObjAttr.idAttribute) == null) {
    raise("Запрещена смена состояния. Заполнены не все атрибуты характеристик");
  }
}
```

Mctdocrls

Используется для административных действий: проверки прав, настройки ограничений доступа, очистки служебных заданий или работы с административными объектами.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
var idvUser = Btk_UserApi.getCurrentUserID();
var bvHasRights = sql(`
        with rights as (
                SELECT rights.*
                FROM Btk_RlsUserRightsFlat as rights
                where rights.gidObj = '`+ rop.gid + `'
        )
        select max(b) as "bHasRights"
        from (
                select
                        coalesce(max(1), 0) as b
                from rights t
                where t.idUser = `+ idvUser + `
                union all
                select
                        coalesce(max(0), 1) as b
                from rights t
        ) t
`).asSingle().bHasRights;
return bvHasRights == 1B;
```

7.5 Документы

Формирование пути к папке документа

Используется чтобы автоматически организовывать документы в навигационной структуре системы: создаёт папки на основе параметров документа, например бизнес-объекта, года и месяца, и помещает документ в соответствующую папку. Месяцы отображаются в русскоязычной форме, например `01 ЯНВАРЬ`.

Место применения: `Настройки системы > Сущности > Классы > Wf_Doc > Тип объекта > [Тип объекта] > Раздел настройки > Настройка документа Wf > Вкладка "Настройки" > Поле "Путь к папке"`

Тип: JEXL-скрипт

```
var gid = Wf_DocApi.getGidLastVer(rop.id);
var ropEntityExec = Rpt_EntityExecApi.getByGidDocVer(gid);
var jParams = toJObject(ropEntityExec.jParams);
var bo = Bs_BisObjApi.findByMnemoCode(jParams.getString("flt_idbisobjmc"));
var idObejctType = Btk_ObjectTypeApi.findByMnemoCode("dirBE");
var sMonth = Clr_MonthApi.getMnemoCode(jParams.getLong("flt_dperiodmonth"));

if (sMonth == '01') {sMonth = sMonth + ' ЯНВАРЬ';}
else if (sMonth == '02') {sMonth = sMonth + ' ФЕВРАЛЬ';}
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
else if (sMonth == '03') {sMonth = sMonth + ' МАРТ';}
else if (sMonth == '04') {sMonth = sMonth + ' АПРЕЛЬ';}
else if (sMonth == '05') {sMonth = sMonth + ' МАЙ';}
else if (sMonth == '06') {sMonth = sMonth + ' ИЮНЬ';}
else if (sMonth == '07') {sMonth = sMonth + ' ИЮЛЬ';}
else if (sMonth == '08') {sMonth = sMonth + ' АВГУСТ';}
else if (sMonth == '09') {sMonth = sMonth + ' СЕНТЯБРЬ';}
else if (sMonth == '10') {sMonth = sMonth + ' ОКТЯБРЬ';}
else if (sMonth == '11') {sMonth = sMonth + ' НОЯБРЬ';}
else if (sMonth =='12') {sMonth = sMonth + ' ДЕКАБРЬ';};

var string = jParams.getString("flt_idbisobjmc")
+ " "
+ jParams.getString("flt_idbisobjhl")
+ "#{\"idObjectType\": \"" + idObejctType + "\",\"idBisObj\": \"" + bo + "\",\"nImage\":␣
↪\"" + 4 + "\"}"
+ "/Учет прочих операций#{\"idObjectType\": \"" + idObejctType + "\",\"idBisObj\": \"" +␣
↪bo + "\",\"nImage\": \"" + 4 + "\"}"
+ "/"
+ toString(jParams.getNumber("flt_nyear"))
+ "#{\"idObjectType\": \"" + idObejctType + "\",\"idBisObj\": \"" + bo + "\",\"nImage\":␣
↪\"" + 30 + "\"}"
+ "/"
+ sMonth
+ "#{\"idObjectType\": \"" + idObejctType + "\",\"idBisObj\": \"" + bo + "\",\"nImage\":␣
↪\"" + 46 + "\"}";

new("ru.bitec.app.gtk.lang.NString", string);
```

Создание позиции счета-фактуры от позиции авансового отчета

Создает позицию счета-фактуры на основании данных позиции авансового отчета. Скрипт показывает пример создания сущностей одного документа с использованием данных другого документа, проверки значений на `null`, вывода идентификаторов созданных объектов в лог и импорта Java-библиотеки.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
// Испортируем методы
let classLoader = session.getClass().getClassLoader();
let BigDecimal = classLoader.loadClass('java.math.BigDecimal');

// Объявляем переменные
const idvDoc = toLong(33203); // id счета-фактуры, с которым работаем
const gidvSrcDet = toString('51451/33557'); // gid позиции документа источника, с␣
↪которой хотим связать создаваемую позицию
const gidvGdsSrv = Gds_ServiceApi.load(Gds_ServiceApi.findByMnemoCode('Аванс')).gid(); //
↪ gid служебной услуги для авансового счета-фактуры
const nvMethodCalculation = 2b;

// Получаем атрибуты позиции документа источника
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
const aSrcDetAttr = sql(`select
docsrcdet.idvatrate as idvatrate
,docsrcdet.scaption as scaption
,docsrcdet.nqty as nqty
,docsrcdet.nsumexprep as nsum
,docsrcdet.nsumrepbase as nsumbase
,docsrcdet.nsumvatexprep as nsumvat
,docsrcdet.nsumvatexpbase as nsumvatbase
,repitem.scode as scoderepitem
from pm_advrepdet docsrcdet
left join bs_advrepitem repitem on docsrcdet.idadvrepitem = repitem.id
where docsrcdet.gid = '`+gidvSrcDet+`'
;`).asSingle();

let nvQtySrcDet = null;
let svSrcDetCaption = null;
let idvSrcDetVATRate = null;
let svRepItemCode = null;
let nvSrcDetSum = null;
let nvSrcDetSumBase = null;
let nvSrcDetSumVAT = null;
let nvSrcDetSumVATBase = null;

if(isNotNull(aSrcDetAttr.idvatrate)) {
   idvSrcDetVATRate = aSrcDetAttr.idvatrate;
}
if(isNotNull(aSrcDetAttr.scoderepitem)) {
   svRepItemCode = aSrcDetAttr.scoderepitem;
}
if(isNotNull(aSrcDetAttr.scaption)) {
   svSrcDetCaption = aSrcDetAttr.scaption;
}
if(isNotNull(aSrcDetAttr.nqty)) {
   nvQtySrcDet = BigDecimal.valueOf(toDouble(aSrcDetAttr.nqty));
}
if(isNotNull(aSrcDetAttr.nsum)) {
   nvSrcDetSum = aSrcDetAttr.nsum;
}
if(isNotNull(aSrcDetAttr.nsumbase)) {
   nvSrcDetSumBase = aSrcDetAttr.nsumbase;
}
if(isNotNull(aSrcDetAttr.nsumvat)) {
   nvSrcDetSumVAT = aSrcDetAttr.nsumvat;
}
if(isNotNull(aSrcDetAttr.nsumvatbase)) {
   nvSrcDetSumVATBase = aSrcDetAttr.nsumvatbase;
}

// Создаем объекты счета-фактуры (с которым работаем) и позиции, которую хотим добавить
const ropDoc = Stm_InvBillDocInApi.load(idvDoc);
const ropDocDet = Stm_BillDetApi.insertByParent(ropDoc);
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
// Заполняем атрибуты созданной позиции
Stm_BillDetApi.setgidSrc(ropDocDet, gidvSrcDet);
Stm_BillDetApi.setgidGdsSrv(ropDocDet, gidvGdsSrv);
Stm_BillDetApi.setsOtherProduct(ropDocDet, toString(svRepItemCode + " / Элемент АО " +␣
↪svRepItemCode + " " + svSrcDetCaption));
Stm_BillDetApi.setidVATRate(ropDocDet, idvSrcDetVATRate);
Stm_BillDetApi.setnMethodCalculation(ropDocDet, nvMethodCalculation);
Stm_BillDetApi.setnQty(ropDocDet, nvQtySrcDet);
Stm_BillDetApi.setnQtyBase(ropDocDet, nvQtySrcDet);
Stm_BillDetApi.setnSum(ropDocDet, nvSrcDetSum);
Stm_BillDetApi.setnSumDoc(ropDocDet, nvSrcDetSum);
Stm_BillDetApi.setnSumBase(ropDocDet, nvSrcDetSumBase);
Stm_BillDetApi.setnSumToPay(ropDocDet, nvSrcDetSum);
Stm_BillDetApi.setnSumToPayBase(ropDocDet, nvSrcDetSum);
Stm_BillDetApi.setnSumToPayDoc(ropDocDet, nvSrcDetSum);
Stm_BillDetApi.setnSumVAT(ropDocDet, nvSrcDetSumVAT);
Stm_BillDetApi.setnSumVATBase(ropDocDet, nvSrcDetSumVATBase);
Stm_BillDetApi.setnSumVATDoc(ropDocDet, nvSrcDetSumVATBase);
Stm_BillDetApi.setnSumNoTax(ropDocDet, nvSrcDetSum - nvSrcDetSumVAT);
Stm_BillDetApi.setnSumNoTaxBase(ropDocDet, nvSrcDetSum - nvSrcDetSumVAT);
Stm_BillDetApi.setnSumNoTaxDoc(ropDocDet, nvSrcDetSum - nvSrcDetSumVAT);
Stm_BillDetApi.setnSumToPayNoTax(ropDocDet, nvSrcDetSum - nvSrcDetSumVAT);
Stm_BillDetApi.setnSumToPayNoTaxBase(ropDocDet, nvSrcDetSum - nvSrcDetSumVAT);
Stm_BillDetApi.setnSumToPayNoTaxDoc(ropDocDet, nvSrcDetSum - nvSrcDetSumVAT);
Stm_BillDetApi.setnSumToPayVAT(ropDocDet, nvSrcDetSumVAT);
Stm_BillDetApi.setnSumToPayVATBase(ropDocDet, nvSrcDetSumVATBase);
Stm_BillDetApi.setnSumToPayVATDoc(ropDocDet, nvSrcDetSumVATBase);

// Выводим в лог gid созданной позиции счета-фактуры
Btk_InfoLogPkg.info("Счет-фактура: " + toString(ropDoc.gid));
Btk_InfoLogPkg.info("Новая позиция счета-фактуры: " + toString(ropDocDet.gid));
```

Создание связи между хозяйственными операциями и документами денежного потока

Создает записи в коллекции `Bts_DocLink` для отображения авансового отчета на закладке **Связанные документы** в интерфейсах хозяйственных операций. Скрипт показывает пример создания связей между документами через коллекцию `Bts_DocLink`.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
var bvDone = false;
while (!bvDone) {
 let l = sql(`select ar.gid as gidsrc, coalesce(td.gid, cf.giddoc) as gidrec
      from pm_advrepcashflow cf
      join pm_advrep ar on ar.id = cf.idadvrep
      left join act_trans t on t.gid = cf.giddoc
      left join act_transdoc td on td.id = t.idtransdoc
      left join bts_doclink dl on dl.gidsrc = ar.gid and dl.gidrec = coalesce(td.gid, cf.
↪giddoc)
     where ar.gid is not null and cf.giddoc is not null and dl.id is null
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    limit 100`).asList();
  for(w:l){
    Bts_DocLinkApi.register(w.gidsrc, w.gidrec);
  }
  if (l.isEmpty()) {
    bvDone = true;
  } else {
    session.commit();
  }
}
```

Получение подкласса документа

Получает подкласс документа. Скрипт показывает два варианта получения подкласса: если класс документа заранее неизвестен и если класс документа известен. Может использоваться как самостоятельный скрипт или как шаблон внутри других JEXL-скриптов.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт` или в теле другого JEXL-скрипта

Тип: JEXL-скрипт

```
// если не знаем класс документа
var rop = Btk_ObjectPkg.loadAnyObject({gid_документа});
var idSubClass = Btk_ObjectTypeApi.load(rop.getByAttrName("idObjectType")).idSubClass //␣
↪кинет ошибку если в классе документа нет атрибута idObjectType
// если знаем класс документа
var rop = Act_TransApi.load({id_документа});
var idSubClass = Btk_ObjectTypeApi.load(rop.idObjectType).idSubClass
```

Смена знака сумм проводки

Меняет знак сумм проводки. Скрипт показывает, что при работе с API важно учитывать тип данных, который ожидают методы: `Act_TransApi.load()` принимает значение типа `Long`, а методы `Act_TransApi.setnSum()` и `Act_TransApi.setnSumBase()` — значения типа `BigDecimal`.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
const nvSum = toBigDecimal(104467.84);
const nvSumBase = toBigDecimal(104467.84);
var ropTrans = Act_TransApi.load(toLong(1204862));
Act_TransApi.setnSum(ropTrans, -nvSum);
Act_TransApi.setnSumBase(ropTrans, -nvSumBase);
```

Поиск объектов коллекции через корневой объект

Показывает структуру работы с объектами коллекции через `id` корневого объекта. Скрипт можно использовать в теле других JEXL-скриптов, когда нужно перебрать все объекты, связанные с основным объектом, например позиции документа.

Место применения: `В теле JEXL-скрипта`

Тип: JEXL-скрипт

```
// Структура для работы с ропами объектов коллекции через id корневого объекта
var rops = ...Api.byParent(id);
for (rop : rops) {
  ...
}
```

Создание записи в журнале связанных документов

Создает запись в журнале связанных документов `Bts_DocLink` для отображения связи на закладке **Связанные документы**. Может использоваться при восстановлении связей между документами, донастройке отображения связанных документов или исправлении последствий ошибок обработки.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
// Загрузка документа-приемника
var ropOrder = Stm_OrderInApi.load(toLong(17502));

// Загрузка документа-источника
var ropReq = Prs_PurchReqApi.load(toLong(14106));

// Создание записи связи документов
var rop = Bts_DocLinkApi.insert();

// Вывод id созданной записи
dialogs.showMessage(toString(rop.id));

// Заполнение документа-приемника
Bts_DocLinkApi.setgidRec(rop, ropOrder.gid);

// Заполнение документа-источника
Bts_DocLinkApi.setgidSrc(rop, ropReq.gid);
```

Удаление аналитики проводки из JSON

Удаляет значения аналитик проводки из JSON-атрибута. Скрипт загружает проводку, получает JSON с аналитиками по дебету, удаляет указанные элементы и записывает обновленное значение обратно в объект. Также полезен как пример работы с JSON в JEXL.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
var ropTrans = Act_TransApi.load(toLong());// id проводки
var currentJson = ropTrans.jAttrDataDeb;// Получаем значение JSON-переменной в виде␣
↪строки
var jsonObject = new("com.google.gson.JsonParser").parse(currentJson).getAsJsonObject();␣
↪// Преобразуем строку в JSON-объект

jsonObject.remove("idSAPAcc");// Удаляем элемент JSON по его имени
jsonObject.remove("bVld_idSAPAcc");// Удаляем элемент JSON по его имени

Act_TransApi.setjAttrDataDeb(ropTrans, toString(jsonObject));// Записываем обновленное␣
↪значение JSON обратно в объект
```

Замена счета-фактуры в позициях авансового отчета

Заменяет счет-фактуру в указанных позициях авансового отчета на целевой счет-фактуру. Скрипт загружает авансовый отчет и счет-фактуру, удаляет записи включения счета-фактуры в книгу покупок/продаж, временно переводит счет-фактуру в состояние создания, заменяет ссылку на счет-фактуру в позициях и возвращает счет-фактуру в исходное состояние.

Место применения: Сервис > Инструменты > Выполнить JEXL-скрипт

Тип: JEXL-скрипт

```
var ropDoc = Pm_AdvRepApi.load(toLong(37302));// id авансового отчета
var ropInvoice = Stm_InvBillDocInApi.load(toLong(33552));// id счета-фактуры, который␣
↪оставляем
const idvDocDet = [33802, 33803];// id позиций авансового отчета, в которых выполняется␣
↪замена

const idvCurrentState = Btk_ClassStateApi.findByNameAndIdClass('Stm_Accepted',␣
↪ropInvoice.idClass, 0B);
const idvAnnuledState = Btk_ClassStateApi.findByNameAndIdClass('Stm_Create', ropInvoice.
↪idClass, 0B);

// Поиск записей включения счета-фактуры в книгу покупок/продаж
var avidVATBook = sql(`
select
id
from
stm_bookinclsetting vatbooksetting
where
vatbooksetting.giddoc = '` + ropInvoice.gid + `'
;`).asList();

// Удаление найденных записей включения
for (w : avidVATBook) {
  Stm_BookInclSettingApi.delete(Stm_BookInclSettingApi.load(toLong(w.id)));
}

// Временный перевод счета-фактуры в состояние создания
Stm_InvBillDocInApi.setidState(ropInvoice, idvAnnuledState);

// Замена счета-фактуры в позициях авансового отчета
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
for (let i = 0; i < size(idvDocDet); ++i) {
  var ropDocDet = Pm_AdvRepDetApi.load(toLong(idvDocDet[i]));

  Pm_AdvRepDetApi.setidInvBillDocIn(ropDocDet, null);
  Pm_AdvRepDetApi.setidInvBillDocInOT(ropDocDet, ropInvoice.idObjectType);
  Pm_AdvRepDetApi.setidInvBillDocIn(ropDocDet, ropInvoice.id);
  Pm_AdvRepDetApi.includeInInvBillDocIn(ropDocDet, ropInvoice.id);
}

// Возврат счета-фактуры в исходное состояние
Stm_InvBillDocInApi.setidState(ropInvoice, idvCurrentState);
```

Проверка ставок НДС договора и спецификаций

Проверяет, что ставка НДС на реквизитах договора совпадает со ставкой НДС на этапах и позициях спецификаций. Используется при переводе состояния договора: если ставка НДС отличается, перевод состояния прерывается с сообщением об ошибке.

Место применения: `Bts-процедура при переводе состояния`

Тип: JEXL-скрипт

```
var ropaStage = Cnt_ContractApi.getStageByContract(rop.id);

if (isNotNull(rop.idVATRate)) {
  for (ropStage : ropaStage) {
    var ropaPosStage = Cnt_ContractSpecApi.byParent(ropStage);

    // Проверка ставки НДС на этапе договора
    if (ropStage.idVATRate != rop.idVATRate) {
      raise("Необходимо указать одинаковые ставки НДС на вкладке \"Реквизиты\" и \
↪"Спецификации\"!");
    }

    if (isNotNull(ropStage.idVATRate)) {
      for (ropPosStage : ropaPosStage) {
        // Проверка ставки НДС на позиции спецификации
        if (ropPosStage.idVATRate != ropStage.idVATRate) {
          raise("Необходимо указать одинаковые ставки НДС на вкладке \"Реквизиты\" и \
↪"Спецификации\"!");
        }
      }
    }
  }
}
```

Jexl для добавления документов к пуям

Используется для обработки документов, версий, позиций или связанных записей документа.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
var nvCount = sql(`
        select count(distinct t.id) as "nCount"
        from (
        (select r.idWorkStructure as id, s.gidDoc, s.gidDocVer
        from Mct_ActionRef r
        join Mct_Structure s on s.id = r.idStructure
        where s.gidDoc is not null
        ) except (
                select getgidid(ml.gidobject) as id, ml.gidlink, ml.gidlinkver
                from mct_link ml
                where ml.gidobject in (select mw.gid from mct_workstructure mw)
                )
        ) t
        `).asSingle().nCount;
var nvIter = nvCount / 100 + 1;
for (i : (1 .. nvIter)){
        var ropList =  sql(`
        select distinct t.id from (
            (select r.idWorkStructure as id, s.gidDoc, s.gidDocVer
             from Mct_ActionRef r
             join Mct_Structure s on s.id = r.idStructure
             where s.gidDoc is not null)
             except
             (select getgidid(ml.gidobject) as id, ml.gidlink, ml.gidlinkver
             from mct_link ml
             where ml.gidobject in (select mw.gid from mct_workstructure mw))
                         ) t
                order by t.id
                limit 100
                `).foreach(function (r){
                Mct_ActionRefApi.refreshSPLinkForWS(Mct_WorkStructureApi.getGid(r.id));
                })
        commit();
}
```

Добавление Msch Root

Используется для обработки документов, версий, позиций или связанных записей документа.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
for (i : (1 .. 100)){
        var nvOffset = (i-1) * 500
        var ropList =  sql(`
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                select d.id as "idMctDocumentVer" from Mct_SpecificationVer d
                order by d.id
                `
                + "offset " + nvOffset + " limit 500").batchObjLoad(Mct_
↪SpecificationVerPosApi, "idMctDocumentVer");
        for (rop : ropList){
                Mct_SpecificationVerPosApi.updateOrderAttr(rop);
        }
        commit();
}
```

Генерация Structure Lotos

Используется для обработки документов, версий, позиций или связанных записей документа.

Место применения: Сервис > Инструменты > Выполнить JEXL-скрипт

Тип: JEXL-скрипт

```
sql(`
select t1.gid as "gid"
from mct_specificationver t1
left join mct_specification t2 on t1.idmctdocument = t2.id
where t2.idprj = 651
and t2.idprjver = 452
 `).foreach(function(r){
Mct_StructureGenPkg.checkExistanceInStruct(r.gid);
commit();
Mct_StructureGenPkg.structGenFromDoc(r.gid);
commit();
}) ;
```

Lotos Set Gds Pos Type

Используется для обработки документов, версий, позиций или связанных записей документа.

Место применения: Сервис > Инструменты > Выполнить JEXL-скрипт

Тип: JEXL-скрипт

```
var idvGdsPosType = Mct_PosTypeApi.findByMnemoCode("ТМЦ");
var ropList =  sql(`
        select sv.id as "idMctDocumentVer"
        from mct_specificationver sv
        left join mct_specification s on sv.idmctdocument = s.id
        where s.idprj = (select max(id) from bs_prj where scode = 'RSD49')`).
↪batchObjLoad(Mct_SpecificationVerPosApi, "idMctDocumentVer");
for (rop : ropList){
        Mct_SpecificationVerPosApi.setidPosType(rop, idvGdsPosType);
}
commit();
```

Миграция документов с файлами из внешнего источника

Используется для массового создания документов `Wf_Doc` по данным внешнего источника и прикрепления файлов к созданным версиям документов. Скрипт проверяет наличие файла на диске, создает файл в файловом хранилище, создает документ и версию документа с файлом.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретный внешний источник, тип документа, файловое хранилище и путь к файлам. Перед запуском проверьте SQL-запрос, `idvObjectType`, `idvDir`, `idvExt_Tronix` и путь к каталогу файлов.

Тип: JEXL-скрипт

```
var idvDir = null;
var idvObjectType = 75901L;
var idvFileStorage = Wf_Lib.idFileStorageWf();
var idvExt_Tronix = 52L;
if(idvDir != null) {
        var nvCount = sql(`
                select count(*) as "nCount"
                from tronix.old_norm_doc ond
                left join wf_doc d on (d.jidext_dz ->> '52') = cast(ond.id as varchar)
                where d.id is null
                `).asSingle().nCount;
        var nvIter = nvCount / 200 + 1;
        for (i : (1 .. nvCount)){
                sql(`
                        select
                                cast(ond.id as varchar) as sid
                                ,cast(ond.inv as varchar) as "sNumInventory"
                                ,to_date(nullif(ond.Srokvved, 'None'), 'YYYY-mm-dd') as
↪"dStartDate"
                                ,to_date(nullif(ond.Srokprod, 'None'), 'YYYY-mm-dd') as
↪"dProlongPeriod"
                                ,cast(ond.Obozn as varchar) as "sDesignation"
                                ,cast(ond.Naimen as varchar) as "sCaption"
                                ,cast(ond.Kolex as numeric) as "nQtyCopies"
                                ,cast(ond.Tfile as varchar) as "sFilePath"
                        from tronix.old_norm_doc ond
                        left join wf_doc d on (d.jidext_dz ->> '52') = cast(ond.id as␣
↪varchar)
                        where d.id is null
                        order by ond.id
                        limit 200
                `).foreach(function(r){
                        @begin{
                                var filePath = "/mnt/attach/upl_normdoc"+r.sFilePath;
                                var file = new ("java.io.File", filePath)
                                if (!file.exists()) {
                                  raise("Файл " + filePath  + " не существует")
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                                }
                                var is = new("java.io.FileInputStream", file);
                                var idFile = null;
                                @begin{
                                 idFile = Btk_FilePkg.createAndFill(file.getName(),␣
↪idvFileStorage, null, is)
                                }@exception function(e){
                                  is.close();
                                  throw(e);
                                } end;
                                 is.close();
                                var ropDoc = Wf_DocApi.insert();
                                Wf_DocApi.setidParent(ropDoc, idvDir);
                                Wf_DocApi.setidObjectType(ropDoc, idvObjectType);
                                Wf_DocApi.setIdExtForExtSystem(ropDoc, idvExt_Tronix, r.
↪sid);
                                Wf_DocApi.setsNumInventory(ropDoc, r.sNumInventory);
                                Wf_DocApi.setdStartDate(ropDoc, r.dStartDate);
                                Wf_DocApi.setdProlongPeriod(ropDoc, r.dProlongPeriod);
                                Wf_DocApi.setsDesignation(ropDoc, r.sDesignation);
                                Wf_DocApi.setsCaption(ropDoc, r.sCaption);
                                Wf_DocApi.setnQtyCopies(ropDoc, r.nQtyCopies);

                                Wf_DocVerApi.createVerWithFile(ropDoc.idJ(), idFile);
                                Btk_FileApi.setgidSrc(Btk_FileApi.load(idFile), ropDoc.
↪gid());
                        }
                        @exception
                        function(exp){
                                var svErr = "Ошибка загрузки документа " + r.sid + ": "␣
↪+ exp.getCause()
                                println(svErr);
                                tsql("update tronix.old_norm_doc set serr = '"+svErr+ "'␣
↪where id = " + r.sid).execute()
                        }end;
                }) ;
                commit();
        };
};
```

7.6 Справочники и НСИ

Id Group Measuring

Используется для создания, обновления или нормализации справочных данных. Требует проверки исходных условий, идентификаторов и используемых справочников.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var rop = Eqp_TypeEquipmentApi.loadByGid(gidObject.toString());
var caption = rop.scaption;
var idGM =  sql(`
                select t.id as "id"
                from Eqp_GroupMeasuring t
                where upper(t.sCaption) = upper(caption)
                `).asSingle().id;
var ropGM = Eqp_GroupMeasuringApi.load(idGM);
```

Добавление New Msch Group

Используется для создания, обновления или нормализации справочных данных. Требует проверки исходных условий, идентификаторов и используемых справочников.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvMschRoot = Btk_GroupApi.findByMnemoCode("MSCH");
Btk_GroupApi.register("newMsch", "Новая группа МСЧ", idvMschRoot);
commit();
```

Добавление Sectors

Используется для создания, обновления или нормализации справочных данных. Требует проверки исходных условий, идентификаторов и используемых справочников.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvParent = Rss_ResourceApi.findByMnemoCode("08");
var idvResType = Rss_ResourceTypeApi.findByMnemoCode(Rss_ResourceTypeApi.mcProdSite());
sql(`
        select "sSector"
        ,'Участок ' || substring("sSector", '\d{2}$') as "sCaption"
        from (
                select distinct
                        t."Участок" as "sSector"
                from "Sormovo_MschTehProc" t
                where "Цех" = '0'||'08'
                and t."Участок" ~ '.{2}'
        ) tt
        left join rss_resource r on tt."sSector" = r.scode
        where r.id is null
`).foreach(function(r){
        var rop = Rss_ResourceApi.insert();
        Rss_ResourceApi.setidParent(rop, idvParent);
        Rss_ResourceApi.setsCode(rop, r.sSector);
        Rss_ResourceApi.setsCaption(rop, r.sCaption);
        Rss_ResourceApi.setidResourceType(rop, idvResType);
});
commit();
```

Очистка Spaces Gds Lotos

Используется для создания, обновления или нормализации справочных данных. Требует проверки исходных условий, идентификаторов и используемых справочников.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
sql(`
with changed as (
        select
                gds.id as id
                ,gds.sArticle as sArticle
                ,regexp_replace(trim(gds.sArticle), '[\s]{2,}', ' ', 'g') as sArticleNew
                ,gds.sDesignation as sDesignation
                ,regexp_replace(trim(gds.sDesignation), '[\s]{2,}', ' ', 'g') as␣
↪sDesignationNew
                ,gds.sDesignationOld as sDesignationOld
                ,regexp_replace(trim(gds.sDesignationOld), '[\s]{2,}', ' ', 'g') as␣
↪sDesignationOldNew
                ,gds.sName as sName
                ,regexp_replace(trim(gds.sName), '[\s]{2,}', ' ', 'g') as sNameNew
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        from Bs_Goods gds )
select
        ch.id as "id"
        ,ch.sArticle as "sArticle"
        ,ch.sArticleNew as "sArticleNew"
        ,ch.sDesignation as "sDesignation"
        ,ch.sDesignationNew as "sDesignationNew"
        ,ch.sDesignationOld as "sDesignationOld"
        ,ch.sDesignationOldNew as "sDesignationOldNew"
        ,ch.sName as "sName"
        ,ch.sNameNew as "sNameNew"
from changed ch
where (ch.sArticle != ch.sArticleNew
                or ch.sDesignation != ch.sDesignationNew
                or ch.sDesignationOld != ch.sDesignationOldNew
                or ch.sName != ch.sNameNew)
 `).foreach(function(r){
@begin{
        var rop = Bs_GoodsApi.load(r.id);
        //правим код
        if(r.sArticle != r.sArticleNew){
                Bs_GoodsApi.setsArticle(rop, r.sArticleNew);
        }
        //правим Обозначение
        if(r.sDesignation != r.sDesignationNew){
                Bs_GoodsApi.setsDesignation(rop, r.sDesignationNew);
        }
        //правим Старое обозначение
        if(r.sDesignationOld != r.sDesignationOldNew){
                Bs_GoodsApi.setsDesignationOld(rop, r.sDesignationOldNew);
        }
        //правим наименование
        if(r.sName != r.sNameNew){
                Bs_GoodsApi.setsName(rop, r.sNameNew);
        }
        commit();
  }
@exception
function(exp){
        println("[JEXL Exception] " + exp.getCause());
}end;
}) ;
```

Миграция Gds Src

Используется для создания, обновления или нормализации справочных данных. Требует проверки исходных условий, идентификаторов и используемых справочников.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
for (i : (1 .. 100)){
        var nvOffset = (i-1) * 500
        var ropList =  sql(`
                select d.id as "id" from Bs_goods d
                where d.gidSrc is not null
                order by d.id
                `
                + "offset " + nvOffset + " limit 500").batchObjLoad(Bs_GoodsApi, "id");
        for (rop : ropList){
                var gidvSrc = rop.copyAro().gidSrc();
                Bs_GoodsApi.addSrc(rop, gidvSrc, 1B);
        }
        commit();
}
```

Обновление MSCH Placer

Используется для создания, обновления или нормализации справочных данных. Требует проверки исходных условий, идентификаторов и используемых справочников.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
for (i : (1 .. 20)){
        var nvOffset = (i-1) * 1000
        var ropList =  sql(`
                select t.id as "id" from Bs_Goods t
                where brossip = 1
                order by t.id
                `
                + "offset " + nvOffset + " limit 1000").batchObjLoad(Bs_GoodsApi, "id");
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        for (rop : ropList){
                Mct_UnitProductPkg.setbPlacer(rop, 1B);
        }
        commit();
}
```

Обновление MSCH Purchase

Используется для создания, обновления или нормализации справочных данных. Требует проверки исходных условий, идентификаторов и используемых справочников.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
for (i : (1 .. 20)){
        var nvOffset = (i-1) * 1000
        var ropList =  sql(`
                select t.id as "id" from Bs_Goods t
                where bpurchase = 1
                order by t.id
                `
                + "offset " + nvOffset + " limit 1000").batchObjLoad(Bs_GoodsApi, "id");
        for (rop : ropList){
                Mct_UnitProductPkg.setbPurchase(rop, 1B);
        }
        commit();
}
```

7.7 Маршруты согласования

Назначение участников маршрута из атрибутов документа

Используется, чтобы назначить пользователей на этапы маршрута на основе значений, хранящихся в атрибутах документа. Поддерживает одновременное назначение нескольких ролей, например исполнитель, подписывающий и согласовывающий. Для каждой роли выполняется проверка существования учетной записи; при необходимости допускаются исключения на основе параметров документа.

Место применения: `Приложение "Управление бизнес-процессами" > Справочники > Схема бизнес-процесса > [Процесс] > Редактировать > Вкладка "Процедуры" > Поле процедуры выполнения`.

Тип: JEXL-скрипт

```
// Получаем информацию о бизне-процессе и записываем переменную с ним и документом
var idProc = Bpm_PrStateApi.getProcessId(idpPrState); // Получаем id поцесса
var idvDoc = Bpm_PrDocApi.getProcDocByPrState(idpPrState); // Получаем id состояния
↪процесса для которого выполняется процедура
var ropProc = Bpm_ProcessApi.load(idProc); // Записываем процесс в переменную как объект
↪выполняемой процедуры

// Получаем параметры-коды объектных привилегий схемы бизнес-процесса и создаем
↪субъектов бизнес-процесса как объекты выполняемой процедуры
var paramArr = param.split(";");

// Исполнитель
var svEmployee = paramArr[0];
var ropTargetOptEmployee = Bpm_PrSubjectApi.getByPSSubject(ropProc, svEmployee);
var ropTargetEmployee = ropTargetOptEmployee.get();
var semployee = `
select coalesce(pers.iduser, 0) as iduseremployee
from wf_doc doc
left join bs_employee e on cast(doc.jobjattrs_dz ->> 'idR1EmployeeSng' as int8) = e.id
left join bs_person pers on e.idperson = pers.id
where doc.gid = '${idvDoc}'
`;

// Подписывающий
var svEmployeeSign = paramArr[2];
var ropTargetOptEmployeeSign = Bpm_PrSubjectApi.getByPSSubject(ropProc, svEmployeeSign);
var ropTargetEmployeeSign = ropTargetOptEmployeeSign.get();
var semployeesign = `
select
coalesce(pers.iduser, 0) as iduseremployeesign
,bo.scode as bocode
from wf_doc doc
left join bs_employee e on cast(doc.jobjattrs_dz ->> 'idR1EmployeeSignSng' as int8) = e.
↪id
left join bs_person pers on e.idperson = pers.id
left join bs_bisobj bo on doc.idbisobj = bo.id
where doc.gid = '${idvDoc}'
`;

// Согласовывающий
var svEmployeeMain = paramArr[1];
var ropTargetOptEmployeeMain = Bpm_PrSubjectApi.getByPSSubject(ropProc, svEmployeeMain);
var ropTargetEmployeeMain = ropTargetOptEmployeeMain.get();
var semployeemain = `
select coalesce(pers.iduser, 0) as iduseremployeemain
from wf_doc doc
left join bs_employee e on cast(doc.jobjattrs_dz ->> 'idR1EmployeeMainSng' as int8) = e.
↪id
left join bs_person pers on e.idperson = pers.id
where doc.gid = '${idvDoc}'
`;
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
// Установка пользователей с селекционного экрана в качестве подписантов

// Исполнитель
var lemployee = sql(semployee).asList();
for (w:lemployee){
        if(w.iduseremployee == 0) {raise('Внимание! Для справки не указан исполнитель.
↪');};}
        Bpm_PrSubjUserApi.register(ropTargetEmployee , w.iduseremployee, 0B, 0B);
}

// Подписывающий
var lemployeesign = sql(semployeesign).asList();
for (w:lemployeesign){
        if(w.iduseremployeesign == 0 and w.bocode != '0100') {raise('Внимание! Для␣
↪справки не указан подписывающий.');};}
        Bpm_PrSubjUserApi.register(ropTargetEmployeeSign , w.iduseremployeesign, 0B, 0B);
}

// Согласовывающий
var lemployeemain = sql(semployeemain).asList();
for (w:lemployeemain){
        if(w.iduseremployeemain == 0) {raise('Внимание! Для справки не указан␣
↪согласовывающий.');};}
        Bpm_PrSubjUserApi.register(ropTargetEmployeeMain, w.iduseremployeemain, 0B, 0B);
}
```

Назначение участника маршрута из связанного документа

Используется, чтобы назначить на этап маршрута пользователя, указанного в связанном документе, например куратора в договоре. Скрипт извлекает данные через связи между сущностями и регистрирует пользователя как исполнителя. При отсутствии данных вызывает ошибку.

Место применения: `Управление бизнес-процессами > Справочники > Схема бизнес-процесса > [Процесс] > Редактировать > Вкладка "Процедуры" > Поле процедуры выполнения`.

Тип: JEXL-скрипт

```
var idProc = Bpm_PrStateApi.getProcessId(idpPrState);
var svSubj = param;
var idvDoc = Bpm_PrDocApi.getProcDocByPrState(idpPrState);
var ropProc = Bpm_ProcessApi.load(idProc);
var ropTargetOpt = Bpm_PrSubjectApi.getByPSSubject(ropProc, svSubj);
var ropTarget = ropTargetOpt.get();
var s = `
select coalesce (u.iduser, 0) as iduser
from cnt_contract r
join bs_employee e on r.idcuratoremployee = e.id
join bs_person u on e.idperson = u.id
where r.gid = ${idvDoc}
`;

var l = sql(s).asList();
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
for (w:1){
        if(w.iduser == 0) {raise('Внимание! Для договора не указан куратор.');}
        Bpm_PrSubjUserApi.register(ropTarget, w.iduser, 0B, 0B);
}
```

Проверка доступности исполнителей на этапе маршрута

Используется, чтобы проверить, доступен ли хотя бы один исполнитель на этапе маршрута. Если все назначенные пользователи и профили недоступны, скрипт устанавливает флаг и отправляет системное уведомление, предотвращая зависание задачи.

Место применения: `Управление бизнес-процессами > Справочники > Схема бизнес-процесса > [Процесс] > Редактировать > Вкладка "Процедуры" > Поле процедуры выполнения`.

Тип: JEXL-скрипт

```
/*
spCheckAvailableExecutorByPsSubject
Проверка свободных исполнителей по субъекту процесса
*/


/*
Проверка свободных исполнителей по субъекту процесса для выполнения задачи
Параметры: {системное имя субъекта процесса};{параметр для записи числового значения}
*/

var NNumber = function (number) { new ("ru.bitec.app.gtk.lang.NNumber", number); }

// param - строка с произвольными параметрами jexl скрипта
var paramArr = param.split(";");
var svSignSubj = paramArr[0];
var svParamName = paramArr[1].toString();

// idpPrState- параметр jexl скрипта с id состояния
var ropProc = Bpm_PrStateApi.getProcess(idpPrState);
var ropSignOpt = Bpm_PrSubjectApi.getByPSSubject(ropProc, svSignSubj);

if (!empty(ropSignOpt)) {
  var ropSign = ropSignOpt.get();
  var aroSign = ropSign.copyAro();
  var idPrSubject = aroSign.id;

  var sqlUser = `select btkUser.gid from Btk_User btkUser
    where btkUser.id in (select prsubjuser.idUser from Bpm_PrSubjUser prSubjUser
    where prsubjuser.idPrSubject = `+idPrSubject+`)`;
  var gidavUser = sql(sqlUser).asList();

  var sqlAcProfile = `select acProfile.gid from Btk_AcProfile acProfile
    where acProfile.id in (select prSubjAcProfile.idAcProfile from Bpm_PrSubjAcProfile␣
↪prSubjAcProfile
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    where prSubjAcProfile.idPrSubject = `+idPrSubject+`)`;
  var gidavAcProfile = sql(sqlAcProfile).asList();
  gidavUser.addAll(gidavAcProfile);

  if (!Bpm_ProcessApi.checkVarExists(ropProc, svParamName)) {
    raise(`Переменной`+svParamName+` для данного процесса не существует!!!`);
  }
  Bpm_ProcessApi.setProcessVar(ropProc, svParamName, 1);
  for (gidvResource : gidavUser) {
    var query = `select t.id
                from Mpr_OperModeResource t
                where t.gidResource = '`+gidvResource.gid+`'
                 and t.dbegin < now() and (t.dend is null or t.dend > now())`;
    var lst = sql(query).asList();
    if (lst.size() == 0) {
      Bpm_ProcessApi.setProcessVar(ropProc, svParamName, 0);
    }
  }

  if (Bpm_ProcessApi.getProcessVar(ropProc, svParamName).toLong() == 0.0) {
    var idvState = Bpm_PrStateApi.load(idpPrState).copyAro().idState();
    var svCaption = Btk_ClassStateApi.load(idvState).copyAro().sCaption();
    selection.coreRep().application().notifications().notify("Задача, направленная по␣
↪маршруту, не может быть выполнена, поскольку в данный момент отсутствуют доступные␣
↪исполнители. Текущее состояние задачи: "+svCaption);
  }
}
```

Инициализация документа параметрами селекционного экрана отчета

Используется, чтобы передать значения фильтров селекционного экрана отчета в атрибуты создаваемого документа, например бизнес-объект или период. Скрипт обеспечивает корректную инициализацию документа при запуске маршрута согласования из отчетной формы.

Место применения: `Настройки системы > Отчеты > Настройка отчетных форм > [Отчетная форма] > Редактировать > Тип документа создаваемого по отчету > [Документ] > Поле "Процедура установки атрибутов документа"`.

Тип: JEXL-скрипт

```
var rop = Wf_DocApi.load(idDoc); // Загружаем объект WF_Doc как переменную
var ropRpt = Rpt_EntityExecApi.load(idEntityExec); // Создаем объект селекционного␣
↪экрана в кэше
var jsonString = ropRpt.jParams.toString(); // Получаем набор параметров
var json = toJObject(jsonString); // Записываем полученные параметры в json
var jData = Wf_DocApi.getJData(rop); // Передаем json с параметрами в Wf_Doc

var idBisObj = json.getLong("flt_idbisobj"); // Из Json параметров получаем значение␣
↪конкретного атрибута строкой

/*
// Возможность получения данных в т.ч. запросом
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
var idacceptmmemo = json.getString("flt_idaccept");
var sql_text = `select id from bs_employee be where smmemocode_dz = '` + idacceptmmemo +
↪`' and be.sposition = '` + idacceptpos + `'`;
*/

// Установка значений
Wf_DocApi.setidBisObj(rop, idBisObj);
// Wf_DocApi.setidBisObj(rop, json.getLong("flt_idBisObj")); // Альтернативный вариант
↪сеттера
// Wf_DocApi.setObjAttrValue(rop, 'nYear', json.getLong("flt_nyear")); Сеттер для Json
↪атрибута
```

Установка флага маршрута по учетным данным документа

Используется, чтобы анализировать учетные данные документа и устанавливать флаг в переменной процесса при выполнении заданного условия. Результат может использоваться для условного перехода в маршруте согласования.

Место применения: `Управление бизнес-процессами > Справочники > Схема бизнес-процесса > [Процесс] > Редактировать > Вкладка "Процедуры" > Поле процедуры выполнения`.

Тип: JEXL-скрипт

```
// Процедура проверки наличия счета "76" в контировке одной из позиций приходного акта

var svVar = param; // Переменная для параметра, в который будет записан результат
↪процедуры
var ropProc = Bpm_PrStateApi.getProcess(idpPrState); // Переменная для хранения
↪контекста выборки конкретного маршрута
var gidvDoc = Bpm_PrDocApi.getProcDocByPrState(idpPrState); // Переменная для хранения
↪документа - объекта маршрута


// Переменная для хранения результата sql-запроса, через выполнение которого проверяется
↪наличие в контировке позиции приходного акта счета "76"
var s =
`
select case when (acc.sCode like '%76%') then 1 else 0 end as isacc76
from bbb_objdistr bo
join bs_acc acc on bo.idaccacc = acc.id
join stm_actindet aid on bo.gidobj = aid.gid
join stm_actin ai on aid.idDoc = ai.id
where ai.gid = '`+ idvDoc + `'
`;

/*
Цикл, который будет выполнять sql запрос для каждой позиции контировки приходного акта и
↪если найдет хотя бы одну,
у которой счет учета услуг содержит код "76" запишет значение переменной для проверки
↪условия наличия счета "76" в маршруте true
*/
var l = sql(s).asList();
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
for(w:l){
if (l.isacc76 == 1)
   Bpm_ProcessApi.setProcessVar(ropProc , "isacc76", true);
}
```

Расчет разницы между датами

Рассчитывает разницу между двумя датами. Скрипт можно использовать для реализации проверок в Bts-процедурах, например при проверке длительности периода, срока действия документа или допустимого интервала между датами. В примере показаны два варианта расчета: через `toDate()` и через `java.text.SimpleDateFormat`.

Место применения: `В теле JEXL-скрипта`.

Тип: JEXL-скрипт

```
// Вариант 1
var b = toDate('27.01.2025');
var c = toDate('27.03.2025');
var d = (c.getTime() - b.getTime()) / (1000 * 60 * 60 * 24);

// Вариант 2
var format = new java.text.SimpleDateFormat("dd.MM.yyyy");
var start = format.parse("01.01.2026");
var end = format.parse("31.01.2026");

var diff = end.getTime() - start.getTime();
var nPeriod = (diff / 86400000) + 1;
```

Очистка Error Proc

Используется для настройки, очистки или запуска элементов маршрутов согласования и бизнес-процессов.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
var ropList =  sql(`
        select distinct t.idprocess as id
        from bpm_prstate st
        left join bpm_prthread t on st.idprthread = t.id
        where st.id = st.idparent
        `).batchObjLoad(Bpm_ProcessApi, "id");
for (rop : ropList){
        Bpm_ProcessApi.delete(rop);
}
commit();
```

Запуск процесса

Используется для настройки, очистки или запуска элементов маршрутов согласования и бизнес-процессов.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
var gidvSurvey = Bpm_PrDocApi.getProcDocByPrState(idpState);
var idvProcessSchema = Bpm_ProcessSchemaApi.findByMnemoCode(spParams);
var idvUser = Btk_UserApi.getCurrentUserId();
if(idvProcessSchema != null) {
        sql(`
                select
                        r.gid
                from sur_respsurvey r
                join sur_survey s on r.idsurvey = s.id
                where s.gid = '` + gidvSurvey + `'
                `).foreach(
                var ropProcess = Bpm_ProcessPkg.createProcessSimple(r.gid,␣
↪idvProcessSchema);
                Bpm_ProcessPkg.startProcess(ropProcess, idvUser);
                )
        } else
        raise ("Схема с кодом ... не найдена")
```

Миграция Bpm Class Collection

Используется для настройки, очистки или запуска элементов маршрутов согласования и бизнес-процессов.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
Btk_Pkg.setRWSharedUOWEditType();
var idvIntWarClass = Stk_InternalWarrantApi.idClass();
var idvPrDocClass = Bpm_PrDocApi.idClass();
var idvPrDocRefAttr = Btk_AttributeApi.findByMnemoCode(Btk_ClassApi.load(idvPrDocClass),
↪"gidRefDocument");
Btk_ClassCollectionApi.register(
        idvIntWarClass,
        idvPrDocClass,
        idvPrDocRefAttr,
        "error"
);
commit();
```

Обновление признаков версии схемы процесса

Используется для массового обновления настроек версий схем бизнес-процессов. Скрипт работает с версиями схем процессов и их элементами, поэтому относится к настройке маршрутов согласования.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретные схемы бизнес-процессов и условия отбора. Перед запуском проверьте SQL-запросы, идентификаторы и значения обновляемых признаков.

Тип: JEXL-скрипт

```
var ropList =  sql(`
        select d.id as "id" from Bpm_PSVersion d
        `).batchObjLoad(Bpm_PSVersionApi, "id");
for (rop : ropList){
        Bpm_PSElementApi.registerDefaultElements(rop);
}
commit();
```

7.8 Массовая обработка

Шаблон массовой обработки через ElExpOQuery

Используется как основа для написания производительных JEXL-скриптов, обрабатывающих тысячи записей. Демонстрирует использование объектного запроса `ElExpOQuery` для пакетной загрузки сущностей и обработки по чанкам.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную сущность и версию системы. Перед использованием убедитесь, что целевой API и соответствующие Scala-объекты доступны в вашей версии платформы.

Тип: JEXL-скрипт

```
const vDocApi = Co_ProdCostDocEntryApi;

const mkDocOQuery = (idap) => {
  const vOQ = new ru.bitec.app.gtk.eclipse.query.ElExpOQuery(vDocApi);
  const idavAL = new java.util.ArrayList();
  idap.foreach(idv => idavAL.add(idv));

  vOQ.forWhere(vBuilder => vBuilder.get("id").in(idavAL.toArray()) );
  vOQ.batchIn(
    asScala([session.sbtClassLoader().loadClass("ru.bitec.app.co.prodcostdocentry.Co_
↪ProdCostDocEntryDetAta$").MODULE$, ...])
  );
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
  return vOQ;
};

asScala(sql(`
  select t.id
    from Co_ProdCostDocEntry t
   where t.dDoc is not null
     and t.idStateMC >= 300
`).asList()).grouped(400).foreach(ravBatch => {
  mkDocOQuery(ravBatch.map(r => r.id)).foreach(ropDoc => {
    Co_ProdCostRegFullApi.createItemsByDoc(ropDoc, ropDoc.dDoc);
  });
  commit();
});
```

Шаблон массовой обработки через Btk_BulkProcessPkg

Используется как основа для массовой обработки записей с автоматическим разбиением на чанки. Демонстрирует применение `Btk_BulkProcessPkg.chunkedQuery` для безопасной работы с большими выборками без переполнения памяти.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную сущность и версию системы. Классы Scala-объектов могут отсутствовать или измениться в новых версиях платформы.

Тип: JEXL-скрипт

```
Btk_BulkProcessPkg.chunkedQuery(
  Co_ProdCostDocEntryApi,
  5000L,
  `select t.id from Co_ProdCostDocEntry t`,
  asScala([...]), // Неизвестен способ передачи NamedParameter из JEXL
  asScala([session.sbtClassLoader().loadClass("ru.bitec.app.co.prodcostdocentry.Co_
↪ProdCostDocEntryDetAta$").MODULE$, ...]),
  false,
  false,
  ropa => {
    ropa.foreach(rop => Btk_InfoLogPkg.info(rop.gid));
  }
);
```

Обработка данных пакетами

Показывает подход для массового изменения большого объёма данных, когда обработка всех записей за один проход может привести к переполнению памяти. Записи выбираются и обрабатываются пачками, после каждой пачки выполняется `commit()`.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

> **Внимание**
>
> Перед использованием замените SQL-запрос, размер пачки и логику обработки на значения, подходящие для вашей задачи.

Тип: JEXL-скрипт

```
// Обработка данных по пачкам
var batchSize = 200;
var nCount = 0;
var nOffset = 0;

var sqlText = `
<запрос>
`;

while (true) {
  // offset не нужен, если после каждой пачки отдается commit в БД
  // и запрос исключает уже обработанные записи
  var listSQL = sql(sqlText + " offset " + nOffset + " limit " + batchSize).asList();

  // Условие выхода из цикла
  if (listSQL.isEmpty()) {
    break;
  }

  for (data : listSQL) {
    var ropPlacement = Bs_PlacementApi.load(data.id);
    Bs_PlacementApi.setObjAttrValue(ropPlacement, "idMonitorTrainLocationType", data.
↪idtype);
    nCount += 1;
  }

  commit();
  nOffset += batchSize;
}

// Результат
if (nCount == 0) {
  dialogs.showMessage("Исправлять нечего.");
} else {
  dialogs.showMessage("Исправлено " + nCount + " записей.");
}
```

Изменение номеров хозяйственных операций из табличного файла

Изменяет номера хозяйственных операций по данным из Excel-файла. Скрипт читает строки, получает новый номер документа и `gid` хозяйственной операции, загружает объект и изменяет значение номера. Если объект не найден, информация записывается в лог.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

> **Внимание**
>
> Перед запуском проверьте имя листа, номера столбцов и способ загрузки объекта по `gid`.

Тип: JEXL-скрипт

```
var fun = x -> {
  var sheet = x.getSheet("Sheet1");
  var lastRowNum = sheet.getLastRowNum();

  // Цикл for с начальным значением 2 до последней строки
  for (var nvi = 2; nvi <= lastRowNum; nvi = nvi + 1) {
    var vRow = sheet.getRow(nvi);
    var svNumDocNew = vRow.getCell(6).getStringCellValue();
    var gidvTransDoc = vRow.getCell(7).getStringCellValue();
    var ropTransDoc = null;

    @begin {
      ropTransDoc = Act_TransDocApi.load(toLong(parseId(gidvTransDoc)));
      var svNumDocOld = ropTransDoc.sNumDoc;
      Act_TransDocApi.setsNumDoc(ropTransDoc, svNumDocNew);
      var svInfoLog = "Номер хозяйственной операции " + gidvTransDoc + " " + svNumDocOld␣
↪+ " заменен на номер " + svNumDocNew;
      Btk_InfoLogPkg.info(svInfoLog);
      commit();
    }

    if (ropTransDoc == null) {
      Btk_InfoLogPkg.info("Объект " + gidvTransDoc + " не найден");
      continue; // Переходим к следующей строке
    }
  }

  audInfo('Данные загружены');
  true;
}

lib("Btk_XlsxLib").uploadParseFiles(fun);
```

Загрузка данных из Excel в загрузочную таблицу

Показывает пример обработки табличного файла: задаёт структуру колонок, читает строки листа, преобразует значения, сопоставляет данные со справочниками и создаёт записи в загрузочной таблице. Если часть данных не удалось сопоставить, для записи устанавливается признак ошибки.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

В текущем виде скрипт содержит захардкоженные названия файла, листа, ресурс и проектные классы. Перед использованием замените значения и API-классы на актуальные для своей задачи.

Тип: JEXL-скрипт

```
const sFile = "Производственные задания ККЦ за 2508"; // Название файла
const sSheetName = "Производственные задания ККЦ 25"; // Название листа
const Bts_XlsxPkg = lib("Bts_XlsxPkg");

if (isNull(sFile) || sFile == "") raise(`Не указано название файла\n\n`);
if (isNull(sSheetName) || sSheetName == "") raise(`Не указано название листа\n\n`);

const CellDataType = Bts_XlsxPkg.getCellDataTypes();
const avRowColumnDesc = [
  ["Бизнес-единица", CellDataType.String], // 0
  ["Организация (склад)", CellDataType.String], // 1
  ["Формула", CellDataType.String], // 2
  ["Версия формулы", CellDataType.String], // 3
  ["Номер ПЗ", CellDataType.String], // 4
  ["Тип позиции", CellDataType.String], // 5
  ["Позиция", CellDataType.String], // 6
  ["Описание", CellDataType.String], // 7
  ["Кол-во", CellDataType.String], // 8
  ["ЕИ", CellDataType.String], // 9
  ["Партия", CellDataType.String], // 10
  ["Дата выполнения", CellDataType.String], // 11
  ["Размещение затрат", CellDataType.String], // 12
  ["Техкарта", CellDataType.String], // 13
  ["Описание техкарты", CellDataType.String], // 14
  ["Шаг техкарты", CellDataType.String], // 15
  ["Описание шага", CellDataType.String], // 16
  ["Операция", CellDataType.String], // 17
  ["Дата начала операции", CellDataType.String], // 18
  ["Дата завершения операции", CellDataType.String] // 19
];

const getIdBisObj = (sBisObj) -> {
  const sBisObjQuery = `
select
  t.id
from Bs_BisObj t
where lower(t.sCaption) = lower('${sBisObj}')`;
  const rowaBisObj = sql(sBisObjQuery).asList();
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
  if (rowaBisObj.size() == 0) return null;
  return rowaBisObj[0].id;
}

const getIdMsr = (sMsr) -> {
  const sMsrQuery = `
select
  t.id
from msr_measureitem t
where lower(t.smnemocode_dz) = lower('${sMsr}') or lower(t.sshortname) = lower('${sMsr}
↪')`;
  const rowaMsr = sql(sMsrQuery).asList();
  if (rowaMsr.size() == 0) return null;
  return rowaMsr[0].id;
}

const getsCodeMatNormType = (sTypePos) -> {
  if (sTypePos == "Побочный продукт") return "byproduct";
  else if (sTypePos == "Продукт") return "product";
  else if (sTypePos == "Ингредиент") return "ingredient";
  else raise(`Ошибка сопоставления мнемокода Bs_MaterialNormType с sTypePos\n\n`);
}

const getGidGds = (sArticle) -> {
  const sProductQuery = `
select
  t.gid
from Bs_Goods t
where lower(t.sArticle) = lower('${sArticle}')`;
  const rowaProduct = sql(sProductQuery).asList();
  if (rowaProduct.size() == 0) return null;
  return rowaProduct[0].gid;
}

const getIdCons = (sCons) -> {
  const sConsQuery = `
select
  t.id
from Stk_Cons t
where lower(t.sCaption) = lower('${sCons}')`;
  const rowaCons = sql(sConsQuery).asList();
  if (rowaCons.size() == 0) return null;
  return rowaCons[0].id;
}

const getIdStock = (sStock) -> {
  const sQuery = `
select
  t.id
from Stk_Stock t
where lower(t.sMnemoCode_dz) = lower('${sStock}')`;
  const rowaStock = sql(sQuery).asList();
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
  if (rowaStock.size() == 0) return null;
  return rowaStock[0].id;
}

const ExcelStrToDate = (sExcelStr) -> {
  if (sExcelStr == null || sExcelStr == '') return null;
  return sql(`select to_timestamp('${sExcelStr}', 'MM/dd/yy HH24:MI:SS') as date`).
↪asSingle().date;
}

const getidQuotaVers = (sQuotaVers, sQuota) -> {
  const sQuotaVersQuery = `
select
  t.id
from Qt_QuotaVersion t
left join Qt_Quota t2 on t2.id = t.idQuota
where t.nRevision = cast('${sQuotaVers}' as int) and t2.sCode = '${sQuota}'`;
  const rowaQuotaVers = sql(sQuotaVersQuery).asList();
  if (rowaQuotaVers.size() == 0) return null;
  return rowaQuotaVers[0].id;
}

const getIdOper = (sOper) -> {
  const sQuery = `
select
  t.idOper as id
from Zmmkimp_ProductOperLoad t
where lower(t.sCode) = lower('${sOper}')
limit 1
`;
  const rowaOper = sql(sQuery).asList();
  if (rowaOper.size() == 0) return null;
  return rowaOper[0].id;
}

const fun = (x) -> {

  const vFormulaEvaluator = x.getCreationHelper().createFormulaEvaluator();
  const sheet = x.getSheet(sSheetName);
  if (sheet == null) raise(`Лист '${sSheetName}' не существует\n`);
  const lastRowNum = sheet.getLastRowNum();
  if (lastRowNum == -1 || lastRowNum == 0) raise(`Пустой лист '${sSheetName}'\n`);

  var i = 1;

  while (i <= lastRowNum) {
    const vRow = sheet.getRow(i);
    const avCellValue = Bts_XlsxPkg.extractRowAttrs(vFormulaEvaluator, vRow,␣
↪avRowColumnDesc);

        const ropWorkAssignmentLoad = Zmmkimp_WorkAssignmentLoadApi.insert();
    var bError = 0;
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    const apiWAL = Zmmkimp_WorkAssignmentLoadApi;

        // В данном случае Ресурс константа
    const sResource = '078';
    if (sResource != "ПУСТОЕ") apiWAL.setsResource(ropWorkAssignmentLoad, sResource);

    const idResource = Rss_ResourceApi.findByMnemoCode(sResource);
    if (isNull(idResource)) bError = 1;
    else apiWAL.setidResource(ropWorkAssignmentLoad, idResource);

    const sBisObj = avCellValue.get(0).asNString();
    if (sBisObj != "ПУСТОЕ") apiWAL.setsBisObj(ropWorkAssignmentLoad, sBisObj);

    const idBisObj = getIdBisObj(sBisObj);
    if (isNull(idBisObj)) bError = 1;
    else apiWAL.setidBisObj(ropWorkAssignmentLoad, idBisObj);

        const sStock = avCellValue.get(1).asNString();
    if (sStock != "ПУСТОЕ") apiWAL.setsStock(ropWorkAssignmentLoad, sStock);

        const idStock = getIdStock(sStock);
        if (isNull(idStock)) bError = 1;
    else apiWAL.setidStock(ropWorkAssignmentLoad, idStock);

        const sQuota = avCellValue.get(2).asNString();
    if (sQuota != "ПУСТОЕ") apiWAL.setsQuota(ropWorkAssignmentLoad, sQuota);

        const sQuotaVers = avCellValue.get(3).asNString();
    if (sQuotaVers != "ПУСТОЕ") apiWAL.setsQuotaVers(ropWorkAssignmentLoad, sQuotaVers);

        const idQuotaVers = getidQuotaVers(sQuotaVers, sQuota);
    if (isNull(idQuotaVers)) bError = 1;
    else apiWAL.setidQuotaVers(ropWorkAssignmentLoad, idQuotaVers);

        const sWorkAssignment = avCellValue.get(4).asNString();
    if (sWorkAssignment != "ПУСТОЕ") apiWAL.setsWorkAssignment(ropWorkAssignmentLoad,␣
↪sWorkAssignment);

        const sTypePos = avCellValue.get(5).asNString();
    if (sTypePos != "ПУСТОЕ") apiWAL.setsTypePos(ropWorkAssignmentLoad, sTypePos);

        const sCodeMatNormType = getsCodeMatNormType(sTypePos);
    const idMatNormType = Bs_MaterialNormTypeApi.findByMnemoCode(sCodeMatNormType);
    if (isNull(idMatNormType)) bError = 1;
    else apiWAL.setidMatNormType(ropWorkAssignmentLoad, idMatNormType);

        if (isNotNull(idMatNormType)) {
      const nDirection = Bs_MaterialNormTypeApi.load(idMatNormType).nDirection;
      apiWAL.setnDirection(ropWorkAssignmentLoad, nDirection);
    } else bError = 1;
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    const sGdsCode = avCellValue.get(6).asNString();
    if (sGdsCode != "ПУСТОЕ") apiWAL.setsGdsCode(ropWorkAssignmentLoad, sGdsCode);

        const sGdsDesc = avCellValue.get(7).asNString();
    if (sGdsDesc != "ПУСТОЕ") apiWAL.setsGdsDesc(ropWorkAssignmentLoad, sGdsDesc);

    const gidGds = getGidGds(sGdsCode);
    if (isNull(gidGds)) bError = 1;
    else apiWAL.setgidGds(ropWorkAssignmentLoad, gidGds);

    const nQty = avCellValue.get(8).asNString();
    if (nQty != "ПУСТОЕ") apiWAL.setnQty(ropWorkAssignmentLoad, nQty.toBigDecimal());

    const sMsr = avCellValue.get(9).asNString();
    if (sMsr != "ПУСТОЕ") apiWAL.setsMsr(ropWorkAssignmentLoad, sMsr);

    const idMsr = getIdMsr(sMsr);
    if (isNull(idMsr)) bError = 1;
    else apiWAL.setidMsr(ropWorkAssignmentLoad, idMsr);

    const nConvert = Bs_GoodMsrItemApi.getGdsnRate(parseId(gidGds), idMsr);
    if (isNull(nConvert)) bError = 1;
    else apiWAL.setnConvert(ropWorkAssignmentLoad, nConvert.toBigDecimal());

    // nQtyBase
        const idGds = parseId(gidGds);
        if (nQty != "ПУСТОЕ" and isNotNull(idGds) and isNotNull(idMsr)) {
                const nQtyBase = nQty.toBigDecimal() * nConvert;
                if (isNull(nQtyBase)) bError = 1;
                else {
                        // Округление
                        const nvRoundPlaces = Bs_GoodMsrItemApi.getGdsnRoundPlaces(idGds,
↪ idMsr);
                        const nvRoundPlacesToInt = round(nvRoundPlaces, 0).toString().
↪toLong().intValue();
                        const nQtyBaseNew = round(nQtyBase, nvRoundPlacesToInt);
                        apiWAL.setnQtyBase(ropWorkAssignmentLoad, nQtyBaseNew);
                }
        }

    const sCons = avCellValue.get(10).asNString();
    if (sCons != "ПУСТОЕ") apiWAL.setsCons(ropWorkAssignmentLoad, sCons);

    const idCons = getIdCons(sCons);
    if (isNotNull(idCons)) apiWAL.setidCons(ropWorkAssignmentLoad, idCons);

    const sDate = avCellValue.get(11).asNString();
    if (sDate != "ПУСТОЕ") apiWAL.setdDate(ropWorkAssignmentLoad, ExcelStrToDate(sDate));

    //Размещение затрат
    const nCostAllocation = avCellValue.get(12).asNString();
    if (nCostAllocation != "ПУСТОЕ") apiWAL.setnCostAllocation(ropWorkAssignmentLoad,␣
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
↪nCostAllocation.toBigDecimal());

    const sTechCardDoc = avCellValue.get(13).asNString();
    if (sTechCardDoc != "ПУСТОЕ") apiWAL.setsTechCardDoc(ropWorkAssignmentLoad,␣
↪sTechCardDoc);

        const sTechCardDocName = avCellValue.get(14).asNString();
    if (sTechCardDocName != "ПУСТОЕ") apiWAL.setsTechCardDocName(ropWorkAssignmentLoad,␣
↪sTechCardDocName);

        const nRowOper = avCellValue.get(15).asNString();
    if (nRowOper != "ПУСТОЕ") apiWAL.setnRowOper(ropWorkAssignmentLoad, nRowOper.
↪toBigDecimal());

        const sOperDesc = avCellValue.get(16).asNString();
    if (sOperDesc != "ПУСТОЕ") apiWAL.setsOperDesc(ropWorkAssignmentLoad, sOperDesc);

        const sOper = avCellValue.get(17).asNString();
    if (sOper != "ПУСТОЕ") apiWAL.setsOper(ropWorkAssignmentLoad, sOper);

    const idOper = getIdOper(sOper);
    if (isNull(idOper) && sOper != "ПУСТОЕ") bError = 1;
    else apiWAL.setidOper(ropWorkAssignmentLoad, idOper);

    const dPlanBeginOper = avCellValue.get(18).asNString();
    if (dPlanBeginOper != "ПУСТОЕ") apiWAL.setdPlanBeginOper(ropWorkAssignmentLoad,␣
↪ExcelStrToDate(dPlanBeginOper));

        const dPlanEndOper = avCellValue.get(19).asNString();
    if (dPlanBeginOper != "ПУСТОЕ") apiWAL.setdPlanEndOper(ropWorkAssignmentLoad,␣
↪ExcelStrToDate(dPlanEndOper));

        if (isNotNull(idGds) and isNotNull(idStock) and isNotNull(idBisObj)) {
      const idAcc = Gds_ValLevelApi.getAcc(
                    idpGds = idGds,
                    idpStock = idStock,
                    idpValKind = new ('ru.bitec.app.gtk.lang.NLong', null),
                    idpAccountType = Bs_AccountTypeApi.findByMnemoCode("Account"),
                    idpAccKind = Bs_AccKindApi.idAcc(),
                    idpDepOwner = Bs_BisObjApi.load(idBisObj).idDepOwner
                  );
      if (isNotNull(idAcc)) {
            const sBC = Bs_AccApi.getMnemoCode(idAcc);
            apiWAL.setsBC(ropWorkAssignmentLoad, sBC);
          }
    }
        if(isNull(ropWorkAssignmentLoad.sBC)) bError = 1;

    const sFile = sFile;
    apiWAL.setsFile(ropWorkAssignmentLoad, sFile);

    const sSheet = sSheetName;
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    apiWAL.setsSheet(ropWorkAssignmentLoad, sSheet);

    if (bError != 0) apiWAL.setbError(ropWorkAssignmentLoad, 1.toBigDecimal());

    i = i + 1;
    commit();
  }

  return true;
}

lib("Btk_XlsxLib").uploadParseFile(fun);
```

Трансформация проводок между учетами

Используется для запуска трансформации проводок между учетами по выбранному документу проводок. Скрипт отбирает проводки SQL-запросом, загружает найденные записи через API и передает их в обработчик трансформации.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретный документ проводок. Перед запуском укажите актуальный идентификатор документа в условии `t.idTransDoc = ....`

Тип: JEXL-скрипт

```
const svSelectTransId = "select t.id from Act_Trans t where t.idTransDoc = 3955";
const ropaTrans = [...];
sql(svSelectTransId).foreach(rvx -> {
  ropaTrans.add(Act_TransApi.load(rvx.id));
});

if (!ropaTrans.isEmpty()) Act_TransferPkg.processTrans(asScala(ropaTrans).toSeq());
```

Обновление дат документов и проводок по списку GID из XLSX

Используется для массового обновления дат документов и связанных с ними проводок по списку GID из XLSX-файла. Скрипт читает значения из первого столбца файла Excel, находит соответствующие документы проводок, устанавливает для них фиксированную дату и обновляет даты связанных проводок.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную задачу. Перед запуском укажите актуальную дату в переменной `dvDate`, проверьте имя листа Excel и убедитесь, что GID документов указаны в первом столбце

файла.

Тип: JEXL-скрипт

```
var GetStringValue = function(row, Column)  {
     row.getCell(Column) == null ? null : toString((row.getCell(Column)).getCellType())␣
↪== "NUMERIC" ? toString(row.getCell(Column).getNumericCellValue()) : toString(row.
↪getCell(Column)).trim();
  }

var fun = x -> {
  var sheet = x.getSheet("Лист1");
  var lastRowNum = sheet.getLastRowNum();
  var i =  1;
  var dvDate = toDate("31.01.2025", "dd.MM.yyyy");
  while (i <= lastRowNum ){
    var row = sheet.getRow(i);
    var gidDoc = GetStringValue(row, 0);
                var s = `select t.id
                                            from act_transdoc t
                                            where t.gidsrc  = '` + gidDoc + `'
                                            `;
                var l = sql(s).asList();
                for(w:l){
                                var ropTr = Act_TransDocApi.load(w.id);
                                Act_TransDocApi.setdDoc(ropTr, dvDate);
                                var s2 = `select t.id
                                                        from act_trans t
                                                        where t.idTransDoc  = ` + ropTr.id
                                                        ;
                                                        var l2 = sql(s2).asList();
                                for(r:l2){
                                                        var rop1 = Act_TransApi.load(r.id);
                                                        Act_TransApi.setdDate(rop1, dvDate);
                                }
                                commit();
                }
                    i = i+1;
   }

commit();
true;
}

lib("Btk_XlsxLib").uploadParseFiles(fun);
```

Замена контрагента договора с обновлением связанных данных

Используется для массовой корректировки контрагента по договору и связанным с ним данным. Скрипт заменяет контрагента договора на головного контрагента, добавляет стороны договора, обновляет связанные платежные документы, регистры, объекты баланса, оплаты, акты, счета-фактуры и проводки.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретный договор и структуру данных. Перед запуском укажите актуальный идентификатор договора в переменной `idvCnt`, проверьте стороны договора в вызовах `Cnt_SideApi.findByMnemoCode(...)` и убедитесь, что прямые SQL-обновления соответствуют задаче. Скрипт изменяет данные сразу в нескольких связанных объектах.

Тип: JEXL-скрипт

```
const idvCnt = 49740L;//id договора
const ropCnt = Cnt_ContractApi.load(idvCnt);
const ropContr = Bs_ContrasApi.load(parseId(ropCnt.gidOtherSettler));
if (isNull(ropContr.idCorporation)) {
  raise("Не найден головной контрагент");
}
const gidvOldSettler = ropCnt.gidOtherSettler;
const gidvNewSettler = Bs_ContrasApi.getGid(ropContr.idCorporation);
if (ropCnt.gidOtherSettler == gidvNewSettler) {
  raise("В договоре уже стоит головной контрагент");
}


var rop_Cnt_ContractSide1 = Cnt_ContractSideApi.insertByParent(ropCnt);
Cnt_ContractSideApi.setidSide(rop_Cnt_ContractSide1, Cnt_SideApi.findByMnemoCode(
↪"consignor")); //!!!подставить нужное!!!
Cnt_ContractSideApi.setgidSettler(rop_Cnt_ContractSide1, gidvOldSettler);
var rop_Cnt_ContractSide2 = Cnt_ContractSideApi.insertByParent(ropCnt);
Cnt_ContractSideApi.setidSide(rop_Cnt_ContractSide2, Cnt_SideApi.findByMnemoCode(
↪"receiver")); //!!!подставить нужное!!!
Cnt_ContractSideApi.setgidSettler(rop_Cnt_ContractSide2, gidvOldSettler);
session.flush();

tsql(`update Cnt_Contract c set gidOtherSettler = '`+gidvNewSettler+`' where c.id =␣
↪`+idvCnt).execute();
tsql(`update Cnt_Contract c set gidOtherSettler = '`+gidvNewSettler+`' where c.
↪idParentContract = `+idvCnt).execute();
//tsql(`update Cnt_ContractSide c set gidSettler = '`+gidvNewSettler+`' where c.bMain =␣
↪1 and c.idContract = `+idvCnt).execute();

var func = function() {
  var sQ0 = `
                        select t.id
                        from Cnt_ContractSide t
                        where  t.bMain = 1 and t.idContract = `+idvCnt;
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        var qq = sql(sQ0).asList();
        for (w: qq) {
          const rop = Cnt_ContractSideApi.load(w.id);
          Cnt_ContractSideApi.setgidSettler(rop, gidvNewSettler);
        }
}
Cnt_ContractApi.forDisabledTaxType(func);

//график платежей
tsql(`update Cnt_PaySchedule c set gidSettler = '`+gidvNewSettler+`' where c.
↪idMainContract = `+idvCnt).execute();
//все ДПП
tsql(`update Pm_ExpectedPay c set gidSettler = '`+gidvNewSettler+`' where c.idContract =␣
↪`+idvCnt).execute();
//все ЗнП
tsql(`update pm_payrequest c set gidSettlerPayFor = '`+gidvNewSettler+`' where c.
↪idContract = `+idvCnt).execute();
tsql(`update Pm_PayRequestSumDistrJournal c set gidSettlerPayFor = '`+gidvNewSettler+`'␣
↪where c.idContract = `+idvCnt).execute();
//все ПП
tsql(`update Pm_OrderBankOut c set gidSettlerPayFor = '`+gidvNewSettler+`' where c.
↪idContract = `+idvCnt).execute();
//ПМ регистр
tsql(`update Pm_PayMoveReg c set gidSettler = '`+gidvNewSettler+`' where c.idContract =␣
↪`+idvCnt).execute();
//Объекты баланса
tsql(`update Pm_BalObj c set gidSettler = '`+gidvNewSettler+`' where c.idContract =␣
↪`+idvCnt).execute();
//ключи поисковые обновить
var sQ1 = `
select t.id
from Pm_BalObj t
where  t.idContract = `+idvCnt
;
var l1 = sql(sQ1).asList();
for (w: l1) {
  const rop = Pm_BalObjApi.load(w.id);
  Pm_BalObjApi.setsFindKey(rop, Pm_BalObjApi.gensFindKey(w.id));
}
flush();

tsql(`update Pm_PayMoveReg c set sFindKey = t.sFindKey from Pm_BalObj t where c.idBalObj␣
↪= t.id and c.idContract = `+idvCnt).execute();
tsql(`update Pm_PayMoveReg c set sFindKeyFrom = t.sFindKey from Pm_BalObj t where c.
↪idBalObjFrom = t.id and c.idContract = `+idvCnt).execute();

//оплаты
tsql(`update Pm_PayMove c set gidSettlerFrom = '`+gidvNewSettler+`' where c.
↪idContractFrom = `+idvCnt).execute();
tsql(`update Pm_PayMove c set gidSettlerTo = '`+gidvNewSettler+`' where c.idContractTo =␣
↪`+idvCnt).execute();
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
var sQ2 = `
select t.id
from Pm_PayMove t
where  t.idContractFrom = `+idvCnt+`
union
select t.id
from Pm_PayMove t
where  t.idContractTo = `+idvCnt
;
var l2 = sql(sQ2).asList();
for (w: l2) {
  const rop = Pm_PayMoveApi.load(w.id);
  Pm_PayMoveApi.refreshFindKeyFrom(rop);
  Pm_PayMoveApi.refreshFindKeyTo(rop);
}
flush();
//акты
tsql(`update Stm_ActIn c set gidSettler = '`+gidvNewSettler+`' where c.idContract =␣
↪`+idvCnt).execute();
// с.ф.
tsql(`update Stm_InvBillDocIn c set gidSettler = '`+gidvNewSettler+`' from stm_actin a␣
↪where c.id = a.idInvBillDocIn and a.idContract = `+idvCnt).execute();
tsql(`update Stm_BookInclSetting c set gidSettler = a.gidSettler from Stm_InvBillDocIn a␣
↪where c.gidDoc = a.gid and a.gidSettler = '`+gidvNewSettler+`'`).execute();
//проводки
var sQ3 = `
select distinct t.idtrans id
from act_transreg t
where  t.idContract = `+idvCnt+`
and    t.gidSettler = '`+gidvOldSettler+`'
and    t.bCredit = 1
`;
var l3 = sql(sQ3).asList();

for (w: l3) {
  const rop = Act_TransApi.load(w.id);
  Act_TransApi.setAttrCr(rop,"gidSettlerBranch",gidvOldSettler);
  Act_TransApi.setAttrCr(rop,"gidSettler",gidvNewSettler);
}
flush();
var sQ4 = `
select distinct t.idtrans id
from act_transreg t
where  t.idContract = `+idvCnt+`
and    t.gidSettler = '`+gidvOldSettler+`'
and    t.bCredit = 0
`;
var l4 = sql(sQ4).asList();

for (w: l4) {
  const rop = Act_TransApi.load(w.id);
  Act_TransApi.setAttrDeb(rop,"gidSettlerBranch",gidvOldSettler);
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
  Act_TransApi.setAttrDeb(rop,"gidSettler",gidvNewSettler);
}
commit();
```

Проверка наличия объектов в БД по GID из XLSX

Используется для проверки наличия объектов в базе данных по списку GID из XLSX-файла. Скрипт читает значения из первого столбца файла Excel, ищет соответствующие записи в БД и выводит в лог GID, по которым записи не найдены.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретный объект и структуру файла. Перед запуском проверьте имя листа Excel, номер столбца с GID и SQL-запрос, по которому выполняется поиск записей в БД.

Тип: JEXL-скрипт

```
var fun = x -> {
  var sheet = x.getSheet("Sheet1");
  var lastRowNum = sheet.getLastRowNum();
  var i = 1;
  var gidavNewTransDoc = "";
  while (i <= lastRowNum ){
    const vRow = sheet.getRow(i);

    var gidvTransDoc = vRow.getCell(0).getStringCellValue(); //Получаем данные из файла

        var gidvTransDocEx = sql(`select td.gid from Act_TransDoc td where td.gid = '$
↪{gidvTransDoc}'`).asList(); //Ищем совпадение в БД

    if (gidvTransDocEx.isEmpty()) {
      gidavNewTransDoc += gidvTransDoc + ', ';
    }
    i = i+1;
  }
  audInfo('Наличие записей в БД по gid проверено.'); // Запись в лог выполнения JEXL␣
↪скрипта
  Btk_InfoLogPkg.info(gidavNewTransDoc); // Вывод в лог сообщений
  return true;
}

lib("Btk_XlsxLib").uploadParseFiles(fun);
```

Загрузка записей из XLSX под заданного родителя

Используется для массового создания записей на основе данных из XLSX-файла. Скрипт читает код и наименование из строк Excel, создаёт новые объекты, задаёт для них родительскую запись и наследует часть параметров от родителя.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретный класс и структуру файла. Перед запуском укажите актуальный идентификатор родительского объекта, проверьте имя листа Excel и номера столбцов, из которых читаются значения.

Тип: JEXL-скрипт

```
var GetStringValue = function(row, Column) {
     row.getCell(Column) == null ? null : toString((row.getCell(Column)).getCellType())␣
↪== "NUMERIC" ? toString(row.getCell(Column).getNumericCellValue()) : toString(row.
↪getCell(Column)).trim();

  }
var ropAcc = Bs_AccApi.load(7752L);
var fun = x -> {
  var sheet = x.getSheet("Лист1");
  var lastRowNum = sheet.getLastRowNum();
  var i = 1;
  while (i <= lastRowNum ){
    var row = sheet.getRow(i);
    var scode = GetStringValue(row, 0);
    var scaption = GetStringValue(row, 1);
    var rop = Bs_AccApi.insert();
    Bs_AccApi.setidParent(rop, ropAcc.id) ;
    //Bs_AccApi.setidAdjustMethod(rop, 2L );
    Bs_AccApi.setsCode(rop, scode);
    Bs_AccApi.setsCaption(rop, scaption);
    Bs_AccApi.setidAccType(rop, ropAcc.idAccType);
    Bs_AccApi.setidAccGroup(rop, ropAcc.idAccGroup) ;
    i = i+1;

   }

commit();

true;

}

lib("Btk_XlsxLib").uploadParseFiles(fun);
```

Создание объектов по результатам SQL-выборки

Используется для массового создания объектов на основе данных, отобранных SQL-запросом. Скрипт получает список исходных записей, рассчитывает дополнительное значение для каждой строки и создаёт новые объекты с заданным периодом действия и параметрами.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретный класс и условия отбора. Перед запуском проверьте SQL-запрос, фиксированные даты, идентификаторы объектов и правила расчёта значений, которые записываются в создаваемые объекты.

Тип: JEXL-скрипт

```
var s = `select t.*,
           case when t.rn <=1500 then '12000000'
                when t.rn >1500 and t.rn <=3000  then '14000000'
                when t.rn >3000 and t.rn <=4500  then '15000000'
                when t.rn >4500 and t.rn <=6000  then '17000000'
                when t.rn >6000 and t.rn <=7500  then '18000000'
                when t.rn >7500 and t.rn <=9000  then '19000000'
                when t.rn >9000 and t.rn <=10500  then '20000000'
                when t.rn >10500 and t.rn <=12000  then '21000000'
                when t.rn >12000 and t.rn <=13500  then '22000000'
                when t.rn >13500 and t.rn <=15000  then '13000000' end as ss
from (
select i.id, ROW_NUMBER() OVER () AS rn
from Asf_TaxPropertyCalcAvg d
join bs_invnumb i on i.id = d.idinvnumb
where d.iddoc  = 52
) t
`;
var a = toDate("01.01.2000 12:42:10");
var b = toDate("31.12.2025 12:42:10");
var l = sql(s).asList();
for(w:l){
 var rvRop = Asf_TaxPropertyInvNumbSpecialApi.insert();
  Asf_TaxPropertyInvNumbSpecialApi.setdFrom(rvRop, a);
  Asf_TaxPropertyInvNumbSpecialApi.setdTo(rvRop, b);
  Asf_TaxPropertyInvNumbSpecialApi.setidInvNumb(rvRop, w.id);
  Asf_TaxPropertyInvNumbSpecialApi.setidPropertyLoc(rvRop, 2L);
 Asf_TaxPropertyInvNumbSpecialApi.setsOKTMO(rvRop, w.ss);
}
```

Заполнение, проведение и формирование проводок по документам за период

Используется для массовой обработки документов, найденных по задаче расчёта и периоду. Скрипт отбирает документы по параметрам исходной задачи, заполняет каждый документ, переводит его в итоговое состояние и запускает формирование проводок.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную задачу и тип документов. Перед запуском укажите актуальный идентификатор задачи в условии `t.id = ...`, проверьте условия отбора документов, правила перехода состояния и настройки формирования проводок.

Тип: JEXL-скрипт

```
//выполнение проводок
for (let rvx: sql(`select t.id from Asf_DeprCalcTask t where t.id = 255`).asList()) {
  const idvDoc = rvx.id;
  const vDocApi = Asf_DeprCalcTaskApi;
  const ropDoc = vDocApi.load(idvDoc);

  const vDepDocApi = Asf_DeprDocApi;
  const idvDeprDocSummarySC = vDepDocApi.idSummarySC();

  const idavBisObj = asJava(Asf_DeprCalcTaskBisObjApi.byParent(ropDoc).map(rop -> rop.
↪idBisObj).toSet());
  const idavAccKind = asJava(Asf_DeprCalcTaskAccKindApi.byParent(ropDoc).map(rop -> rop.
↪idAccKind).toSet());

  const mkSqlDate = (dp) -> `'${toString(dp, "dd.MM.yyyy")}'::timestamp`;
  const svSqlFromDate = mkSqlDate(ropDoc.dFrom);
  const svSqlToDate = mkSqlDate(ropDoc.dTo);

  const makeTrans = (ropDoc) -> {
    const ropaSortedTots = Bs_TransObjTypeSettingApi
      .getByObjectTypeActive(ropDoc.idObjectType).sortWith((ropA, ropB) -> ropA.
↪nStateOrder < ropB.nStateOrder);
    const gidvDoc = ropDoc.gid();
    const avParam = asScala({:});
    ropaSortedTots.foreach(ropTots -> {
     // Btk_InfoLogPkg.info(toString(ropTots.nStateOrder));
      Bs_TransPkg.procTransDoc(
        gidvDoc, idpTransSetting = ropTots.id,
        spOperationType = Bs_TransPkg.sMakeOperType(),
        apParam = avParam,
        bpWillValidateTrans = false
      );
    });
  };

  for (let idvBisObj: idavBisObj) {
    for (let idvAccKind: idavAccKind) {
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
      sql(`
        select t.id
          from Asf_DeprDoc t
         where t.idBisObj = ${idvBisObj}
           and t.idAccKind = ${idvAccKind}
           and t.dTrans between ${svSqlFromDate} and ${svSqlToDate}
           and t.idStateMC >= 100
           and t.idSubClass = ${idvDeprDocSummarySC}
      order by t.dTrans
      `).foreach(rvx -> {
        const ropDeprDoc = vDepDocApi.load(rvx.id);
        vDepDocApi.fillDoc(ropDeprDoc);
        commit();
        vDepDocApi.setidState(ropDeprDoc, vDepDocApi.idDoneState());
        commit();
        makeTrans(ropDeprDoc);
        commit();
      });
    }
  }
}
```

Couter

Используется для массовой обработки набора записей, найденных запросом или обработанных по пачкам.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var nvCount = sql(`
        select count(*) as "nCount"
        from mct_techprocnorm
        where sOrder is null
        and sPosNumber is not null
        `).asSingle().nCount;
var nvIter = nvCount / 5000 + 1;
for (i : (1 .. nvIter)){
        var ropList =  sql(`
                select t.id as "id"
                from mct_techprocnorm t
                where t.sOrder is null
                and t.sPosNumber is not null
                order by t.id
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                limit 5000
                `).batchObjLoad(Mct_TechProcNormApi, "id");
        for (rop : ropList){
                Mct_TechProcNormApi.updateOrderAttr(rop);
        }
        commit();
}
```

Jexl дата последнего изменения Qj Question с датой создания если пустые dmodifydate dz

Используется для массовой обработки набора записей, найденных запросом или обработанных по пачкам.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var nvCount = sql(`select count(*) as "nCount" from qj_question qq where qq.dmodifydate␣
↪is null`).asSingle().nCount;

var nvIter = nvCount / 1000 + 1;

for (i : (1 .. nvIter)){
        var ropList =  sql(`
        with t as (select distinct greatest(qq.dmodifydate_dz, qq.dcreatedate_dz, qf.
↪dmodifydate_dz, qf.dcreatedate_dz, qa.dmodifydate_dz, qa.dcreatedate_dz) as dd, qq.id
                                from qj_question qq
                                left join qj_questionfile qf on qf.idquestion = qq.id
                                left join qj_answer qa on qa.idquestion = qq.id
                                where qq.dmodifydate is null
                                order by dd desc)
                select t.id, max(t.dd) date
                from t
                group by t.id
                limit 1000
                `);
                ropList.batchObjLoad(Qj_QuestionApi, "id");
                ropList.foreach(function (r){
                var rop = Qj_QuestionApi.load(r.id);
                Qj_QuestionApi.setdModifyDate(rop, r.date)
                });
        commit();
}
```

Jexl дата последнего изменения Qj Question

Используется для массовой обработки набора записей, найденных запросом или обработанных по пачкам.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var nvCount = sql(`select count(*) as "nCount" from qj_question qq where qq.dmodifydate␣
↪is null`).asSingle().nCount;

var nvIter = nvCount / 1000 + 1;

for (i : (1 .. nvIter)){
        var ropList =  sql(`
        with t as (select distinct greatest(qq.dmodifydate_dz, qf.dmodifydate_dz, qa.
↪dmodifydate_dz) as dd, qq.id
                                    from qj_question qq
                                    left join qj_questionfile qf on qf.idquestion = qq.id
                                    left join qj_answer qa on qa.idquestion = qq.id
                                    where coalesce(qq.dmodifydate_dz, qf.dmodifydate_dz,␣
↪qa.dmodifydate_dz) is not null and qq.dmodifydate is null
                                    order by dd desc)
                select t.id, max(t.dd) date
                from t
                group by t.id
                limit 1000
                `);
                ropList.batchObjLoad(Qj_QuestionApi, "id");
                ropList.foreach(function (r){
                var rop = Qj_QuestionApi.load(r.id);
                Qj_QuestionApi.setdModifyDate(rop, r.date)
                });
        commit();
}
```

Очистка Doc State Doc копия

Используется для массовой обработки набора записей, найденных запросом или обработанных по пачкам.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var ropList =  sql(`
        select id
        from Stk_RegistryMoveType
        `).batchObjLoad(Stk_RegistryMoveTypeApi, "id");
for (rop : ropList){
        var aro = rop.copyAro();
        var svHeadLine = aro.sCaption();
        var svMnemoCode = aro.sCode();
        var ropObj = Btk_ObjectApi.insertByParent(rop);
        Btk_ObjectApi.setsHeadLine(ropObj, svHeadLine);
        Btk_ObjectApi.setsMnemoCode(ropObj, svMnemoCode);
}
commit();
```

Couter

Используется для массовой обработки набора записей, найденных запросом или обработанных по пачкам.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var nvCount = sql(`
        select count(*) as "nCount"
        from mct_techprocnorm
        where sOrder is null
        and sPosNumber is not null
        `).asSingle().nCount;
var nvIter = nvCount / 100 + 1;
for (i : (1 .. nvIter)){
        var ropList =  sql(`
                select t.id as "id"
                from mct_techprocnorm t
                where t.sOrder is null
                and t.sPosNumber is not null
                order by t.id
                limit 100
                `).batchObjLoad(Mct_TechProcNormApi, "id");
        for (rop : ropList){
                Mct_TechProcNormApi.updateOrderAttr(rop);
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        }
        commit();
}
```

Offset Batch Obj Load

Используется для массовой обработки набора записей, найденных запросом или обработанных по пачкам.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
for (i : (1 .. 100)){
        var nvOffset = (i-1) * 500
        var ropList =  sql(`
                select g.id as "id" from gds_articlecontras g
                order by g.id
                `
                + "offset " + nvOffset + " limit 500").batchObjLoad(Gds_
↪ArticleContrasApi, "id");
        for (rop : ropList){
                Gds_ArticleContrasApi.delete(rop);
        }
        commit();
}
```

Обновление SP Emptys Position

Используется для массовой обработки набора записей, найденных запросом или обработанных по пачкам.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
for (i : (1 .. 100)){
        var nvOffset = (i-1) * 1000
        var ropList =  sql(`
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                select d.id as "id" from Mct_SpecificationVerPos d
                where sPosition is null
                order by d.id
                `
                + "offset " + nvOffset + " limit 1000").batchObjLoad(Mct_
↪SpecificationVerPosApi, "id");
        for (rop : ropList){
                var svPosition = "[G]"+rop.idJ();
                Mct_SpecificationVerPosApi.setsPosition(rop, svPosition);
        }
        commit();
}
```

7.9 Печатные формы

Запуск печатной формы по кнопке

Используется для привязки вызова печатной формы к пользовательской кнопке через обозреватель проекта. При нажатии запускается построение отчёта с передачей идентификаторов текущего объекта.

Место применения: `Сущности > Обозреватель проекта > [Объект] > Редактор операций > [JEXL-операция] > Вкладка "JEXL-скрипт"`

Тип: JEXL-скрипт

```
var idvOrderClass = selection ("idClass");
var idvOrder = selection.getVar ("id");
var NLong= function (number) { // локальная функция для преобразования Long в NLong
    return new ("ru.bitec.app.gtk.lang.NLong", number);
                                };
var spReportName = "Stm_InvoiceOut";
var dpReportVersionDate = sysDate ();
var vPostBuildAction = session
.sbtClassLoader()
.loadClass('ru.bitec.app.gtk.gl.postbuildaction.PostBuildAction')
.design();
var propertyMap = {
                    "idSrcObject" : NLong (idvOrder),
                    "idSrcClass"  : NLong (idvOrderClass)
                    };
var idObjectTypePrintForm = null;
Rpt_Lib.createReportExJexl (
spReportName,
dpReportVersionDate,
vPostBuildAction,
asScala (propertyMap),
idpObjectTypePrintForm
);
```

Формирование имени файла печатной формы

Используется для формирования понятного и структурированного имени файла при экспорте печатной формы. Скрипт заменяет системные или неинформативные значения имени файла и может использовать параметры связанного документа или сущности выполнения отчёта.

Место применения: `Отчёты > Печатные формы > [Печатная форма] > Редактировать > Режим формирования имени файла: JEXL > Вкладка "JEXL для формирования имени файла"`

Примечание

Все параметры, которые были переданы в отчёт для построения, доступны по соответствующим именам в верхнем регистре. Например, если в файле отчёта используется параметр `idBisObj`, то в скрипте к нему можно обратиться как к `IDBISOBJ`.

Внимание

Скрипт рассчитан на формирование имени отчёта при построении из отчётной формы `Rpt_Entity`. Он не будет работать при формировании печатной формы, если в контексте отсутствует `IDENTITYEXEC`.

Тип: JEXL-скрипт

```
var ropEntityExec = null;
if ((IDSRCCLASS) == Wf_DocApi.idClass()) {
  var gid = Wf_DocApi.getGidLastVer(IDSRCOBJECT);
  ropEntityExec = Rpt_EntityExecApi.getByGidDocVer(gid);
} else {
  ropEntityExec = Rpt_EntityExecApi.load(IDENTITYEXEC);
}
var jParams = toJObject(ropEntityExec.jParams);
var nvMonth = " ";
if (isNotNull(jParams.getLong("flt_dperiodmonth"))) {
   nvMonth = toString(Clr_MonthApi.load(jParams.getLong("flt_dperiodmonth")).sCode);
}
"СНГ-Р1_№" + nvl(jParams.getString("flt_snumref"), " ") + "_за_" + nvMonth + "_" +␣
↪nvl(toString(jParams.getNumber("flt_dperiodyear")), " ") + "_для_" + nvl(jParams.
↪getString("flt_idbisobjmc"), " ");
```

7.10 Управление потребностями

Коррекция ссылок на источник и объект потребности в проводке закупки

Используется для ручного обновления ссылок на источник (`idSrcTrans`) и объект потребности (`idNeedObj`) в проводках закупок. Применяется при постобработке после миграции или ошибок настройки.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

В текущем виде скрипт содержит захардкоженные ID. Перед использованием замените их на актуальные значения.

Тип: JEXL-скрипт

```
const avRow = sql(`
    select t.id, t.idsrctrans, t.idneedobj
    from prs_purchtrans t
    where t.gidsrcdetobj = '48051/77672'
`).asList();

for (r : avRow) {
  ropPT = Prs_PurchTransApi.load(r.id);
  Prs_PurchTransApi.setidSrcTrans(ropPT, 187367L);
  Prs_PurchTransApi.setidNeedObj(ropPT, 121832L);
  Prs_PurchTransApi.recalcQtyOpenBase(r.idsrctrans, true, true);
  Prs_TransPkg.reCalcNeedObjQtyOther(r.idneedobj, true, true);
}
```

Синхронизация количества для заявок на услуги

Используется для синхронизации количества в объектах потребности и проводках закупок с суммой без НДС из позиции заявки. Применяется для заявок на услуги, где объем определяется финансово. При наличии связи с договором также пересчитывает спецификацию.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
const avRow = sql(`
select det.id as iddet
      ,n.id as idneedobj
      ,pt.id as idpurchtrans
from Prs_PurchReqDet det
join Prs_PurchReqVer ver on ver.id = det.idPurchReqVer
join Prs_PurchReq req on req.id = ver.idPurchReq
join Prs_NeedObj n on det.idPurchReqPos = getGidId(n.gidDetNeed)
join Prs_PurchTrans pt on pt.idNeedObj = n.id
join Btk_ObjectType ot on req.idObjectType = ot.id
where ot.sCode = 'Prs_PurchReqCC'
and coalesce(det.nPriceRate, 0) <> 0
order by det.id
`).asList();

for (r : avRow) {
  var ropDet = Prs_PurchReqDetApi.load(r.iddet);
  var ropNeedObj = Prs_NeedObjApi.load(r.idneedobj);
  var ropPurchTrans = Prs_PurchTransApi.load(r.idpurchtrans);

  if (parseIdClass(ropPurchTrans.gidSrcDetObj) == Cnt_ContractSpecApi.idClass()) {
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    // Обновляем объект потребности, а так же проводку
    if (ropDet.nTotalSumNoTax != ropNeedObj.nQtyBase) {
      Prs_NeedObjApi.setnQtyBase(ropNeedObj, ropDet.nTotalSumNoTax);
    }
    if (ropDet.nTotalSumNoTax != ropPurchTrans.nQtyBase) {
      Prs_PurchTransApi.setnQtyBase(ropPurchTrans, ropDet.nTotalSumNoTax);
      // Обновляем позицию спецификации
      var ropCntSpec = Cnt_ContractSpecApi.loadByGid(ropPurchTrans.gidSrcDetObj);
      Prs_ContractPkg.recalcQtyByPurchTrans(ropCntSpec);
    }
  }
  session.flush();
};

session.commit();
```

Ручное распределение позиции акта по проводкам закупки

Используется для ручного распределения количества из позиции приходного акта по нескольким проводкам закупки. Для каждой проводки рассчитывается доступный объем и фиксируется привязка к объекту потребности.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

В текущем виде скрипт содержит захардкоженные ID. Перед использованием замените их на актуальные значения.

Тип: JEXL-скрипт

```
const ropActDet = Stm_ActInDetApi.load(136129L);
var nQty = ropActDet.nQty;
const ropPT1 = Prs_PurchTransApi.load(301653L);
const nQty1 = nQty.min(ropPT1.nQty);
Stmprs_ActInPkg.setObjNeedDetQtyBase(ropActDet.idDoc, ropActDet.gid, nQty1, ropPT1.
↪idNeedObj, ropActDet.idMsr, ropPT1.id);
nQty = nQty - nQty1;
Btk_InfoLogPkg.info(nQty.toString());
const ropPT2 = Prs_PurchTransApi.load(476615L);
const nQty2 = nQty.min(ropPT2.nQty);
Stmprs_ActInPkg.setObjNeedDetQtyBase(ropActDet.idDoc, ropActDet.gid, nQty2, ropPT2.
↪idNeedObj, ropActDet.idMsr, ropPT2.id);
nQty = nQty - nQty2;
Btk_InfoLogPkg.info(nQty.toString());
const ropPT3 = Prs_PurchTransApi.load(476623L);
const nQty3 = nQty.min(ropPT.nQty);
Stmprs_ActInPkg.setObjNeedDetQtyBase(ropActDet.idDoc, ropActDet.gid, nQty3, ropPT3.
↪idNeedObj, ropActDet.idMsr, ropPT3.id);
nQty = nQty - nQty3;
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
Btk_InfoLogPkg.info(nQty.toString());
const ropPT4 = Prs_PurchTransApi.load(3016666L);
const nQty4 = nQty.min(ropPT4.nQty);
Stmprs_ActInPkg.setObjNeedDetQtyBase(ropActDet.idDoc, ropActDet.gid, nQty4, ropPT4.
↪idNeedObj, ropActDet.idMsr, ropPT4.id);
nQty = nQty - nQty4;
Btk_InfoLogPkg.info(nQty.toString());
session.commit();
```

7.11 Файлы и вложения

Добавление File By Path

Используется для работы с файлами и вложениями из JEXL-скрипта.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
var idFileStorage = Btk_FileStorageApi.findByMnemoCode("testLocal");
var filePath = "C:\Users\a.kobelev\tmp\DbLink_Postgresql.txt";
var file = new ("java.io.File", filePath)
if (!file.exists()) {
  raise("Файл " + filePath  + " не существует")
}
var is = new("java.io.FileInputStream", file);
var idFile = null;
@begin{
 idFile = Btk_FilePkg.createAndFill(file.getName(), idFileStorage, null, is)
}@exception function(e){
  is.close();
  throw(e);
} end;
 is.close();
dialogs.showMessage(idFile.toString());
```

7.12 Метаданные и типы объектов

Добавление WS State

Используется для настройки или корректировки метаданных системы: типов объектов, вкладок, состояний, операций или атрибутов.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
Btk_Pkg.setRWSharedUOWEditType();
var idvClass = Mct_WorkStructureApi.idClass();
Btk_ClassStateApi.register(idvClass, "ManPowerNorming", "Нормируется трудоемкость",␣
↪bpStartState = 0B, npOrer = 250B);
commit();
```

Очистка Proc Oper Clear Spaces

Используется для настройки или корректировки метаданных системы: типов объектов, вкладок, состояний, операций или атрибутов.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
sql(`
select
        mot.id as "idMctObjectTypeSpec"
from mct_objecttype mot
        left join btk_objecttype ot on mot.idbtkobjecttype = ot.id
        left join btk_class c on ot.idrefclass = c.id
where c.sname = 'Mct_Specification'
 `).foreach(function(r){
@begin{
        var idvMctOt = r.idMctObjectTypeSpec;
        var rop = Mct_ObjectTypeApi.load(idvMctOt);
        var idvClearSpaces = Mct_ObjectTypeProcOperApi.findByMnemoCode(rop, "clearSpaces
↪");
        if (idvClearSpaces != null) {
                Mct_ObjectTypeProcOperApi.delete(Mct_ObjectTypeProcOperApi.
↪load(idvClearSpaces));
        }
  }
@exception
function(exp){
        println("[JEXL Exception] " + exp.getCause());
}end;
}) ;
commit();
```

Миграция Bs Goods Obj Attr

Используется для настройки или корректировки метаданных системы: типов объектов, вкладок, состояний, операций или атрибутов.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvGoodsClass = Bs_GoodsApi.idClass();
var idvMeasureItemAttr = Btk_AttributeApi.findByMnemoCode(Btk_ClassApi.
↪load(idvGoodsClass), "idMeasureItem");
if (idvMeasureItemAttr != null){
Btk_Pkg.setRWSharedUOWEditType();
Btk_AttributeApi.setbObjectAttr(Btk_AttributeApi.load(idvMeasureItemAttr), 1B);
commit();
}
```

Миграция Find Gds Proc Oper

Используется для настройки или корректировки метаданных системы: типов объектов, вкладок, состояний, операций или атрибутов.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
sql(`
select
                mot.id as "idObjectType"
                ,ot.sCaption as "sCaption"
        from btk_ObjectType ot
        left join mct_objectType mot on ot.id = mot.idbtkobjecttype
        where ot.idrefclass = (select id from btk_class where sname = 'Mct_Specification
↪')
 `).foreach(function(r){
@begin{
        println(r.sCaption);
        Mct_ObjectTypeProcOperApi.register(idpMctObjectType = r.idObjectType,
      spSystemName = "findGDSByMatCode",
      spCaption = "Определение ТМЦ по кодировке контрагента",
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
      npImage = 28B,
      npOrder = 10B,
      spApplicationMethod = "Mct_SpecificationVerPosApi().findGDSByVer()",
          bpIsActive = 0B);
        Mct_ObjectTypeProcOperApi.register(idpMctObjectType = r.idObjectType,
      spSystemName = "findGdsByCodeByVer",
      spCaption = "Определение ТМЦ по коду",
      npImage = 28B,
      npOrder = 10B,
      spApplicationMethod = "Mct_SpecificationVerPosApi().findGdsByCodeByVer()",
          bpIsActive = 0B);
        Mct_ObjectTypeProcOperApi.register(idpMctObjectType = r.idObjectType,
      spSystemName = "fillGds",
      spCaption = "Определение ТМЦ",
      npImage = 28B,
      npOrder = 10B,
      spApplicationMethod = "Mct_SpecificationVerPosApi().fillGds()",
          bpIsActive = 1B);
  }
@exception
function(exp){
        println("[JEXL Exception] " + exp.getCause());
}end;
}) ;
commit();
```

Обновление Msch Pos Mixin

Используется для настройки или корректировки метаданных системы: типов объектов, вкладок, состояний, операций или атрибутов.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
sql(`
with mschOT as (select ot.id as idMctObjectType
        from btk_objecttype ot
        where ot.scode = 'UnitProduct')
select
        svp.id as "idSpecVerPos"
        ,dvp.gidref as "gidDocumentVerPos"
from mct_specificationverpos svp
        left join mct_specificationver sv on svp.idmctdocumentver = sv.id
        left join mct_specification s on sv.idmctdocument = s.id
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        left join mct_documentverpos dvp on svp.gid = dvp.gidref
where s.idobjecttype in (select idMctObjectType from mschOT)
and dvp.gidref is not null
  `).foreach(function(r){
@begin{
        var ropSVP = Mct_SpecificationVerPosApi.load(r.idSpecVerPos);
        var rvSVP = ropSVP.copyAro();
        var ropDocVerPos = Mct_DocumentVerPosApi.loadByGid(r.gidDocumentVerPos);
        Mct_DocumentVerPosApi.setgidMctDocumentVer(ropDocVerPos, rvSVP.
↪gidMctDocumentVer());
//нужно сетить gidParent
        Mct_DocumentVerPosApi.setgidParent(ropDocVerPos, Mct_SpecificationVerPosApi.
↪getGid(rvSVP.idParent()));
        Mct_DocumentVerPosApi.setidPosType(ropDocVerPos, rvSVP.idPosType());
        Mct_DocumentVerPosApi.setsPosition(ropDocVerPos, rvSVP.sPosition());
        Mct_DocumentVerPosApi.setsDesignation(ropDocVerPos, rvSVP.sDesignation());
        Mct_DocumentVerPosApi.setsCaption(ropDocVerPos, rvSVP.sCaption());
        Mct_DocumentVerPosApi.setsMatMarkProj(ropDocVerPos, rvSVP.sMatMarkProj());
        Mct_DocumentVerPosApi.setnQtyTotal(ropDocVerPos, rvSVP.nQtyTotal());
        Mct_DocumentVerPosApi.setnQty(ropDocVerPos, rvSVP.nQty());
        Mct_DocumentVerPosApi.setnNormPerUnit(ropDocVerPos, rvSVP.nNormPerUnit());
        Mct_DocumentVerPosApi.setnUseCoef(ropDocVerPos, rvSVP.nUseCoef());
        Mct_DocumentVerPosApi.setnUseCoefManual(ropDocVerPos, rvSVP.nUseCoefManual());
        Mct_DocumentVerPosApi.setnMass(ropDocVerPos, rvSVP.nMass());
        Mct_DocumentVerPosApi.setidMsr(ropDocVerPos, rvSVP.idMsr());
        Mct_DocumentVerPosApi.setidMsrNorm(ropDocVerPos, rvSVP.idMsrNorm());
        Mct_DocumentVerPosApi.setidGoods(ropDocVerPos, rvSVP.idGoods());
        Mct_DocumentVerPosApi.setsNote(ropDocVerPos, rvSVP.sNote());
        Mct_DocumentVerPosApi.setsGdsCode(ropDocVerPos, rvSVP.sGdsCode());
        Mct_DocumentVerPosApi.setnMassFull(ropDocVerPos, rvSVP.nMassFull());
        Mct_DocumentVerPosApi.setbIsolated(ropDocVerPos, rvSVP.bIsolated());
        Mct_DocumentVerPosApi.setsLocatedPlace(ropDocVerPos, rvSVP.sLocatedPlace());
        Mct_DocumentVerPosApi.setsOrderSheet(ropDocVerPos, rvSVP.sOrderSheet());
        Mct_DocumentVerPosApi.setbProtected(ropDocVerPos, rvSVP.bProtected());
        Mct_DocumentVerPosApi.setsSectionNumber(ropDocVerPos, rvSVP.sSectionNumber());
        Mct_DocumentVerPosApi.setidOrderSheet(ropDocVerPos, rvSVP.idOrderSheet());
        Mct_DocumentVerPosApi.setsCoverType(ropDocVerPos, rvSVP.sCoverType());
        Mct_DocumentVerPosApi.setsWBS(ropDocVerPos, rvSVP.sWBS());
        commit();
  }
@exception
function(exp){
        println("[JEXL Exception] " + exp.getCause());
}end;
}) ;
```

Обновление Struct OT Det Room

Используется для настройки или корректировки метаданных системы: типов объектов, вкладок, состояний, операций или атрибутов.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvStructOT = Btk_ObjectTypeApi.findByMnemoCode("stucture");
if (idvStructOT != null){
        var ropStructOT = Btk_ObjectTypeApi.load(idvStructOT);
        var idvRoomDOTet = Btk_ObjectTypeTabApi.findByMnemoCode(ropStructOT, "StructRoom
↪");
        if (idvRoomDOTet != null){
                Btk_Pkg.setRWSharedUOWEditType();
                var ropRoomOTDete = Btk_ObjectTypeTabApi.load(idvRoomDOTet);
                Btk_ObjectTypeTabApi.setsSelectionName(ropRoomOTDete, "gtk-ru.bitec.app.
↪mct.Mct_CrossLinkAvi");
                Btk_ObjectTypeTabApi.setsRepresentationName(ropRoomOTDete, "List_
↪StructureRoom");
                commit();
        }
}
```

Регистрация типа объекта для прочих ресурсов

Используется для регистрации типа объекта для ресурсов и удаления автоматически созданного типа ресурса после настройки. Скрипт относится к настройке метаданных и типов объектов.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу. Перед запуском проверьте системное имя и параметры регистрируемого типа объекта, а также необходимость удаления связанного типа ресурса.

Тип: JEXL-скрипт

```
var idvResourceClass = Rss_ResourceApi.idClass();
Btk_Pkg.setRWSharedUOWEditType();
var idvOT = Btk_ObjectTypeApi.register("Rss_Resource_Пр", "Прочие ресурсы","Прочие␣
↪ресурсы",idvResourceClass,null,null,null,null,null,null,null,null,null,null,null,null,
↪null,null);
flush();
var idvRT = sql(`select id from Rss_ResourceType r where r.idbtkobjecttype = ` + idvOT).
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
↪asSingle().id;
if(idvRT != null){
        var ropRT = Rss_ResourceTypeApi.load(idvRT);
        Rss_ResourceTypeApi.delete(ropRT);
}
commit();
```

7.13 Управление конфигурацией

Удаление дубликатов записей после переноса конфигурации

Используется для удаления избыточных дублирующихся записей, которые могут возникнуть при некорректной настройке ключа сопоставления в задачах переноса конфигурации. Скрипт оставляет последнюю версию записи по максимальному `id` для каждого уникального значения ключевого поля и удаляет остальные.

Место применения: `Настройки и сервисы > Управление конфигурацией > Менеджер конфигурации > [Нужная конфигурация] > Вкладка "Задачи" > [Нужная задача] > Вкладка "Состав задачи" > Тип: JEXL-скрипт`

Внимание

Требует адаптации под конкретный класс и ключ сопоставления. Перед запуском проверьте имя таблицы, поле группировки в `partition by` и API-класс, через который выполняется удаление записей.

Тип: JEXL-скрипт

```
var l = sql(`select t.id
from ( select id, row_number() over (
partition by idAdvRepItem
order by id desc
) as row_num from Bs_AdvRepItemDetType
) as t
where row_num > 1`).asList();
for (w:l){
Bs_AdvRepItemDetTypeApi.delete(Bs_AdvRepItemDetTypeApi.load(w.id))
}
commit();
```

Добавление Data Install Sript

Используется при подготовке, переносе или постобработке конфигурационных данных.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var list = ["Mct_DataInstallPkg.regObjectTypes",
                "Mct_DataInstallPkg.regCrossLinkType",
                "Mct_DataInstallPkg.regObjectTypeDet",
                "Mct_DataInstallPkg.regStatusPos",
                "Mct_DataInstallPkg.regActionType",
                "Mct_DataInstallPkg.regPosTypes",
                "Mct_DataInstallPkg.registerCommonVal",
                "Mct_DataInstallPkg.regRoomCat",
                "Mct_DataInstallPkg.registerLifeCycle",
                "Mct_DataInstallPkg.registerUP",
                "Sormovo_DispatcherPlan.dataInstall",
                "Mct_Eskd.dataInstall",
                "Mct_EskdPosType.dataInstall",
                "Mct_SupplyChannel.dataInstall",
                "Mct_TechComplectSheetVer.dataInstall",
                "Mct_PDSResourceType.dataInstall",
                "Mct_ObjectType.dataInstall",
                "Mct_SpecificationVer.dataInstall",
                "Mct_LaborType.dataInstall",
                "Mct_ChangeReport.dataInstall",
                "Mct_OrderSheetVer.dataInstall",
                "Mct_UnitProdSheetVer.dataInstall",
                "Mct_MatCardVer.dataInstall"];
for (sSystemName : list){
        var ropScript = Btk_DataInstallScriptApi.insert();
        Btk_DataInstallScriptApi.setsSystemName(ropScript, sSystemName);
        Btk_DataInstallScriptApi.setnVersion(ropScript, 1L);
}
commit();
```

Миграция Change State

Используется при подготовке, переносе или постобработке конфигурационных данных.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
Btk_ObjectTypeDetApi.dataInstall();
sql(`
with cls as (
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
  select sc.idmasterclass
        from btk_ClassState sc
   group by sc.idmasterclass
)
select c.id as idClass, c.sname, c.scaption
  from btk_class c
          join cls on cls.idmasterclass = c.id
 where exists (select 1 from btk_objecttype t where t.idrefclass = c.id)
 order by c.sname
 `).foreach(function(r){
@begin{
        println(r.sname);
        Btk_StateChangeApi.migrateClass(r.idClass);
        commit();
  }
@exception
function(exp){
        println("[JEXL Exception] " + exp.getCause());
}end;
}) ;
```

Обновление Proc Oper

Используется при подготовке, переносе или постобработке конфигурационных данных.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
sql(`
select
        mot.id as "idMctObjectTypeSpec"
from mct_objecttype mot
        left join btk_objecttype ot on mot.idbtkobjecttype = ot.id
        left join btk_class c on ot.idrefclass = c.id
where c.sname = 'Mct_Specification'
 `).foreach(function(r){
@begin{
        var idvMctOt = r.idMctObjectTypeSpec;
        Mct_DataInstallPkg.registerOTProcOper(idvMctOt);
  }
@exception
function(exp){
        println("[JEXL Exception] " + exp.getCause());
}end;
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
}) ;
commit();
```

7.14 Очистка и восстановление данных

Очистка Doc State Doc

Используется для очистки некорректных записей, восстановления связей или пересчета служебных данных после ошибок обработки или миграции.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var ropList =  sql(`
        select t.id from mct_documentstatedoc t
        left join mct_document tt on t.gidsrc = tt.gidref
        where tt.gidref is null
        `).batchObjLoad(Mct_DocumentStateDocApi, "id");
for (rop : ropList){
        Mct_DocumentStateDocApi.delete(rop);
}
commit();
```

Очистка Mct Doc Ver Pos For Applic

Используется для очистки некорректных записей, восстановления связей или пересчета служебных данных после ошибок обработки или миграции.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var ropList =  sql(`
        select t.id from Mct_PosNotApplicableToPV t
        left join mct_ordersheetverdet tt on t.gidsrc = tt.gid
        where tt.gid is null
        `).batchObjLoad(Mct_PosNotApplicableToPVApi, "id");
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
for (rop : ropList){
        Mct_PosNotApplicableToPVApi.delete(rop);
}
commit();
```

Очистка Old Bts Job

Используется для очистки некорректных записей, восстановления связей или пересчета служебных данных после ошибок обработки или миграции.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var ropList =  sql(`
        select id from btk_job
        where scaption in ('Рассылка сообщений', 'Архивирование сообщений', 'Обработка␣
↪исходящих сообщений')
        `).batchObjLoad(Btk_JobApi, "id");
for (rop : ropList){
        Btk_JobApi.delete(rop);
}
commit();
```

Обновление Denom Struct

Используется для очистки некорректных записей, восстановления связей или пересчета служебных данных после ошибок обработки или миграции.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Требует адаптации под конкретную базу и версию системы. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvViewType = Mct_StructViewTypeApi.findByMnemoCode("ESKD");
Mct_StructureLinkApi.refreshDenorm(idvViewType);
commit();
```

7.15 Интерфейс и операции

Подтверждение удаления объекта из списка

При нажатии на кнопку **Удалить** в списке авансовых отчетов выводит модальное окно с подтверждением операции.

Скрипт показывает пример донастройки операции через точку расширения на уровне `Avi`.

Место применения: `Обозреватель проектов > нужный объект > вкладка Расширения ДО`

Тип: JEXL-скрипт

```
var message = "Вы уверены, что хотите удалить этот авансовый ответ?";
var res = dialogs.showConfirmDialog(message);
if (!res) {
        raise ("Удаление отменено пользователем");
}
```

Открытие выборки на новой вкладке с параметрами

Открывает выборку на новой вкладке и передает в нее параметры из текущего контекста.

Скрипт показывает пример открытия другой выборки с заранее заполненными параметрами. Может использоваться в сценариях, когда из одной выборки нужно перейти в другую и сразу применить фильтры по значениям текущей записи или текущего контекста.

Место применения: `Обозреватель проектов > нужный объект > точка расширения операции или действия, из которого открывается новая выборка`

Тип: JEXL-скрипт

```
var NLong = function (number) { // локальная функция для преобразования Long в NLong
    return new ("ru.bitec.app.gtk.lang.NLong", number);
};

// Получение атрибутов с текущей выборки для использования в качестве параметров
var idvGds = selection.getVar("idGds");
var idvStock = selection.getVar("idStock");

// Проверка на заполнение атрибутов
if (empty(idvGds)) {
    raise("Не найдено значение ТМЦ в текущем контексте. Ожидается атрибут idGds.");
};
if (empty(idvStock)) {
    raise("Не найдено значение склада в текущем контексте. Ожидается атрибут idStock.");
};

// Формирование карты параметров для открытия выборки
var param = {
    "flt_idGds": NLong(idvGds),
    "flt_idaStock": NLong(idvStock),
    "flt_idaStock": toString(NLong(idvStock)),
    "flt_bShowStock": 1B,
    "flt_bShowMaster": 1B,
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    "flt_sGroupBy": "владельцам; складам"
};

// Открытие выборки на новой вкладке с параметрами
Stk_RevolutionListAvi.ListTurnGdsNew().newForm().params(param).open();
```

Заполнение бизнес-единицы по этапу договора

Заполняет бизнес-единицу приходного акта по выбранному этапу договора.

Скрипт используется в точке расширения API `Script Stm_ActInApi.setidContractStage`. В этой точке расширения параметр `value` содержит идентификатор выбранного этапа договора, а параметр `rop` — объект приходного акта, для которого выполняется изменение.

Скрипт получает бизнес-единицу из этапа договора. Если на этапе бизнес-единица не указана, скрипт пытается получить ее из родительского договора. Найденное значение записывается в приходный акт.

Место применения: `Обозреватель проектов > нужный API-класс > точка расширения API`

Тип: JEXL-скрипт

```
if (isNotNull(value)) {
  // Загрузка этапа договора по значению, переданному в точку расширения
  const ropStage = Cnt_ContractApi.load(value);
  var ropContract = null;

  // Загрузка родительского договора, если он указан на этапе
  if (isNotNull(ropStage.idParentContract)) {
    ropContract = Cnt_ContractApi.load(ropStage.idParentContract);
  }

  var idvBisObj = null;

  // Получение бизнес-единицы из этапа договора
  if (isNotNull(ropStage.idBisObj)) {
    idvBisObj = ropStage.idBisObj;
  }

  // Если на этапе бизнес-единица не заполнена, получение значения из родительского␣
↪договора
  if (isNotNull(ropContract) && isNull(idvBisObj) && isNotNull(ropContract.idBisObj)) {
    idvBisObj = ropContract.idBisObj;
  }

  // Заполнение бизнес-единицы в приходном акте
  if (isNotNull(idvBisObj)) {
    Stm_ActInApi.setidBisObj(rop, idvBisObj);
  }
}
```

Заполнение дат приходного акта по позиции спецификации договора

Заполняет даты приходного акта на основании даты поставки из позиции спецификации договора.

Скрипт используется в точке расширения API `Script Stm_ActInDetApi.setidContractSpec`. В этой точке параметр `value` содержит идентификатор позиции спецификации договора, а параметр `rop` — объект позиции приходного акта.

Если дата поставки по позиции спецификации больше даты исполнения приходного акта, скрипт обновляет дату исполнения. Если после этого дата исполнения становится больше даты приходного акта, скрипт также обновляет дату приходного акта.

Место применения: `Обозреватель проектов > нужный API-класс > точка расширения API`

Тип: JEXL-скрипт

```
if (isNotNull(value)) {
  // Загрузка приходного акта по позиции акта
  const ropAct = Stm_ActInApi.load(rop.idDoc);

  // Загрузка позиции спецификации договора по значению, переданному в точку расширения
  const ropSpec = Cnt_ContractSpecApi.load(value);

  // Обновление даты исполнения акта по дате поставки из позиции спецификации
  if (ropSpec.dDateSupply > ropAct.dDocExec) {
    Stm_ActInApi.setdDocExec(ropAct, ropSpec.dDateSupply);
  }

  // Обновление даты акта, если дата исполнения стала больше даты акта
  if (ropAct.dDocExec > ropAct.dDocIn) {
    Stm_ActInApi.setdDocIn(ropAct, ropAct.dDocExec);
  }
}
```

Подбор потребности к позиции приходного акта

Выполняет подбор потребности к позиции приходного акта.

Скрипт используется в точке расширения API `Script Stm_ActInDetApi.setnSum`. В этой точке параметр `rop` содержит объект позиции приходного акта, для которой выполняется обработка.

В примере вызывается метод `Stmprs_ActInPkg.setObjNeedDetQtyBase(...)`, который связывает позицию акта с указанной потребностью и передает количество в базовой единице измерения.

Место применения: `Обозреватель проектов > нужный API-класс > точка расширения API`

Внимание

В текущем виде скрипт содержит захардкоженный идентификатор потребности `idvNeedObj`. Перед использованием замените его на актуальное значение.

Тип: JEXL-скрипт

```
// Идентификатор потребности
const idvNeedObj = toLong(32353);
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
// Подбор потребности к позиции приходного акта
Stmprs_ActInPkg.setObjNeedDetQtyBase(rop.idDoc, rop.gid, rop.nQtyBase, idvNeedObj, rop.
↪idMsr, null);
```

Обновление выборки после выполнения операции

Выполняет обновление текущей выборки после выполнения операции.

Скрипт используется в точке расширения операции `Avi`. Может применяться, если после выполнения пользовательской операции нужно сразу обновить данные в списке без ручного нажатия **Обновить**.

Место применения: `Обозреватель проектов > нужный Avi > точка расширения операции`

Тип: JEXL-скрипт

```
// Обновление текущей выборки
selection.refresh();
```

Отключение пересчета суммы договора при создании спецификации

Снимает признак пересчета суммы договора по дочерним объектам при создании спецификации.

Скрипт используется в точке расширения обозревателя проектов. При создании спецификации он получает родительский договор и проверяет его состояние. Если договор находится в состоянии **Действующий**, скрипт отключает признак `bCalcSumByChild` на реквизитах договора.

Место применения: `Обозреватель проектов > нужный Avi > точка расширения операции`

Тип: JEXL-скрипт

```
var ropContract = Cnt_ContractApi.load(getVar("super$id").asNLong());

if (ropContract.idState == Cnt_ContractApi.idStateExecuting()) {
  // Отключение пересчета суммы договора по дочерним объектам
  Cnt_ContractApi.setbCalcSumByChild(ropContract, 0b);
}
```

7.16 Интеграции

Определение вида сценария JEXL

Определяет сценарий интеграции — значение XML-тега схемы — в зависимости от подкласса документа, например хозяйственной операции или бухгалтерской справки, и класса документа-источника.

Скрипт показывает пример определения сценария по нескольким условиям. Также может использоваться как пример структурирования JEXL-скрипта на логические блоки и использования функций.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт` или в теле другого JEXL-скрипта

Тип: JEXL-скрипт

```
//Определение вида сценария интеграции
// Входящие параметры: id объекта интеграции
// Блок переменных
var svDocSubClass = null; // Подкласс хозяйственной операции
var svDocObjType = null; // Тип объекта хозяйственной операции
var svSrcDocClass = null; // Класс документа-источника хозяйственной операции
var svSrcDocSubClass = null; // Подкласс документа-источника хозяйственной операции
var svSrcDocObjType = null; // Тип объекта документа-источника хозяйственной операции
var svScenario = null; // Сценарии бухгалтерской справки

// Блок функций
// Функция для получения кода сценария бухгалтерской справки
var procGetScenarioCode = function(rop) {
   svScenario =  nvl(Act_TransDocApi.getObjAttrValue(rop, 'sScenarioSng'), 'NOTDEFINED');
   if (svScenario != 'NOTDEFINED') {
       @begin{
          let selectScenarioCode = `select
                                    cast(scen.scode as varchar) as scode
                                    from zsngbs_expscenarioname scen
                                    where scen.scaption = '`+svScenario+`'`;
          svScenario = sql(selectScenarioCode).asSingle().scode;
       }
   }
   return svScenario;
}


// Функция получения подкласса документа
var procGetSubClass = function(rop) {
   let result =  null;
       @begin{
          result = Btk_ObjectTypeApi.load(rop.getByAttrName("idObjectType")).idSubClass;
       }
   return result;
}

// Блок создания объектов
var rop = Act_TransDocApi.load(id); // Хозяйственная операция
  svDocSubClass = Btk_SubClassApi.load(procGetSubClass(rop)).sCode;
  svDocObjType = Btk_ObjectTypeApi.load(rop.idObjectType).sCode;

if (isNotNull(rop.gidSrc)) {
  var ropDoc = Btk_ClassApi.loadObject(rop.gidSrc); // Документ-источник
  svSrcDocClass = Btk_ClassApi.load(parseIdClass(rop.gidSrc)).sName;
  svSrcDocSubClass = Btk_SubClassApi.load(procGetSubClass(rop)).sCode;
  svSrcDocObjType = Btk_ObjectTypeApi.load(ropDoc.idObjectType).sCode;
}

// Блок определения вида сценария
if (svDocSubClass == 'Act_TransDoc_BusinessOperation') {
   if(svSrcDocClass = 'Pm_AdvRep') {
      return 'ACC_AO';
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
   }
}
else if (svDocSubClass == 'Act_TransDoc_AccountingInfo') {
  return procGetScenarioCode(rop);
}
else return 'NOTDEFINED';
```

7.17 Производительность и тестирование

Нагрузочный тест сервера приложений

Запускает встроенный тест производительности и создает нагрузку на сервер приложений.

Скрипт генерирует нагрузку в основном по CPU и операциям в памяти. Может использоваться для проверки поведения системы под нагрузкой и базового тестирования производительности.

При 100 итерациях выполняется примерно 1 минуту.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
var i = 1;

while (i <= 100) {
    // запуск одного тестового прогона
    Btk_PerformanceTestApi.runTest();

    // переход к следующей итерации
    i = i + 1;
}
```

7.18 Служебные примеры JEXL

Добавление MSCH Status

Служебный пример JEXL для демонстрации технического приема, преобразования данных или вызова API.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
Mct_StatusPosApi.register(
      npCode = 670B,
      npPriority = 670B,
      spCaption = "МСЧ не расцехован",
      spDescription = null,
      spColor = "Mct_IncorrectPosition",
      bpIsIncorrect = 1B
    );
commit();
```

Bts procedure

Служебный пример JEXL для демонстрации технического приема, преобразования данных или вызова API.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
var idvClass = gidpSrc.parseIdClass();
var idvClassStruct = Mct_StructureApi.idClass();
if(idvClass == idvClassStruct){
        var idvGds = Mct_StructureApi.load(gidpSrc.parseIdNLong()).copyAro().idGds();
        var rvGds = Bs_GoodsApi.load(idvGds).copyAro();
        rvGds.nWeight();
} else {
        null;
}

sql(`
        select
                max(g.jTypeSizeattrs ->> 'diameter' ) as "nWeight"
        from mct_structure s
        join bs_goods g on s.idGds = g.id
        where s.gid = '` + gidpSrc + "'"
        ).asSingle().nWeight;

var res = Bts_ProcedureApi.execWithGidSrc(6L, "43501/6238641");
//dialogs.showMessage(res.toString());
raise(res);
//println(res);
```

Pg Array To N Long List

Служебный пример JEXL для демонстрации технического приема, преобразования данных или вызова API.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
  var arr = sql("select array_agg(d.id) as ida from (select t.id from btk_class t limit␣
↪10) d").asSingle().ida;
  var list = pgArrayToNLongList(arr);
  var emptyList = pgArrayToNLongList(null);
```

Test SQL

Служебный пример JEXL для демонстрации технического приема, преобразования данных или вызова API.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
sql(`
with migratePosType as (select
        cast('ТК' as varchar) as sCodeOld
        ,cast('ТКЧ' as varchar) as sCodeNew
        ,cast('Техкомплекты по чертежу' as varchar) as sCaption
)
select t.sCodeOld as "sCodeOld"        ,t.sCodeNew as "sCodeNew"        ,t.sCaption as
↪"sCaption"        ,t1.id as idOld        ,t2.id as idNew
from migratePosType t
        left join Mct_PosType t1 on t.sCodeOld = t1.scode
        left join Mct_PosType t2 on t.sCodeNew = t2.scode
 `).foreach(function(r){
@begin{
        println(r.sCodeOld);
        println(r.sCodeNew);
        println(r.sCaption);
  }
@exception
function(exp){
        println("[JEXL Exception] " + exp.getCause());
}end;
}) ;
```

Обновление Def Det Method

Служебный пример JEXL для демонстрации технического приема, преобразования данных или вызова API.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
var id = sql(`
select
        dm.id as "id"
from mct_determinationmethod dm
where dm.scode = 'UtilizationCoef'
 `).asSingle() ;
if (id != null) {
        Mct_DeterminationMethodApi.setbIsDefault(Mct_DeterminationMethodApi.load(id),␣
↪1B);
}
commit();
```

7.19 Локальные скрипты модулей

WMS

Выполнение отгрузки без мастер-документа

Позволяет выполнить отгрузку `Wms_Shipping` без изменения состояния мастер-документа.

Скрипт используется в ситуации, когда из-за сбоя в модуле WMS состояние мастер-документа уже стало **Выполнен**, а состояние заявки на отгрузку `Wms_ShipOrd` и самой отгрузки `Wms_Shipping` осталось **Исполняется**. В таком случае обычное переключение состояния может быть недоступно.

Скрипт временно отвязывает заявку на отгрузку от мастер-документа, выполняет отгрузку и заявку на отгрузку, а затем возвращает связь с мастер-документом. За счет этого удается завершить отгрузку без попытки повторно изменить состояние мастер-документа.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
// id отгрузки, которую необходимо выполнить
var shipIdToBeDone = {вставить id отгрузки, которую нужно выполнить}L;

// загрузка отгрузки и связанной заявки на отгрузку
var ropShip = Wms_ShippingApi.load(shipIdToBeDone);
var ropOrd = Wms_ShipOrdApi.loadByGid(ropShip.gidSource);

// сохранение ссылки на мастер-документ
var src = ropOrd.gidSource;
var ropInternalWarrant = Stk_InternalWarrantApi.loadByGid(src);

// временное отключение связи с мастер-документом
Wms_ShipOrdApi.setgidSource(ropOrd, null);

// перевод отгрузки в выполненное состояние
Wms_ShippingApi.setidState(ropShip, Wms_ShippingApi.idgStateDone());

// восстановление связи с мастер-документом
Wms_ShipOrdApi.setgidSource(ropOrd, src);

// перевод мастер-документа в выполненное состояние
Stk_InternalWarrantApi.setidStateOut(ropInternalWarrant, Stk_InternalWarrantApi.
↪idStDone());
Stk_InternalWarrantApi.setidStateIn(ropInternalWarrant, Stk_InternalWarrantApi.
↪idStDone());
```

Высвобождение остаточного резервирования

Снимает резерв типа `Res` в регистре оборотов `Wms_RegTurn` для выполненных позиций отгрузки.

Скрипт используется в ситуации, когда из-за сбоя в модуле WMS резерв типа `Res` в регистре оборотов `Wms_RegTurn` не списался после выполнения позиции отгрузки. Скрипт находит такие позиции и выполняет корректирующее списание резерва.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
// поиск выполненных позиций отгрузки, по которым остался ненулевой резерв
for (s : sql(`select rt.gidsrcobj
   from wms_regturn rt
   join wms_shipdet s
     on s.gid = rt.gidsrcobj
  where stype = 'Res'
    and s.idoperstate = (select id from wms_operstate where norder = 300)
  group by gidsrcobj
  having sum(rt.nqtysm1) <> 0`).asList()) {

  // загрузка позиции отгрузки
  var rop = Wms_ShipDetApi.loadByGid(s.gidsrcobj);

  // повторное проведение движения между состояниями для списания остаточного резерва
  Wms_ShipDetApi.doOutLot(rop, Wms_OperStateApi.idgDone(), Wms_OperStateApi.idgReg());
  Wms_ShipDetApi.doOutLot(rop, Wms_OperStateApi.idgReg(), Wms_OperStateApi.idgDone());
}
```

Открытие карточки миксина документа Wms_Document

Открывает документ в карточке миксина документов `Wms_Document`.

Скрипт используется как служебный пример, если нужно посмотреть, как выглядит карточка миксина документа `Wms_Document`.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Тип: JEXL-скрипт

```
// открытие карточки миксина документа Wms_Document
Wms_DocumentAvi
  .card()
  .newForm()
  .params({
    "IdItem#": "167200/6802",
    "EDITINGTYPE": "edit"
  })
  .open();
```

MCT

Добавление Doc Tech Route Tab

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
Btk_TabApi.register(
        "Mct_StructureAvi.Tree_DocTechRoute",
        "Тех. маршрут",
        "gtk-Mct_StructureAvi",
        "Tree_DocTechRoute",
        Mct_SpecificationApi.idClass(),
        null,
        null,
        null,
        null,
        null,
        null
        );
commit();
```

Добавление Gds Info Tab OS

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvGdsInfoTab = Btk_TabApi.register(
        "Mct_OrderSheetVerDetAvi.Tab_GdsInfo",
        "Сопоставление со справочником ТМЦ/МСЧ",
        "gtk-Mct_OrderSheetVerDetAvi",
        "Tab_GdsInfo",
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        Mct_OrderSheetApi.idClass(),
        null,
        null,
        null,
        null,
        null,
        null
        );
var idvPosOs = Mct_PosTypeApi.idPosOS();
sql(`
        select ot.id as "idObjectType"
        from btk_ObjectType ot
        join btk_class c on ot.idrefclass = c.id
        where c.sname = 'Mct_OrderSheet'
`).foreach(function(r){
    var idvTypePosApplic = Mct_TypePosApplicApi.register(idvPosOs, r.idObjectType);
        Mct_TypePosApplicTabApi.register(idvTypePosApplic, idvGdsInfoTab, 50B);
});
commit();
```

Добавление Up Content Tab

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
Btk_TabApi.register(
        "Mct_TechProcAvi.List_UPContent",
        "Перечень МСЧ состава",
        "gtk-Mct_TechProcAvi",
        "List_UPContent",
        Mct_TechProcApi.idClass(),
        null,
        null,
        null,
        null,
        null,
        null
        );
commit();
```

Добавление WS Link Tab

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
Btk_Pkg.setRWSharedUOWEditType();
var idvWSOT = Mct_WorkStructureApi.idDefObjectType();
Btk_ObjectTypeTabApi.register(
        idvWSOT,
        "Mct_LinkAvi.List_gidWorkStructure",
        "Документы",
        "gtk-Mct_LinkAvi",
        "List_gidWorkStructure",
        1B,
        80B,
        18B,
        null,
        null,
        null,
        null,
        1B
        );
commit();
```

Добавление WS Link Tab1

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
Btk_Pkg.setRWSharedUOWEditType();
var idvWSOT = Mct_WorkStructureApi.idDefObjectType();
Btk_ObjectTypeTabApi.register(
        idvWSOT,
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        "Mct_WSLinkAvi.List_idWorkStructure",
        "Базовые ПУЕ",
        "gtk-Mct_WSLinkAvi",
        "List_idWorkStructure",
        1B,
        200B,
        18B,
        null,
        null,
        null,
        null,
        0B
        );
commit();
```

Очистка WS Labor

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvWS = null;
var idvActualLaborType = Mct_LaborTypeApi.idltActual();
var ropList =  sql(`
        select l.id
        from Mct_WorkStructure ws
        join mct_wrkstructtree wt on wt.idparent = ws.id
        join Mct_WorkStructure wsc on wt.idchild = wsc.id
        join Mct_Labor l on l.idworkstructure = wsc.id
        where wt.idparent != wt.idchild
        and ws.id = ` + idvWS + `
        and l.idlabortype = ` + idvActualLaborType
).batchObjLoad(Mct_LaborApi, "id");
for (rop : ropList){
        Mct_LaborApi.delete(rop);
}
var ropWS = Mct_WorkStructureApi.load(idvWS);
Mct_WorkStructureApi.setnActualManPower(ropWS, null);
commit();
```

Удаление OS Error Journal

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvOSVerClass = Mct_OrderSheetVerApi.idClass();
var ropList =  sql(`
                select
                        j.id
                from mct_journal j
                join btk_objecttype ot on j.idobjecttype = ot.id
                left join mct_ordersheetver osv on j.gidsrcver = osv.gid
                where coalesce(cast(j.jobjattrs_dz ->> 'bProcessed' as numeric), 0) = 0
                and ot.scode in ('InitMatList', 'WorkSpSheet', 'PkMatSheet')
                and getgidclass(j.gidsrcver) = ` + idvOSVerClass + `
                and osv.id is null
                        `).batchObjLoad(Mct_JournalApi, "id");
        for (rop : ropList){
                Mct_JournalApi.delete(rop);
        }
        commit();
```

Удаление TPLN Class Collection

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
Btk_Pkg.setRWSharedUOWEditType();
sql(`
        select
                cc.id
        from btk_classcollection cc
        join btk_class mc on cc.idbtkclass = mc.id
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        join btk_class rc on cc.idrefclass = rc.id
        join btk_attribute ra on cc.idrefattr = ra.id
        where mc.sname = 'Mct_TechProcListNorm'
        and rc.sname = 'Mct_TechProcNorm'
        and ra.ssystemname = 'gidSrc'
`).foreach(function(r){
        var rop = Btk_ClassCollectionApi.load(r.id);
        Btk_ClassCollectionApi.delete(rop);
});
commit();
```

Удаление WS Error Journal

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvWSClass = Mct_WorkStructureApi.idClass();
var ropList =  sql(`
                select
                        j.id
                from mct_journal j
                join btk_objecttype ot on j.idobjecttype = ot.id
                left join Mct_WorkStructure ws on j.gidsrc = ws.gid
                where coalesce(cast(j.jobjattrs_dz ->> 'bProcessed' as numeric), 0) = 0
                and ot.scode in ('InitMatList', 'WorkSpSheet', 'PkMatSheet')
                and getgidclass(j.gidsrc) = ` + idvWSClass + `
                and ws.id is null
                        `).batchObjLoad(Mct_JournalApi, "id");
        for (rop : ropList){
                Mct_JournalApi.delete(rop);
        }
        commit();
```

Mark Deleted UP Structure

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var nvCount = sql(`
        select count(s.id) as "nCount"
        from mct_structure s
        join bs_prjver pv on s.idprjver = pv.id
        where pv.scode = 'МСЧ'
        and coalesce(s.bdeleted, 0) = 0
        `).asSingle().nCount;
var nvIter = nvCount / 5000 + 1;
for (i : (1 .. nvIter)){
        var ropList =  sql(`
                select s.id
                from mct_structure s
                join bs_prjver pv on s.idprjver = pv.id
                where pv.scode = 'МСЧ'
                and coalesce(s.bdeleted, 0) = 0
                limit 5000
                `).batchObjLoad(Mct_StructureApi, "id");
        for (rop : ropList){
                Mct_StructureApi.setbDeleted(rop, 1B);
                Mct_StructureApi.setsCode(rop, "[A]" + rop.getByAttrName("sCode") + "_"␣
↪+ rop.idJ().toString());
        }
        commit();
}
```

Миграция Msch Prj Ver

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvPrj = Bs_PrjApi.findByMnemoCode("МСЧ");
var idvPrjVer = Bs_PrjVerApi.findByMnemoCode(Bs_PrjApi.load(idvPrj), "МСЧ");
sql(`
with mschOT as (select ot.id as idMctObjectType
        from btk_objecttype ot
        where ot.scode = 'UnitProduct')
select s.id as "idSpecification" from mct_specification s
where s.idobjecttype in (select idMctObjectType from mschOT)
 `).foreach(function(r){
@begin{
        println(r.idSpecification);
        var rop = Mct_SpecificationApi.load(r.idSpecification);
    Mct_SpecificationApi.setidPrjVer(rop, idvPrjVer);
  }
@exception
function(exp){
        println("[JEXL Exception] " + exp.getCause());
}end;
}) ;
commit();
```

Миграция OT Zip OS

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
Btk_Pkg.setRWSharedUOWEditType();
var idvClass = Mct_OrderSheetApi.idClass();
var idvClassVer = Mct_OrderSheetVerApi.idClass();
var idvOSZipWrk = Btk_ObjectTypeApi.register(
      spCode = "OrderSheetZip",
      spCaption = "Заказная ведомость - ЗИП",
      spShortCaption = "ЗВ - ЗИП",
      idpRefClass = idvClass,
      spDescription = null,
      bpIsDefault = 0B,
      bpIsSystem = 0B,
      spDefCardSel = null,
      spDefCardRep = null,
      idpLifeCycle = null,
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
      bpHasVersion = 1B,
      idpVersionClass = idvClassVer,
      idpSubClass = null,
      bpRewrite = 0B
    );
    Btk_ObjectTypeTabApi.register(idvOSZipWrk, "verdet", "Позиции", "Mct_
↪OrderSheetVerDetAvi", "List_gidVerEquip", 1B, 10B, 22B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvOSZipWrk, "ver", "Версии", "Mct_DocumentVersionAvi",
↪ "List_Master", 1B, 20B, 33B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvOSZipWrk, "NSDistr", "Распределение по этапам␣
↪потребности", "MctPrs_OrderSheetVerDetAvi", "list_gidVerNS", 1B, 25B, 58B, null, null,␣
↪null, null);
    Btk_ObjectTypeTabApi.register(idvOSZipWrk, "NeedStage", "Этапы потребности", "MctPrs_
↪OSNeedStageAvi", "list_idOrderSheet", 1B, 26B, 58B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvOSZipWrk, "note", "Примечание", "Btk_DocNoteAvi",
↪"Html_Frame", 1B, 30B, 43B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvOSZipWrk, "link", "Связанные объекты", "Mct_LinkAvi
↪", "List_gidObject", 1B, 50B, 30B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvOSZipWrk, "contentAnalysis", "Анализ состава ВЗ",
↪"Mct_OrderSheetVerDetAvi", "List_ContentAnalysis", 1B, 50B, 1B, null, null, null,␣
↪null);
    Btk_ObjectTypeTabApi.register(idvOSZipWrk, "compare", "Сравнение версий", "Mct_
↪SpecificationVerPosAvi", "CompareVersions", 1B, 60B, 30B, null, null, null, null);
    //регистрируем настройку типа объекта Mct
    Mct_ObjectTypeApi.register(idpObjectType = idvOSZipWrk,
      spDefCaption = "Заказная ведомость - ЗИП",
      npOrder = null,
      bpCreateFromRkd = 0B,
          bpAcceptWithoutGen = 0B,
      bpIsPrjDoc = 0B,
      idpWrkDoc = null,
          bpUseInFilterCs = 0B);
var idvOSZipPrj = Btk_ObjectTypeApi.register(
      spCode = "OrderSheetZipPrj",
      spCaption = "Заказная ведомость - ЗИП (проектная)",
      spShortCaption = "ЗВП - ЗИП",
      idpRefClass = idvClass,
      spDescription = "Проектный документ",
      bpIsDefault = 0B,
      bpIsSystem = 0B,
      spDefCardSel = null,
      spDefCardRep = null,
      idpLifeCycle = null,
      bpHasVersion = 1B,
      idpVersionClass = idvClassVer,
      idpSubClass = null,
      bpRewrite = 0B
    );
    Btk_ObjectTypeTabApi.register(idvOSZipPrj, "verdet", "Позиции", "Mct_
↪OrderSheetVerDetAvi", "List_gidVerEquipPrj", 1B, 10B, 22B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvOSZipPrj, "ver", "Версии", "Mct_DocumentVersionAvi",
↪ "List_Master", 1B, 20B, 33B, null, null, null, null);
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    Btk_ObjectTypeTabApi.register(idvOSZipPrj, "note", "Примечание", "Btk_DocNoteAvi",
↪"Html_Frame", 1B, 30B, 43B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvOSZipPrj, "prjver", "Применяемость документа к␣
↪заказам", "Mct_DocPrjVerAvi", "List_gidDoc", 1B, 40B, 40B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvOSZipPrj, "link", "Связанные объекты", "Mct_LinkAvi
↪", "List_gidObject", 1B, 50B, 30B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvOSZipPrj, "compare", "Сравнение версий", "Mct_
↪SpecificationVerPosAvi", "CompareVersions", 1B, 60B, 30B, null, null, null, null);
    //регистрируем настройку типа объекта Mct
    Mct_ObjectTypeApi.register(idpObjectType = idvOSZipPrj,
      spDefCaption = "Заказная ведомость - ЗИП (проектная)",
      npOrder = null,
      bpCreateFromRkd = 0B,
          bpAcceptWithoutGen = 0B,
      bpIsPrjDoc = 1B,
      idpWrkDoc = idvOSZipWrk,
          bpUseInFilterCs = 0B);
commit();
```

Миграция Object Type Proc Oper

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
sql(`
select
                mot.id as "idObjectType"
                ,ot.sCaption as "sCaption"
        from btk_ObjectType ot
        left join mct_objectType mot on ot.id = mot.idbtkobjecttype
        where ot.idrefclass = (select id from btk_class where sname = 'Mct_Specification
↪')
 `).foreach(function(r){
@begin{
        println(r.sCaption);
        Mct_ObjectTypeProcOperApi.register(idpMctObjectType = r.idObjectType,
      spSystemName = "recalcNormPerUnit",
      spCaption = "Пересчитать нормы расхода на ед.",
      npImage = 122B,
      npOrder = 90B,
      spApplicationMethod = "Mct_SpecificationVerPosApi().recalcNormPerUnit()",
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
          bpIsActive = 1B);
  }
@exception
function(exp){
        println("[JEXL Exception] " + exp.getCause());
}end;
}) ;
commit();
sql(`
        select id as "idProcOper"
        from Mct_ObjectTypeProcOper
        where ssystemname = 'validateAllPos'
 `).foreach(function(r){
@begin{
        var ropProcOper = Mct_ObjectTypeProcOperApi.load(r.idProcOper);
        Mct_ObjectTypeProcOperApi.setnOrder(ropProcOper, 999B);
  }
@exception
function(exp){
        println("[JEXL Exception] " + exp.getCause());
}end;
}) ;
commit();
```

Миграция Pos Type

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var list = [["Pk", "ТК", "Техкомплект"], ["ТК", "ГТЧ", "Группа техкомплектов по чертежу
↪"], [null, "ПК", "Подкомплект"], [null, "ГТК", "Группа техкомплектов"]];
for (posType : list){
        var idvPosTypeOld = Mct_PosTypeApi.findByMnemoCode(posType.get(0));
        if (idvPosTypeOld != null) {
                //если нашли старый тип позиции - меняем код и наименование на новые
                println(posType.get(0) + " -> " + idvPosTypeOld);
                var ropPosType = Mct_PosTypeApi.load(idvPosTypeOld);
                Mct_PosTypeApi.setsCode(ropPosType, posType.get(1));
                Mct_PosTypeApi.setsCaption(ropPosType, posType.get(2));
        }
        else {
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                //ищем новый тип позиции
                var idvPosTypeNew = Mct_PosTypeApi.findByMnemoCode(posType.get(1));
                if (idvPosTypeNew == null) {
                        //если не нашли - создаем
                        println("Создаем новый тип позиции " + posType.get(2));
                        Mct_PosTypeApi.register(posType.get(1), posType.get(2), null);
                }
        }
}
commit();
```

Миграция Pos Type Gds Contras Set Type

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var listNone = pgArrayToNLongList(null);
var ropListAll =  sql(`
        select id
        from mct_docposset ds
        where coalesce(ds.bSetContrasCode, 0) = 0
        `).batchObjLoad(Mct_DocPosSetApi, "id");
for (rop : ropListAll){
        Mct_DocPosSetApi.setGdsArticleSetType(rop, false, listNone);
}
var ropListAll =  sql(`
        select id
        from mct_docposset ds
        where coalesce(ds.bSetContrasDesignation, 0) = 0
        `).batchObjLoad(Mct_DocPosSetApi, "id");
for (rop : ropListAll){
        Mct_DocPosSetApi.setGdsDesignationSetType(rop, false, listNone);
}
var ropListAll =  sql(`
        select id
        from mct_docposset ds
        where coalesce(ds.bSetContrasCaption, 0) = 0
        `).batchObjLoad(Mct_DocPosSetApi, "id");
for (rop : ropListAll){
        Mct_DocPosSetApi.setGdsCaptionSetType(rop, false, listNone);
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
}
commit();
```

Миграция Pos Type Gds Match Type

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var listNone = pgArrayToNLongList(null);
var ropListAll =  sql(`
        select id
        from mct_docposset ds
        where ds.bCanSetGoods = 1
        and ds.bCanSetUnitProduct = 1
        `).batchObjLoad(Mct_DocPosSetApi, "id");
for (rop : ropListAll){
        Mct_DocPosSetApi.setGdsMatchTypes(rop, true, listNone);
}
var arrGds = sql(`select array_agg(gg.id) as ida from gds_goodsandservicetype gg
        where smnemocode != 'MSCH'`).asSingle().ida;
var listGds = pgArrayToNLongList(arrGds);
var ropListGds =  sql(`
        select id
        from mct_docposset ds
        where ds.bCanSetGoods = 1
        and coalesce(ds.bCanSetUnitProduct, 0) = 0
        `).batchObjLoad(Mct_DocPosSetApi, "id");
for (rop : ropListGds){
        Mct_DocPosSetApi.setGdsMatchTypes(rop, false, listGds);
}
var arrUP = sql(`select array_agg(gg.id) as ida from gds_goodsandservicetype gg
        where smnemocode = 'MSCH'`).asSingle().ida;
var listUP = pgArrayToNLongList(arrUP);
var ropListUP =  sql(`
        select id
        from mct_docposset ds
        where coalesce(ds.bCanSetGoods, 0) = 0
        and ds.bCanSetUnitProduct = 1
        `).batchObjLoad(Mct_DocPosSetApi, "id");
for (rop : ropListUP){
        Mct_DocPosSetApi.setGdsMatchTypes(rop, false, listUP);
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
}
var ropListNone =  sql(`
        select id
        from mct_docposset ds
        where coalesce(ds.bCanSetGoods, 0) = 0
        and coalesce(ds.bCanSetUnitProduct, 0) = 0
        `).batchObjLoad(Mct_DocPosSetApi, "id");
for (rop : ropListNone){
        Mct_DocPosSetApi.setGdsMatchTypes(rop, false, listNone);
}
commit();
```

Миграция Pos Type Izd

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvIzd = Mct_PosTypeApi.register("СБ", "Сборка", null, ОВ);
//регистрируем доступных потомков
Mct_PosChildAvailApi.register(idvIzd, Mct_PosTypeApi.findByMnemoCode("Д"), null);
Mct_PosChildAvailApi.register(idvIzd, Mct_PosTypeApi.findByMnemoCode("ТМЦ"), null);
Mct_PosChildAvailApi.register(idvIzd, Mct_PosTypeApi.findByMnemoCode("SpecPosGroup"),␣
↪null);
Mct_PosChildAvailApi.register(idvIzd, Mct_PosTypeApi.findByMnemoCode("У"), null);
Mct_PosChildAvailApi.register(idvIzd, Mct_PosTypeApi.findByMnemoCode("МСЧ"), Mct_
↪SpecificationVerPosApi.idClass());
Mct_PosChildAvailApi.register(idvIzd, Mct_PosTypeApi.findByMnemoCode("В"), null);
commit();
//регистрируем потомка "Сборка" для корня СП
var idvSpecPosRoot = Mct_PosTypeApi.findByMnemoCode("SpecPosRoot");
Mct_PosChildAvailApi.register(idvSpecPosRoot, idvIzd, null);
commit();
```

Миграция Pos Type Msch Load

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
Btk_Pkg.setRWSharedUOWEditType();
var idvObjectType = Btk_ObjectTypeApi.findByMnemoCode("workstructure");
var ropOT = Btk_ObjectTypeApi.load(idvObjectType);
var idvOTDet = Btk_ObjectTypeTabApi.register(idvObjectType, "Mct_WsProductManuf",
↪"Изготовляемые МСЧ", "gtk-Mct_WsProductManufAvi", "List_idWorkStructure", 1B, 90B, 18B,
↪ null, null, null, null);
//ТКМСЧ
var idvTkMsch = Mct_PosTypeApi.register("ТКМСЧ", "Техкомплект МСЧ", null, 0B);
//регистрируем доступных потомков
Mct_PosChildAvailApi.register(idvTkMsch, Mct_PosTypeApi.findByMnemoCode("ПК"), null);
//регистрация закладок
var idvDetTK = Mct_TypePosApplicApi.register(idvTkMsch, idvObjectType);
Mct_TypePosApplicTabApi.register(idvDetTK, Btk_ObjectTypeTabApi.findByMnemoCode(ropOT,
↪"card"));
Mct_TypePosApplicTabApi.register(idvDetTK, Btk_ObjectTypeTabApi.findByMnemoCode(ropOT,
↪"Mct_Link"));
Mct_TypePosApplicTabApi.register(idvDetTK, Btk_ObjectTypeTabApi.findByMnemoCode(ropOT,
↪"Mct_TechReference"));
Mct_TypePosApplicTabApi.register(idvDetTK, Btk_ObjectTypeTabApi.findByMnemoCode(ropOT,
↪"Mct_WorkSructureRoom"));
Mct_TypePosApplicTabApi.register(idvDetTK, Btk_ObjectTypeTabApi.findByMnemoCode(ropOT,
↪"Mct_Labor"));
Mct_TypePosApplicTabApi.register(idvDetTK, Btk_ObjectTypeTabApi.findByMnemoCode(ropOT,
↪"Mct_WorkStructureGdsList"));
Mct_TypePosApplicTabApi.register(idvDetTK, Btk_ObjectTypeTabApi.findByMnemoCode(ropOT,
↪"ActionRef"));
Mct_TypePosApplicTabApi.register(idvDetTK, Btk_ObjectTypeTabApi.findByMnemoCode(ropOT,
↪"note"));
commit();
//Запуск МСЧ
var idvLoadMsch = Mct_PosTypeApi.register("ЗМСЧ", "Запуск МСЧ", null, 0B);
//регистрируем доступных потомков
Mct_PosChildAvailApi.register(idvLoadMsch, Mct_PosTypeApi.findByMnemoCode("ТКМСЧ"),␣
↪null);
Mct_PosChildAvailApi.register(idvLoadMsch, Mct_PosTypeApi.findByMnemoCode("ГТК"), null);
//регистрация закладок
var idvDetLoad = Mct_TypePosApplicApi.register(idvLoadMsch, idvObjectType);
Mct_TypePosApplicTabApi.register(idvDetLoad, Btk_ObjectTypeTabApi.findByMnemoCode(ropOT,
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
↪"card"));
Mct_TypePosApplicTabApi.register(idvDetLoad, idvOTDet);
Mct_TypePosApplicTabApi.register(idvDetLoad, Btk_ObjectTypeTabApi.findByMnemoCode(ropOT,
↪"ActionRef"));
Mct_TypePosApplicTabApi.register(idvDetLoad, Btk_ObjectTypeTabApi.findByMnemoCode(ropOT,
↪"Mes_WorkOrder"));
Mct_TypePosApplicTabApi.register(idvDetLoad, Btk_ObjectTypeTabApi.findByMnemoCode(ropOT,
↪"Mct_Labor"));
Mct_TypePosApplicTabApi.register(idvDetLoad, Btk_ObjectTypeTabApi.findByMnemoCode(ropOT,
↪"Mct_TechReference"));
Mct_TypePosApplicTabApi.register(idvDetLoad, Btk_ObjectTypeTabApi.findByMnemoCode(ropOT,
↪"Mct_WorkSructureRoom"));
Mct_TypePosApplicTabApi.register(idvDetLoad, Btk_ObjectTypeTabApi.findByMnemoCode(ropOT,
↪"Mct_Link"));
Mct_TypePosApplicTabApi.register(idvDetLoad, Btk_ObjectTypeTabApi.findByMnemoCode(ropOT,
↪"note"));
commit();
//регистрируем потомка "Запуск МСЧ" для Технологический этап
var idvBigAction = Mct_PosTypeApi.findByMnemoCode("BigAction");
Mct_PosChildAvailApi.register(idvBigAction, idvLoadMsch, null);
commit();
//регистрируем потомка "Запуск МСЧ" для ПУГ
var idvPug = Mct_PosTypeApi.findByMnemoCode("Pug");
Mct_PosChildAvailApi.register(idvPug, idvLoadMsch, null);
commit();
```

Миграция типов позиций по таблице соответствия

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var list = [["Рк", "ТК", "Техкомплект"], ["ТК", "УР", "Укрупненная работа"], ["ПТК", "ПК
↪", "Технологический подкомплект"], [null, "ГТК", "Группа техкомплектов"], [null, "ГТЧ",
↪ "Группа техкомплектов по чертежу"]];
for (posType : list){
        var idvPosTypeOld = Mct_PosTypeApi.findByMnemoCode(posType.get(0));
        if (idvPosTypeOld != null) {
                //если нашли старый тип позиции - меняем код на новый
                println(posType.get(0) + " -> " + idvPosTypeOld);
                var ropPosType = Mct_PosTypeApi.load(idvPosTypeOld);
                Mct_PosTypeApi.setsCode(ropPosType, posType.get(1));
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        }
        else {
                //ищем новый тип позиции
                var idvPosTypeNew = Mct_PosTypeApi.findByMnemoCode(posType.get(1));
                if (idvPosTypeNew == null) {
                        //если не нашли - создаем
                        println("Создаем новый тип позиции " + posType.get(2));
                        Mct_PosTypeApi.register(posType.get(1), posType.get(2), null);
                }
        }
}
commit();
```

Миграция Pre Order Sheet

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
Btk_Pkg.setRWSharedUOWEditType();
var idvClass = Mct_OrderSheetApi.idClass();
var idvClassVer = Mct_OrderSheetVerApi.idClass();
var idvPreOrderSheet = Btk_ObjectTypeApi.register(
      spCode = "PreOrderSheet",
      spCaption = "Предварительная заказная ведомость - материалы",
      spShortCaption = "ПЗВ - Материалы",
      idpRefClass = idvClass,
      spDescription = null,
      bpIsDefault = 0B,
      bpIsSystem = 0B,
      spDefCardSel = null,
      spDefCardRep = null,
      idpLifeCycle = null,
      bpHasVersion = 1B,
      idpVersionClass = idvClassVer,
      idpSubClass = null,
      bpRewrite = 0B
    );
    Btk_ObjectTypeTabApi.register(idvPreOrderSheet, "verdet", "Позиции", "gtk-Mct_
↪OrderSheetVerDetAvi", "List_gidVer", 1B, 10B, 22B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvPreOrderSheet, "ver", "Версии", "gtk-Mct_
↪DocumentVersionAvi", "List_Master", 1B, 20B, 33B, null, null, null, null);
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    Btk_ObjectTypeTabApi.register(idvPreOrderSheet, "note", "Примечание", "gtk-Btk_
↪DocNoteAvi", "Html_Frame", 1B, 30B, 43B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvPreOrderSheet, "link", "Связанные объекты", "gtk-
↪Mct_LinkAvi", "List_gidObject", 1B, 50B, 30B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvPreOrderSheet, "compare", "Сравнение версий", "gtk-
↪Mct_SpecificationVerPosAvi", "CompareVersions", 1B, 60B, 30B, null, null, null, null);
    //регистрируем настройку типа объекта Mct
    Mct_ObjectTypeApi.register(idpObjectType = idvPreOrderSheet,
      spDefCaption = "Предварительная заказная ведомость - материалы",
      npOrder = null,
      bpCreateFromRkd = 0B,
          bpAcceptWithoutGen = 0B,
      bpIsPrjDoc = 0B,
      idpWrkDoc = null,
          bpUseInFilterCs = 0B);
var idvPreOrderSheetEquip = Btk_ObjectTypeApi.register(
      spCode = "PreOrderSheetEquip",
      spCaption = "Предварительная заказная ведомость - оборудование",
      spShortCaption = "ПЗВ - Оборудование",
      idpRefClass = idvClass,
      spDescription = null,
      bpIsDefault = 0B,
      bpIsSystem = 0B,
      spDefCardSel = null,
      spDefCardRep = null,
      idpLifeCycle = null,
      bpHasVersion = 1B,
      idpVersionClass = idvClassVer,
      idpSubClass = null,
      bpRewrite = 0B
    );
    Btk_ObjectTypeTabApi.register(idvPreOrderSheetEquip, "verdet", "Позиции", "gtk-Mct_
↪OrderSheetVerDetAvi", "List_gidVerEquip", 1B, 10B, 22B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvPreOrderSheetEquip, "ver", "Версии", "gtk-Mct_
↪DocumentVersionAvi", "List_Master", 1B, 20B, 33B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvPreOrderSheetEquip, "note", "Примечание", "gtk-Btk_
↪DocNoteAvi", "Html_Frame", 1B, 30B, 43B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvPreOrderSheetEquip, "link", "Связанные объекты",
↪"gtk-Mct_LinkAvi", "List_gidObject", 1B, 50B, 30B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvPreOrderSheetEquip, "compare", "Сравнение версий",
↪"gtk-Mct_SpecificationVerPosAvi", "CompareVersions", 1B, 60B, 30B, null, null, null,␣
↪null);
    //регистрируем настройку типа объекта Mct
    Mct_ObjectTypeApi.register(idpObjectType = idvPreOrderSheetEquip,
      spDefCaption = "Предварительная заказная ведомость - оборудование",
      npOrder = null,
      bpCreateFromRkd = 0B,
          bpAcceptWithoutGen = 0B,
      bpIsPrjDoc = 0B,
      idpWrkDoc = null,
          bpUseInFilterCs = 0B);
commit();
```

Миграция Prj Doc Gds Compare Tab

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var ropList = sql(`
        select ott.id, ot.scode
        from btk_objecttypetab ott
        left join btk_objecttype ot on ott.idobjecttype = ot.id
        left join btk_class c on ot.idrefclass = c.id
        left join mct_objecttype mot on ot.id = mot.idbtkobjecttype
        where c.sname = 'Mct_Specification'
        and ott.ssystemname = 'gdssummary'
        and mot.bisprjdoc = 1
        `).batchObjLoad(Btk_ObjectTypeTabApi, "id");
Btk_Pkg.setRWSharedUOWEditType();
for (rop : ropList){
                    Btk_ObjectTypeTabApi.setsRepresentationName(rop, "Report_
↪gidVersionPrj");
        }
commit();
```

Миграция Prj Ver Det Prj Doc

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var ropList = sql(`
        select ott.id
        from btk_objecttypetab ott
        left join btk_objecttype ot on ott.idobjecttype = ot.id
        left join btk_class c on ot.idrefclass = c.id
        where c.sname = 'Mct_Specification'
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        and ott.ssystemname = 'prjver'
        order by ott.id
        `).batchObjLoad(Btk_ObjectTypeTabApi, "id");
Btk_Pkg.setRWSharedUOWEditType();
for (rop : ropList){
                    Btk_ObjectTypeTabApi.setsSelectionName(rop, "gtk-Mct_
↪DocumentPrjVerAvi");
                    Btk_ObjectTypeTabApi.setsRepresentationName(rop, "List_gidDoc");
        }
commit();
```

Миграция Proc Oper Clear Spaces

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
sql(`
select
        mot.id as "idMctObjectTypeSpec"
from mct_objecttype mot
        left join btk_objecttype ot on mot.idbtkobjecttype = ot.id
        left join btk_class c on ot.idrefclass = c.id
where c.sname = 'Mct_Specification'
 `).foreach(function(r){
@begin{
        var idvMctOt = r.idMctObjectTypeSpec;
        Mct_ObjectTypeProcOperApi.register(idvMctOt, "clearSpaces", "Удалить лишние␣
↪пробелы", 14B, 5B, "Mct_SpecificationVerPosApi().clearSpaces()", 1B);
  }
@exception
function(exp){
        println("[JEXL Exception] " + exp.getCause());
}end;
}) ;
commit();
```

Миграция Status Pos

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
    Mct_StatusPosApi.register(
      npCode = 97B,
      npPriority = 97B,
      spCaption = "Не указана ЕИ нормирования",
      spDescription = null,
      spColor = "Mct_IncorrectPosition",
      bpIsIncorrect = 1B
    );
commit();
```

Миграция Tariff Scale Price

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var select = sql(`
        select distinct
                t.id
                ,t.nprice as "nPrice"
        from Mct_TariffScale t
        left join Mct_TariffScalePaymentRate tt on t.id = tt.idTariffScale
        where t.nprice is not null
        and tt.id is null
`);
//прогружаем тарифные сетки
select.batchObjLoad(Mct_TariffScaleApi, "id");
//создаем записи в коллекции с тарифными ставками
select.foreach(function(r){
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        var ropParent = Mct_TariffScaleApi.load(r.id);
        var ropRate = Mct_TariffScalePaymentRateApi.insertByParent(ropParent);
        Mct_TariffScalePaymentRateApi.setnPrice(ropRate, r.nPrice);
});
commit();
```

Миграция Tech Par Calc Method

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvObjectType = Mct_TechParameterApi.idProcObjectType();
sql(`
        select
                t.sCode as "sCode"
                ,t.scaption as "sCaption"
                ,t.sOperation as "sProcedure"
                ,t.sDescription as "sDescription"
        from Mct_TechParCalcMethod t
        left join bts_procedure p on t.scode = p.scode
        where p.id is null
        `).foreach(function(r){
      var rop = Bts_ProcedureApi.insert();
          Bts_ProcedureApi.setsCode(rop, r.sCode);
          Bts_ProcedureApi.setsCaption(rop, r.sCaption);
          Bts_ProcedureApi.setidObjectType(rop, idvObjectType);
          Bts_ProcedureApi.setsProcedure(rop, r.sProcedure);
          Bts_ProcedureApi.setsDescription(rop, r.sDescription);
    });
commit();
sql(`
        select
                t.id as "id"
                ,p.id as "idProcedure"
        from Mct_Techparameter t
        join mct_techparcalcmethod tpcm on t.idcalcmethod = tpcm.id
        join bts_procedure p on tpcm.sCode = p.sCode
        where t.idprocedure is null
        `).foreach(function(r){
      var rop = Mct_TechParameterApi.load(r.id);
          Mct_TechParameterApi.setidProcedure(rop, r.idProcedure);
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    });
commit();
```

Миграция Tech Param Src

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
//простановка класса источника ссылочных параметров
sql(`
        select
                tp.id
                ,case when a.ssystemname = 'id'
                        then c.id
                        else ac.id
                end as "idRefClass"
        from mct_techparameter tp
        join btk_class c on tp.idsrcclass = c.id
        join btk_attribute a on tp.idsrcattr = a.id
        left join btk_class ac on a.srefclass = ac.sname
        where tp.stype = 'Ref'
        and tp.idsrcattr is not null
        and tp.idrefclass is null
        `).foreach(function(r){
                var rop = Mct_TechParameterApi.load(r.id);
                Mct_TechParameterApi.dpi().setidRefClass(rop, r.idRefClass);
        });
commit();
//перенос в коллекцию источника значений
sql(`
        select
                tp.id
                ,tp.idsrcclass as "idSrcClass"
                ,tp.idsrcattr as "idSrcAttr"
        from mct_techparameter tp
        where tp.idsrcattr is not null
        and not exists (
                                        select 1 from mct_techparametersrc s
                                        where s.idtechparameter = tp.id
                                        and s.idsrcclass = tp.idsrcclass
                                        and s.idsrcattr = tp.idsrcattr
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                                        )
        `).foreach(function(r){
                Mct_TechParameterSrcApi.register(r.id, r.idSrcClass, r.idSrcAttr);
        });
commit();
```

Миграция Unit Prod Sheet

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
Btk_Pkg.setRWSharedUOWEditType();
var idvClass = Mct_UnitProdSheetApi.idClass();
var idvClassVer = Mct_UnitProdSheetVerApi.idClass();
var idvUnitProdSheet = Btk_ObjectTypeApi.register(
      spCode = "UnitProdSheet",
      spCaption = "Ведомость МСЧ рабочая",
      spShortCaption = "Ведомость МСЧ",
      idpRefClass = idvClass,
      spDescription = null,
      bpIsDefault = 0B,
      bpIsSystem = 1B,
      spDefCardSel = null,
      spDefCardRep = null,
      idpLifeCycle = null,
      bpHasVersion = 1B,
      idpVersionClass = idvClassVer,
      idpSubClass = null,
      bpRewrite = 1B
    );
    Btk_ObjectTypeTabApi.register(idvUnitProdSheet, "verdet", "Позиции", "Mct_
↪UnitProdSheetListAvi", "List_gidVer", 1B, 10B, 22B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvUnitProdSheet, "compareByPrj", "Сравенение данных␣
↪по проектной ведомости и СП", "Mct_UnitProdSheetListAvi", "List_CompareByPrj", 1B, 15B,
↪ 17B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvUnitProdSheet, "ver", "Версии", "Mct_
↪DocumentVersionAvi", "List_Master", 1B, 20B, 33B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvUnitProdSheet, "note", "Примечание", "Btk_DocNoteAvi
↪", "Html_Frame", 1B, 30B, 43B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvUnitProdSheet, "link", "Связанные объекты", "Mct_
↪LinkAvi", "List_gidObject", 0B, 50B, 30B, null, null, null, null);
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    Btk_ObjectTypeTabApi.register(idvUnitProdSheet, "compare", "Сравнение версий", "Mct_
↪UnitProdSheetListAvi", "CompareVersions", 1B, 60B, 30B, null, null, null, null);
    //регистрируем настройку типа объекта Mct
    Mct_ObjectTypeApi.register(idpObjectType = idvUnitProdSheet,
      spDefCaption = "Ведомость МСЧ рабочая",
      npOrder = null,
      bpCreateFromRkd = 0B,
          bpAcceptWithoutGen = 0B,
      bpIsPrjDoc = 0B,
      idpWrkDoc = null,
          bpUseInFilterCs = 0B);
var idvUnitProdSheetPrj = Btk_ObjectTypeApi.register(
      spCode = "UnitProdSheetPrj",
      spCaption = "Ведомость МСЧ проектная",
      spShortCaption = "Ведомость МСЧ-П",
      idpRefClass = idvClass,
      spDescription = null,
      bpIsDefault = 0B,
      bpIsSystem = 0B,
      spDefCardSel = null,
      spDefCardRep = "CardVerPrj",
      idpLifeCycle = null,
      bpHasVersion = 1B,
      idpVersionClass = idvClassVer,
      idpSubClass = null,
      bpRewrite = 1B
    );
    Btk_ObjectTypeTabApi.register(idvUnitProdSheetPrj, "verdet", "Позиции", "Mct_
↪UnitProdSheetListAvi", "List_gidVerPrj", 1B, 10B, 22B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvUnitProdSheetPrj, "ver", "Версии", "Mct_
↪DocumentVersionAvi", "List_Master", 1B, 20B, 33B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvUnitProdSheetPrj, "note", "Примечание", "Btk_
↪DocNoteAvi", "Html_Frame", 1B, 30B, 43B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvUnitProdSheetPrj, "link", "Связанные объекты", "Mct_
↪LinkAvi", "List_gidObject", 0B, 50B, 30B, null, null, null, null);
    Btk_ObjectTypeTabApi.register(idvUnitProdSheetPrj, "compare", "Сравнение версий",
↪"Mct_UnitProdSheetListAvi", "CompareVersions", 1B, 60B, 30B, null, null, null, null);
    //регистрируем настройку типа объекта Mct
    Mct_ObjectTypeApi.register(idpObjectType = idvUnitProdSheetPrj,
      spDefCaption = "Ведомость МСЧ проектная",
      npOrder = null,
      bpCreateFromRkd = 0B,
          bpAcceptWithoutGen = 0B,
      bpIsPrjDoc = 1B,
      idpWrkDoc = idvUnitProdSheet,
          bpUseInFilterCs = 0B);
commit();
```

Миграция WS Relation Tab

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
Btk_Pkg.setRWSharedUOWEditType();
var idvWorkStructureOT = Btk_ObjectTypeApi.findByMnemoCode("workstructure");
//закладка на тип объекта
var idvWorkRelationTab = Btk_ObjectTypeTabApi.register(idvWorkStructureOT, "WorkRelation
↪", "Связанные работы", "gtk-Mct_WorkRelationAvi", "List_RelationPrev", 1B, 100B, 43B,␣
↪null, null, null, null);
var idvTKPosType = Mct_PosTypeApi.idTK();
var idvDet = Mct_TypePosApplicApi.register(idvTKPosType, idvWorkStructureOT);
//отображение закладки для ТК в ПУЕ
Mct_TypePosApplicTabApi.register(idvDet, idvWorkRelationTab);
commit();
```

Миграция Wrk OS Content Analysis Tab

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
Btk_Pkg.setRWSharedUOWEditType();
sql(`
select
        ot.id as "idObjectType"
from Btk_objecttype ot
where ot.scode in ('OrderSheetEquip', 'OrderSheet')
 `).foreach(function(r){
@begin{
        var idvOt = r.idObjectType;
        Btk_ObjectTypeTabApi.register(idvOt, "contentAnalysis", "Анализ состава ВЗ",
↪"Mct_OrderSheetVerDetAvi", "List_ContentAnalysis", 1B, 50B, 1B, null, null, null,␣
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
↪null);
  }
@exception
function(exp){
        println("[JEXL Exception] " + exp.getCause());
}end;
}) ;
commit();
```

Восстановление Dim Val Flt

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
Mct_RefTableParamApi.synchParams(47301L);
commit();
var ropListAll =  sql(`
        select dv.id
        from mct_reftable rt
        join Mct_DimSet ds on ds.idreftable = rt.id
        join Mct_DimVal dv on dv.idDimSet = ds.id
        where rt.id = 47301
        `).batchObjLoad(Mct_DimValApi, "id");
for (rop : ropListAll){
      var manager = Btk_FltPkg.initApiFilter();
          var jFlt = Mct_DimValApi.getFltJObject(rop.idJ());
      Btk_FltPkg.applyJson(manager, jFlt);
          Mct_ObjectFltTechParAliasApi.updateParams(rop.gid(), manager);
}
commit();
```

Восстановление SVP Parent

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var nvCount = sql(`
        select  count(*) as "nCount"
        from mct_specificationverpos svp
        join mct_specificationverpos svp2 on svp.idparent  = svp2.id
        join mct_specificationver sv on svp.idmctdocumentver = sv.id
        join mct_specification s on sv.idmctdocument  = s.id
        join btk_objecttype bo on s.idobjecttype = bo.id
        where svp.idmctdocumentver != svp2.idmctdocumentver
        `).asSingle().nCount;
var nvIter = nvCount / 5000 + 1;
for (i : (1 .. nvIter)){
        var ropList =  sql(`
                select
                        svp.id
                from mct_specificationverpos svp
                join mct_specificationverpos svp2 on svp.idparent  = svp2.id
                join mct_specificationver sv on svp.idmctdocumentver = sv.id
                join mct_specification s on sv.idmctdocument  = s.id
                join btk_objecttype bo on s.idobjecttype = bo.id
                where svp.idmctdocumentver != svp2.idmctdocumentver
                order by svp.id
                limit 5000
                `).batchObjLoad(Mct_SpecificationVerPosApi, "id");
        for (rop : ropList){
                Mct_SpecificationVerPosApi.setidParent(rop, null);
        }
        commit();
}
```

Repait Tech Proc Ver Id State MC

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var select =  sql(`
        select
                tpv.id
                ,bc.nOrder as "nOrder"
        from mct_techprocver tpv
        join btk_classstate bc on tpv.idstate = bc.id
        where tpv.idstatemc  is null`)
select.batchObjLoad(Mct_TechProcVerApi, "id");
select.foreach(function(r){
        var rop = Mct_TechProcVerApi.load(r.id);
        Mct_TechProcVerApi.setidStateMC(rop, r.nOrder);
});
commit();
```

Добавление TTWO Def Group

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var ropList =  sql(`
        select id from Mct_TypalTechWorkOrder
        `).batchObjLoad(Mct_TypalTechWorkOrderApi, "id");
for (rop : ropList){
        Mct_TTWOGroupApi.registerDefaultGroup(rop.idJ());
}
commit();
```

Jexl Update OS Material List

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var nvCount = sql(`
        select count(distinct osv.gid )         as "nCount"
        from mct_ordersheet os
        join mct_ordersheetver osv on osv.idmctdocument = os.id and osv.idstatemc = 300
        join mct_ordersheetverdet d on d.idordersheetver = osv.id
        join mct_journal j on j.gidsrcver = osv.gid
        join mct_materiallist ml on ml.idjournal = j.id
        where (os.bnotformingneed = 1
        or d.bpurchnotreq = 1)
        and ml.bNotFormNeed is null
        `).asSingle().nCount;
var nvIter = nvCount / 100 + 1;
for (i : (1 .. nvIter)){
        sql(`
                select distinct
                        osv.gid as "gid"
                from mct_ordersheet os
                join mct_ordersheetver osv on osv.idmctdocument = os.id and osv.
↪idstatemc = 300
                join mct_ordersheetverdet d on d.idordersheetver = osv.id
                join mct_journal j on j.gidsrcver = osv.gid
                join mct_materiallist ml on ml.idjournal = j.id
                where (os.bnotformingneed = 1
                or d.bpurchnotreq = 1)
                and ml.bNotFormNeed is null
                order by osv.gid
                limit 100`).foreach(function(f){
                        Mct_JournalPkg.fillByOrderSheet(f.gid);
                });
        commit();
}
```

Jexl Update WS Material List

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var nvCount = sql(`
        select count(distinct ws.gid) as "nCount"
        from mct_materiallist ml
        join mct_journal j on ml.idjournal = j.id and j.blastversion = 1
        join mct_workstructure ws on j.gidsrc = ws.gid
        where ml.swssource is null
        `).asSingle().nCount;
var nvIter = nvCount / 100 + 1;
for (i : (1 .. nvIter)){
        sql(`
                select distinct
                        ws.gid as "gid"
                from mct_materiallist ml
                join mct_journal j on ml.idjournal = j.id and j.blastversion = 1
                join mct_workstructure ws on j.gidsrc = ws.gid
                where ml.swssource is null
                order by ws.gid
                limit 100`).foreach(function(f){
                        Mct_JournalPkg.fillByWorkStructure(f.gid, true);
                });
        commit();
}
```

Updade Cross Link Type

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvOld = Mct_CrossLinkTypeApi.findByMnemoCode("Bss->Room");
if(idvOld != null){
        var ropOld = Mct_CrossLinkTypeApi.load(idvOld);
        Mct_CrossLinkTypeApi.setsCode(ropOld, "Str->Room");
        Mct_CrossLinkTypeApi.setsCaption(ropOld, "Связь с помещением");
        commit();
};
```

Обновление BSS And Room For WS By Prj Ver

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
for (i : (1 .. 100)){
        var nvOffset = (i-1) * 500
        var svSelect = sql(`
                select
                        ws.id as "id"
                from mct_workstructure ws
                join bs_prjVer pv on ws.idPrjVer = pv.id
                where pv.scode = '1112'
                order by ws.id
                `
                + "offset " + nvOffset + " limit 500");
        svSelect.batchObjLoad(Mct_WorkStructureApi, "id");
        svSelect.foreach(function(r) {
                Mct_ActionRefApi.refreshBSSAndRoomsForWS(r.id, true, true);
        });
        commit();
}
```

Обновление DVP By OS

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvOS = null;
if (idvOS != null) {
        var gidvDoc = Mct_OrderSheetApi.getGid(idvOS);
        var idvEskd = Mct_EskdApi.findByMnemoCode("NotInESKD");
        //проставляем в документ группу ескд "NotInEskd"
        tsql("update Mct_Document set idEskd = " + idvEskd + " where gidRef = '" +␣
↪gidvDoc + "' ").execute();
        var gidvDocVer = Mct_OrderSheetApi.getGidLastVer(idvOS);
        //генерим записи в Mct_DocumentVerPos по последней версии
        tsql(`with ver as (
                        select '` + gidvDocVer + `' as gid
                )
                ,d as (
                        select
                                (select gid from ver) as gidVer
                                ,getGidId((select gid from ver)) as idVer
                                ,(select id from Btk_Class where sName = 'Mct_
↪OrderSheetVerDet') as idOSVDClass
                                ,(select id from Btk_Class where sName = 'Mct_
↪DocumentVerPos') as idDVPClass
                )
                , ins as (
                        select
                                osvd.*
                        from MCt_OrderSheetVerDet osvd
                        left join Mct_DocumentVerPos as dvp on osvd.gid = dvp.gidRef
                        where osvd.idordersheetver = (select idVer from d)
                        and dvp.gidRef is null
                )
                insert into Mct_DocumentVerPos(
                        gidRef
                        ,idClass
                        ,gidMctDocumentVer
                        ,gidParent
                        ,idPosType
                        ,sPosition
                        ,sDesignation
                        ,sCaption
                        ,nQtyTotal
                        ,nQty
                        ,nNormPerUnit
                        ,nUseCoef
                        ,nMass
                        ,idMsr
                        ,idMsrNorm
                        ,idGoods
                        ,sNote
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                        ,nMassFull
                        ,sLocatedPlace
                        ,dcreatedate_dz
                )
                select
                        i.gid
                        ,(select idDVPClass from d)
                        ,(select gidVer from d)
                        ,(select idOSVDClass from d)|||'/'|||i.idParent
                        ,i.idPosType
                        ,i.sPosition
                        ,i.sDesignation
                        ,i.sMatCaption
                        ,i.nQty
                        ,i.nNetQty
                        ,i.nQty / nullif(i.nNetQty, 0)
                        ,i.nUseCoef
                        ,i.nMass
                        ,i.idMsrQty
                        ,i.idMsrQty
                        ,i.idGoods
                        ,i.sNote
                        ,i.nMassFull
                        ,i.sLocatedPlace
                        ,now()
                from ins i`).execute();
        //генерим КС
        Mct_StructureGenPkg.checkExistanceInStruct(gidvDocVer);
        Mct_StructureGenPkg.structGenFromDoc(gidvDocVer);
        commit();
};
```

Обновление Doc SC

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
Btk_Pkg.setRWSharedUOWEditType();
var idvScShipSpec = Mct_SpecificationApi.idScShipSpec();
if(idvScShipSpec != null){
        var idvShipDraftWork = Btk_ObjectTypeApi.findByMnemoCode("ShipDraftWork");
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        if(idvShipDraftWork != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.load(idvShipDraftWork),
↪ idvScShipSpec);
        }
        var idvShipDraft = Btk_ObjectTypeApi.findByMnemoCode("ShipDraft");
        if(idvShipDraft != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.load(idvShipDraft),␣
↪idvScShipSpec);
        }
}
var idvScMountSpec = Mct_SpecificationApi.idScMountSpec();
if(idvScMountSpec != null){
        var idvMountDraftWork = Btk_ObjectTypeApi.findByMnemoCode("MountDraftWork");
        if(idvMountDraftWork != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.
↪load(idvMountDraftWork), idvScMountSpec);
        }
        var idvMountDraft = Btk_ObjectTypeApi.findByMnemoCode("MountDraft");
        if(idvMountDraft != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.load(idvMountDraft),␣
↪idvScMountSpec);
        }
}
var idvScUnitProduct = Mct_SpecificationApi.idScUnitProduct();
if(idvScUnitProduct != null){
        var idvUnitProduct = Btk_ObjectTypeApi.findByMnemoCode("UnitProduct");
        if(idvUnitProduct != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.load(idvUnitProduct),␣
↪idvScUnitProduct);
        }
}
var idvScMatCard = Mct_MatCardApi.idScMatCard();
if(idvScMatCard != null){
        var idvmatCard = Btk_ObjectTypeApi.findByMnemoCode("matCard");
        if(idvmatCard != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.load(idvmatCard),␣
↪idvScMatCard);
        }
        var idvMatCardPrj = Btk_ObjectTypeApi.findByMnemoCode("MatCardPrj");
        if(idvMatCardPrj != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.load(idvMatCardPrj),␣
↪idvScMatCard);
        }
        var idvMatCardByTech = Btk_ObjectTypeApi.findByMnemoCode("MatCardByTech");
        if(idvMatCardByTech != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.load(idvMatCardByTech),
↪ idvScMatCard);
        }
}
var idvScTechComplectSheet = Mct_TechComplectSheetApi.idScTechComplectSheet();
if(idvScTechComplectSheet != null){
        var idvTechComplectSheet = Btk_ObjectTypeApi.findByMnemoCode("TechComplectSheet
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
↪");
        if(idvTechComplectSheet != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.
↪load(idvTechComplectSheet), idvScTechComplectSheet);
        }
}
var idvScUnitProdSheet = Mct_UnitProdSheetApi.idScUnitProdSheet();
if(idvScUnitProdSheet != null){
        var idvUnitProdSheet = Btk_ObjectTypeApi.findByMnemoCode("UnitProdSheet");
        if(idvUnitProdSheet != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.load(idvUnitProdSheet),
↪ idvScUnitProdSheet);
        }
        var idvUnitProdSheetPrj = Btk_ObjectTypeApi.findByMnemoCode("UnitProdSheetPrj");
        if(idvUnitProdSheetPrj != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.
↪load(idvUnitProdSheetPrj), idvScUnitProdSheet);
        }
}
var idvScPlanLaborDistribution = Mct_PlanLaborDistributionApi.
↪idScPlanLaborDistribution();
if(idvScPlanLaborDistribution != null){
        var idvPlanLaborDistribution = Btk_ObjectTypeApi.findByMnemoCode(
↪"PlanLaborDistribution");
        if(idvPlanLaborDistribution != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.
↪load(idvPlanLaborDistribution), idvScPlanLaborDistribution);
        }
        var idvPlanLaborList = Btk_ObjectTypeApi.findByMnemoCode("PlanLaborList");
        if(idvPlanLaborList != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.load(idvPlanLaborList),
↪ idvScPlanLaborDistribution);
        }
}
var idvScTechCompJournalDoc = Mct_TechCompJournalDocApi.idScTechCompJournalDoc();
if(idvScTechCompJournalDoc != null){
        var idvTechCompJournalDoc = Btk_ObjectTypeApi.findByMnemoCode("TechCompJournalDoc
↪");
        if(idvTechCompJournalDoc != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.
↪load(idvTechCompJournalDoc), idvScTechCompJournalDoc);
        }
}
//ВЗ материалы
var idvScMaterial = Mct_OrderSheetApi.idScMaterial();
if(idvScMaterial != null){
        var idvOrderSheet = Btk_ObjectTypeApi.findByMnemoCode("OrderSheet");
        if(idvOrderSheet != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.load(idvOrderSheet),␣
↪idvScMaterial);
        }
        var idvOrderSheetPrj = Btk_ObjectTypeApi.findByMnemoCode("OrderSheetPrj");
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        if(idvOrderSheetPrj != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.load(idvOrderSheetPrj),
↪ idvScMaterial);
        }
        var idvOrderSheetSupply = Btk_ObjectTypeApi.findByMnemoCode("OrderSheetSupply");
        if(idvOrderSheetSupply != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.
↪load(idvOrderSheetSupply), idvScMaterial);
        }
        var idvOrderSheetSupplyPrj = Btk_ObjectTypeApi.findByMnemoCode(
↪"OrderSheetSupplyPrj");
        if(idvOrderSheetSupplyPrj != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.
↪load(idvOrderSheetSupplyPrj), idvScMaterial);
        }
        var idvPreOrderSheet = Btk_ObjectTypeApi.findByMnemoCode("PreOrderSheet");
        if(idvPreOrderSheet != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.load(idvPreOrderSheet),
↪ idvScMaterial);
        }
        var idvOrderSheetVsp = Btk_ObjectTypeApi.findByMnemoCode("OrderSheetVsp");
        if(idvOrderSheetVsp != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.load(idvOrderSheetVsp),
↪ idvScMaterial);
        }
        var idvOrderSheetVspPrj = Btk_ObjectTypeApi.findByMnemoCode("OrderSheetVspPrj");
        if(idvOrderSheetVspPrj != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.
↪load(idvOrderSheetVspPrj), idvScMaterial);
        }
        var idvOrderSheetTechMat = Btk_ObjectTypeApi.findByMnemoCode("OrderSheetTechMat
↪");
        if(idvOrderSheetTechMat != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.
↪load(idvOrderSheetTechMat), idvScMaterial);
        }
}
//ВЗ оборудование
var idvScEquip = Mct_OrderSheetApi.idScEquip();
if(idvScEquip != null){
        var idvOrderSheetEquip = Btk_ObjectTypeApi.findByMnemoCode("OrderSheetEquip");
        if(idvOrderSheetEquip != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.
↪load(idvOrderSheetEquip), idvScEquip);
        }
        var idvOrderSheetEquipPrj = Btk_ObjectTypeApi.findByMnemoCode("OrderSheetEquipPrj
↪");
        if(idvOrderSheetEquipPrj != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.
↪load(idvOrderSheetEquipPrj), idvScEquip);
        }
        var idvPreOrderSheetEquip = Btk_ObjectTypeApi.findByMnemoCode("PreOrderSheetEquip
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
↪");
        if(idvPreOrderSheetEquip != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.
↪load(idvPreOrderSheetEquip), idvScEquip);
        }
        var idvOrderSheetZip = Btk_ObjectTypeApi.findByMnemoCode("OrderSheetZip");
        if(idvOrderSheetZip != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.load(idvOrderSheetZip),
↪ idvScEquip);
        }
        var idvOrderSheetZipPrj = Btk_ObjectTypeApi.findByMnemoCode("OrderSheetZipPrj");
        if(idvOrderSheetZipPrj != null){
                Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.
↪load(idvOrderSheetZipPrj), idvScEquip);
        }
}

commit();
```

Обновление Doc Ver Pos Pos Action

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var select = sql(`
        select
                dvp.gidref as "gidRef"
                ,svp.idposaction as "idPosAction"
        from mct_specificationverpos svp
        left join mct_documentverpos dvp on svp.gid = dvp.gidref
        where svp.idposaction is not null
        and svp.idposaction is distinct from dvp.idPosAction
        and dvp.gidref is not null
`);
select.batchObjLoadMixin(Mct_DocumentVerPosApi, "gidRef");
select.foreach(function(r){
        var rop = Mct_DocumentVerPosApi.loadByGid(r.gidRef);
        Mct_DocumentVerPosApi.setidPosAction(rop, r.idPosAction);
});
commit();
```

Обновление Eta Mct Tab Class

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvTab = Btk_TabApi.findByMnemoCode("Eta_DocAvi.List_RKDDocument");
if(idvTab != null){
        var ropTab = Btk_TabApi.load(idvTab);
        Btk_TabClassApi.register(ropTab, Mct_SpecificationApi.idClass());
        commit();
}
```

Обновление Isolation OT

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvDefOT = Btk_ObjectTypeApi.getDefaultObjType(Mct_IsolationApi.idClass(), null);
var ropList = sql(`
        select id from Mct_Isolation g
        `).batchObjLoad(Mct_IsolationApi, "id");
for (rop : ropList){
        Mct_IsolationApi.setidObjectType(rop, idvDefOT);
}
commit();
```

Обновление Journal

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var nvCount = sql(`
        select
        count(j.id ) as "nCount"
        from mct_journal j
        join btk_objecttype ot on j.idobjecttype = ot.id
        where coalesce(cast(j.jobjattrs_dz ->> 'bProcessed' as numeric), 0) = 0
        and ot.scode in ('InitMatList', 'WorkSpSheet', 'PkMatSheet')
        `).asSingle().nCount;
var nvIter = nvCount / 100 + 1;
for (i : (1 .. nvIter)){
        var ropList =  sql(`
                select
                        j.id
                from mct_journal j
                join btk_objecttype ot on j.idobjecttype = ot.id
                where coalesce(cast(j.jobjattrs_dz ->> 'bProcessed' as numeric), 0) = 0
                and ot.scode in ('InitMatList', 'WorkSpSheet', 'PkMatSheet')
                order by j.id
                limit 100
                `).batchObjLoad(Mct_JournalApi, "id");
        for (rop : ropList){
                Mct_JournalPkg.updateJournal(rop);
                Mct_JournalApi.setAttrValue(rop, "bProcessed", 1B);
        }
        commit();
}
```

Обновление OS Journal

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var nvCount = sql(`
        select
                count(*) as "nCount"
        from mct_ordersheetver sv
        join mct_ordersheet s on sv.idmctdocument = s.id
        where sv.idstatemc = 300
        and exists (
                select 1
                from mct_ordersheetverdet svp
                where svp.idordersheetver  = sv.id
                and svp.nqty > 0
                and svp.idgoods is not null
        )
        `).asSingle().nCount;
var nvIter = nvCount / 100 + 1;
for (i : (1 .. nvIter)){
        var nvOffset = (i-1) * 100
        var sel =  sql(`
                select
                        sv.gid
                        ,sv.id
                from mct_ordersheetver sv
                join mct_ordersheet s on sv.idmctdocument = s.id
                where sv.idstatemc = 300
                and exists (
                        select 1
                        from mct_ordersheetverdet svp
                        where svp.idordersheetver  = sv.id
                        and svp.nqty > 0
                        and svp.idgoods is not null
                )
                order by sv.id
                        `
                        + "offset " + nvOffset + " limit 100")
        sel.batchObjLoad(Mct_OrderSheetVerApi, "id");
        sel.foreach(function(r){
                Mct_JournalPkg.fillByOrderSheet(r.gid);
        });
        commit();
}
```

Обновление OS Sfi

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvSfiNotInSfi = Mct_SfiApi.idNotInSFI();
var nvCount = sql(`
        select count(*) as "nCount"
        from Mct_OrderSheet s
        where s.idsfi is null
        `).asSingle().nCount;
var nvIter = nvCount / 5000 + 1;
for (i : (1 .. nvIter)){
        sql(`
                select t.id as "id"
                from Mct_OrderSheet t
                where t.idsfi is null
                order by t.id
                limit 5000
                `).foreach(function(r){
                        var rop = Mct_OrderSheetApi.load(r.id);
                        Mct_OrderSheetApi.setidSfi(rop, idvSfiNotInSfi);
                });
        commit();
}
```

Обновление Ot Spec SC

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvSubClassSpec = Btk_SubClassApi.findByMnemoCode("Mct_Specification");
var ropav = sql(`
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        select ot.id
        from Btk_ObjectType ot
        left join Btk_Class c on ot.idrefclass = c.id
        where ot.idsubclass is null
        and c.sname = 'Mct_Specification'
`).batchObjLoad(Btk_ObjectTypeApi, "id");
Btk_Pkg.setRWSharedUOWEditType();
for (rop : ropav) {
        Btk_ObjectTypeApi.setidSubClass(rop, idvSubClassSpec);
};
commit();
```

Обновление Plan Lab Distr Ver State Old

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvClassState = Btk_ClassStateApi.findByNameAndIdClass("InArchive", Mct_
↪PlanLaborDistributionVerApi.idClass(), 0B);
var idvState = Btk_StateApi.findByMnemoCode("Old");
if (idvClassState != null && idvState != null){
        Btk_Pkg.setRWSharedUOWEditType();
        var rop = Btk_ClassStateApi.load(idvClassState);
        Btk_ClassStateApi.setidState(rop, idvState);
        Btk_ClassStateApi.setnOrder(rop, 350B);
        commit();
};
```

Обновление Plan Labor Dist State

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvClassState = Btk_ClassStateApi.findByNameAndIdClass("InArchive", Mct_
↪PlanLaborDistributionVerApi.idClass(), 0B);
if (idvClassState != null){
        var idvStateOld = Btk_StateApi.findByMnemoCode("Old");
        if (idvStateOld != null) {
                Btk_Pkg.setRWSharedUOWEditType();
                Btk_ClassStateApi.setidState(Btk_ClassStateApi.load(idvClassState),␣
↪idvStateOld);
                commit();
        }
};
```

Обновление Res Tech Proc

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var nvCount = sql(`
        select count(*) as "nCount"
        from (
                select tpv.id
                from mct_techProc tp
                join btk_objectType ot on tp.idObjectType = ot.id
                join Btk_SubClass sc on ot.idsubclass = sc.id
                join mct_techProcVer tpv on (tpv.idmctdocument = tp.id and tpv.bislast =␣
↪1 and tpv.idstatemc = 300)
                where sc.scode = 'Mct_TechProc_UnitProduct'
                and exists (
                        select 1
                        from mct_techproclist tpl
                        join mct_techproclistnorm tpln on tpl.id = tpln.idtechproclist
                        join mct_techprocnorm tpn on tpln.idtechprocnorm = tpn.id
                        where tpl.idtechprocver = tpv.id
                        and tpn.idresource is not null
                        )
        ) t
        `).asSingle().nCount;
var nvIter = nvCount / 300 + 1;
for (i : (1 .. nvIter)){
        var ropList =  sql(`
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                select tpv.id
                from mct_techProc tp
                join btk_objectType ot on tp.idObjectType = ot.id
                join Btk_SubClass sc on ot.idsubclass = sc.id
                join mct_techProcVer tpv on (tpv.idmctdocument = tp.id and tpv.bislast =␣
↪1 and tpv.idstatemc = 300)
                where sc.scode = 'Mct_TechProc_UnitProduct'
                and exists (
                        select 1
                        from mct_techproclist tpl
                        join mct_techproclistnorm tpln on tpl.id = tpln.idtechproclist
                        join mct_techprocnorm tpn on tpln.idtechprocnorm = tpn.id
                        where tpl.idtechprocver = tpv.id
                        and tpn.idresource is not null
                        )
                order by tpv.id
                `
                + " limit 300 offset " + ((i-1) * 300)).batchObjLoad(Mct_TechProcVerApi,
↪"id");
        for (rop : ropList){
                Mct_TechProcApi.registerResTechProc(rop);
        }
        commit();
}
```

Обновление Sfi Struct

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
sql(`
        select
                pv.id as "id"
        from bs_prj  p
        join bs_prjver pv on pv.idproj = p.id
        where sCDClassifier = 'SFI'
        and exists (
                select 1
                from mct_document d
                where d.idprjver = pv.id
                and d.gidlastverinuse is not null
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                )
        order by pv.id
`).foreach(function(r){
        var nvCount = sql(`
                select count(*) as "nCount"
                from mct_specification s
                join mct_document d on s.gid = d.gidref
                where s.idprjver = `+r.id+`
                and d.gidlastverinuse is not null
                `).asSingle().nCount;
        var nvIter = nvCount / 200 + 1;
        for (i : (1 .. nvIter)){
                var nvOffset = (i-1) * 200
                sql(`
                        select s.gid as "gid"
                        from mct_specification s
                        join mct_document d on s.gid = d.gidref
                        where s.idprjver = `+r.id+`
                        and d.gidlastverinuse is not null
                        order by s.id
                        `
                        + "offset " + nvOffset + " limit 200").foreach(function(f){
                                flush();
                                Mct_StructureGenPkg.updateStructureByDoc(f.gid, Mct_
↪StructViewTypeApi.idSFI());
                        });
                commit();
        }
});
```

Обновление Sfi Struct By Prj Ver

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvPrjVer = 68652;
var nvCount = sql(`
        select count(*) as "nCount"
        from mct_specification s
        join mct_document d on s.gid = d.gidref
        where s.idprjver = `+idvPrjVer+`
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        and d.gidlastverinuse is not null
        `).asSingle().nCount;
var nvIter = nvCount / 100 + 1;
for (i : (1 .. nvIter)){
        var nvOffset = (i-1) * 100
        sql(`
                select s.gid as "gid"
                from mct_specification s
                join mct_document d on s.gid = d.gidref
                where s.idprjver = `+idvPrjVer+`
                and d.gidlastverinuse is not null
                order by s.id
                `
                + "offset " + nvOffset + " limit 200").foreach(function(f){
                        flush();
                        Mct_StructureGenPkg.updateStructureByDoc(f.gid, Mct_
↪StructViewTypeApi.idSFI());
                });
        commit();
}
```

Обновление Spec Journal

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var nvCount = sql(`
        select
                count(*) as "nCount"
        from mct_specificationver sv
        join mct_specification s on sv.idmctdocument = s.id
        left join mct_objecttype mot on mot.idbtkobjecttype = s.idobjecttype
        where sv.idstatemc = 300
        and coalesce(mot.bisprjdoc, 0) = 0
        `).asSingle().nCount;
var nvIter = nvCount / 100 + 1;
for (i : (1 .. nvIter)){
        var nvOffset = (i-1) * 100
        var sel =  sql(`
                select
                        sv.gid
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                        ,sv.id
                from mct_specificationver sv
                join mct_specification s on sv.idmctdocument = s.id
                left join mct_objecttype mot on mot.idbtkobjecttype = s.idobjecttype
                where sv.idstatemc = 300
                and coalesce(mot.bisprjdoc, 0) = 0
                order by sv.id
                        `
                        + "offset " + nvOffset + " limit 100")
        sel.batchObjLoad(Mct_SpecificationVerApi, "id");
        sel.foreach(function(r){
                Mct_JournalPkg.fillBySpecification(r.gid);
        });
        commit();
}
```

Обновление Spec Sfi

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvSfiNotInSfi = Mct_SfiApi.idNotInSFI();
var nvCount = sql(`
        select count(*) as "nCount"
        from mct_Specification s
        where s.idsfi is null
        `).asSingle().nCount;
var nvIter = nvCount / 5000 + 1;
for (i : (1 .. nvIter)){
        sql(`
                select t.id as "id"
                from mct_Specification t
                where t.idsfi is null
                order by t.id
                limit 5000
                `).foreach(function(r){
                        var rop = Mct_SpecificationApi.load(r.id);
                        Mct_SpecificationApi.setidSfi(rop, idvSfiNotInSfi);
                });
        commit();
}
```

Обновление State Doc Tech Proc

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var ropList =  sql(`
        select id from mct_techproc
        where idStateDoc is null
        `).batchObjLoad(Mct_TechProcApi, "id");
for (rop : ropList){
        var idvOT = rop.copyAro().idObjectType();
        var idvStateDoc = Btk_StateDocApi.getFirstStateDocByOT(idvOT);
        Mct_TechProcApi.setidStateDoc(rop, idvStateDoc);
}
commit();
```

Обновление Struct View Type Can Choose In Struct

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvBSStructViewType = Mct_StructViewTypeApi.idBS();
if(idvBSStructViewType != null){
        var ropBSSVT = Mct_StructViewTypeApi.load(idvBSStructViewType);
        Mct_StructViewTypeApi.setbCanChooseInStruct(ropBSSVT, 1B);
}
commit();
```

Обновление TPL Res Path

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var ropav = sql(`
        select
                tpl.id
        from mct_techproclist tpl
        left join mct_postype pt on tpl.idpostype = pt.id
        where pt.scode != 'МСЧ'
        and srespath is null
        and exists (
                select 1 from mct_techproclistnorm tln
                where tln.idtechproclist = tpl.id
                )
`).batchObjLoad(Mct_TechProcListApi, "id");
for (rop : ropav) {
        var svResPath = Mct_TechProcListApi.getResPath(rop, null).getCaption("-");
        Mct_TechProcListApi.setsResPath(rop, svResPath);
};
commit();
```

Обновление TPL b Manual

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var ropList =  sql(`
        select id from mct_techproclist
        where idposreg is null
        and coalesce(bManual, 0) = 0
        `).batchObjLoad(Mct_TechProcListApi, "id");
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
for (rop : ropList){
        Mct_TechProcListApi.setbManual(rop, 1B);
}
commit();
```

Обновление TTWO Group

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvGroup = Btk_ClassApi.load(Mct_TypalTechWorkOrderApi.idClass()).copyAro().
↪idGroupRoot();
var ropList =  sql(`
        select id from Mct_TypalTechWorkOrder
        `).batchObjLoad(Mct_TypalTechWorkOrderApi, "id");
for (rop : ropList){
        Btk_ObjectGroupApi.register(rop, idvGroup, 0B, 1B);
}
commit();
```

Обновление Tech Proc Full Route

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
for (i : (1 .. 400)){
        var nvOffset = (i-1) * 500
        var ropList =  sql(`
                select t.id as "id"
                from mct_techProc t
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                join Btk_ObjectType ot on t.idObjectType = ot.id
                join mct_techprocver tpv on tpv.idmctdocument = t.id and tpv.bislast = 1
                where ot.sCode = 'techProcUnitProd'
                and not exists (
                        select 1
                        from mct_techproclist tpl
                        join mct_techproclistnorm tpln on tpl.id = tpln.idtechproclist
                        where tpl.idtechprocver = tpv.id
                        and tpln.idtechprocnorm is null
                )
                order by t.id
                `
                + "offset " + nvOffset + " limit 500").batchObjLoad(Mct_TechProcApi, "id
↪");
        for (rop : ropList){
                Mct_TechProcApi.updateFullRoute(rop, false);
        }
        commit();
}
```

Обновление Tech Proc Norm Pos Reg

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
Mct_TechProcNormApi.generateTechProcNormPosReg();
commit();
```

Обновление Unit Prod Sheet List Manual

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
update Mct_UnitProdSheetList
set bmanual = 1
where bpurchase = 1
```

Обновление Ver Expiry Date

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var nvRowPerIter = 25
sql(`
        select
                c.sname as "sName"
        from btk_classmixin cm
        join btk_class c on cm.idbtkclass = c.id
        where cm.ssystemname = 'Mct_DocumentVersion'
`).foreach(function(r){
        var svClass = r.sName;
        var api = Btk_ClassApi.getApiBySimpleClassName(svClass);
        var nvCount = sql(`
                select
                        count(*) as "nCount"
                from ` + svClass +` v
                join mct_documentversion dv on v.gid = dv.gidref
                join mct_documentversion dvn on (dvn.gidmctdocument = dv.gidmctdocument␣
↪and dvn.nnumber = dv.nnumber + 1)
                where v.idstatemc = 300
                and dvn.idstatemc = 300
                and dvn.daccept is not null
                and dv.dexpirydate is null
        `).asSingle().nCount;
        var nvIter = nvCount / nvRowPerIter + 1;
        for (i : (1 .. nvIter)){
                var select = sql(`
                        select
                                v.id
                                ,dv.gidRef as "gidRef"
                                ,dvn.daccept as "dExpiryDate"
                                ,dv.gidmctdocument
                        from ` + svClass +` v
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                        join mct_documentversion dv on v.gid = dv.gidref
                        join mct_documentversion dvn on (dvn.gidmctdocument = dv.
↪gidmctdocument and dvn.nnumber = dv.nnumber + 1)
                        where v.idstatemc = 300
                        and dvn.idstatemc = 300
                        and dvn.daccept is not null
                        and dv.dexpirydate is null
                        order by v.id
                        limit ` + nvRowPerIter.toString());
                select.batchObjLoad(api, "id");
                select.foreach(function(r){
                        var rop = api .load(r.id);
                        api .setdExpiryDate(rop, r.dExpiryDate);
                });
                commit();
        };
});
```

Обновление WS Journal

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var nvCount = sql(`
        with states as (
                select
                        cast(jsonb_array_elements(jWSJournalFillState) as bigint) as id
                from mct_setting ms
                )
        ,ws as (
                select
                        ws.gid
                from mct_workstructure ws
                join states s on ws.idstate = s.id
                join Mct_WrkStructSet wss on ws.idPostype = wss.idpostype
                join  mct_workstructuregds wsg on wsg.gidSrc = ws.gid
                where coalesce(ws.bImpExtSystem, 0) = 0
                and wss.bFillMaterialJournal = 1
                --and not exists (select 1 from mct_materiallistSrc mls where mls.gidsrc␣
↪= wsg.gid)
                group by ws.gid
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        )
        select count(*) as "nCount" from ws
        `).asSingle().nCount;
var nvIter = nvCount / 100 + 1;
for (i : (1 .. nvIter)){
        var nvOffset = (i-1) * 100
        var sel =  sql(`
                with states as (
                        select
                                cast(jsonb_array_elements(jWSJournalFillState) as␣
↪bigint) as id
                        from mct_setting ms
                        )
                        select
                                ws.id
                                ,ws.gid
                        from mct_workstructure ws
                        join states s on ws.idstate = s.id
                        join Mct_WrkStructSet wss on ws.idPostype = wss.idpostype
                        join  mct_workstructuregds wsg on wsg.gidSrc = ws.gid
                        where coalesce(ws.bImpExtSystem, 0) = 0
                        and wss.bFillMaterialJournal = 1
                        group by ws.id
                        order by ws.id
                        `
                        + "offset " + nvOffset + " limit 100")
        sel.batchObjLoad(Mct_WorkStructureApi, "id");
        sel.foreach(function(r){
                Mct_JournalPkg.fillByWorkStructure(r.gid, true);
        });
        commit();
}
```

Обновление WS Oper Order

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var nvCount = sql(`
        select count(*) as "nCount"
        from mct_workstructureoperation
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        where sOrder is null
        and sCode is not null
        `).asSingle().nCount;
var nvIter = nvCount / 5000 + 1;
for (i : (1 .. nvIter)){
        var ropList =  sql(`
                select t.id as "id"
                from mct_workstructureoperation t
                where t.sOrder is null
                and t.sCode is not null
                order by t.id
                limit 5000
                `).batchObjLoad(Mct_WorkStructureOperationApi, "id");
        for (rop : ropList){
                Mct_WorkStructureOperationApi.updateOrderAttr(rop);
        }
        commit();
}
```

Обновление WS Order

Локальный скрипт модуля **MCT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var nvCount = sql(`
        select count(*) as "nCount"
        from mct_workstructure
        where sOrder is null
        and sCode is not null
        `).asSingle().nCount;
var nvIter = nvCount / 5000 + 1;
for (i : (1 .. nvIter)){
        var ropList =  sql(`
                select t.id as "id"
                from mct_workstructure t
                where t.sOrder is null
                and t.sCode is not null
                order by t.id
                limit 5000
                `).batchObjLoad(Mct_WorkStructureApi, "id");
        for (rop : ropList){
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                Mct_WorkStructureApi.updateOrderAttr(rop);
        }
        commit();
}
```

Добавление разделов в рабочую структуру

Добавляет ссылки на разделы в рабочую структуру на основе сопоставления элементов структуры проекта. Используется для восстановления или донастройки связей между рабочей структурой и разделами.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
sql(`
        select distinct
                ws.id as "idWorkStructure"
                ,ps.snote
                ,ss.id as "idSection"
        from mct_actionref ar
        join mct_workstructure ws on ar.idWorkStructure = ws.id
        join bs_prjver pv on ws.idprjver = pv.id
        join mct_structure ps on ar.idstructure = ps.id
        join mct_structure ss on (ss.idprjver = pv.id and ps.snote = ss.scaption)
        join mct_postype pt on ss.idpostype = pt.id
        left join mct_techreference tr on tr.idworkstructure = ws.id and tr.idstructure␣
↪= ss.id
        where pv.scode = '1006'
        and pt.scode = 'CK'
        and tr.id is null
        `).foreach(function(r){
        var ropParent = Mct_WorkStructureApi.load(r.idWorkStructure);
        var rop = Mct_TechReferenceApi.insertByParent(ropParent);
        Mct_TechReferenceApi.setidStructure(rop, r.idSection);
});
commit();
```

Удаление неиспользуемых спецификаций единиц продукции

Удаляет спецификации, связанные с ТМЦ по заданным условиям, если они не используются в списках единиц продукции. Перед удалением последняя версия переводится в редактируемое состояние и удаляется из структуры.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var nvCount = 400
var nvIter = nvCount / 10 + 1;
for (i : (1 .. nvIter)){
        var ropList =  sql(`
                select distinct
                        s.id
                from bs_goods g
                join bs_goodssrc gs on g.id = gs.idgds
                join mct_specification s on gs.gidsrc = s.gid
                left join mct_unitprodsheetlist ul on s.id = ul.idspecification
                where (upper(g.sdesignation) like 'ТЛИШ.745311.001%'
                or upper(g.sdesignation) like 'ТЛИШ.745311.001%'
                or upper(g.sdesignation) like 'ТЛИШ.363621.069%'
                or upper(g.sdesignation) like 'ТЛИШ.363621.058-%'
                or upper(g.sdesignation) like 'ВПИЕ.632721.012%'
                or upper(g.sdesignation) like 'ИЕАШ.632731.018%'
                or upper(g.sdesignation) like 'ТЛИШ.363611.059%'
                or upper(g.sdesignation) like 'НИМБ.324219.210%'
                or upper(g.sdesignation) like '22350.363212.009%')
                and ul.id is null
                order by s.id
                limit 10
                `).batchObjLoad(Mct_SpecificationApi, "id");
                for (rop : ropList){
                        //@begin{
                                var gidvLastVer = Mct_SpecificationApi.getGidLastVer(rop.
↪idJ())
                                var idvLastVer = parseId(gidvLastVer);
                                var ropLastVer = Mct_SpecificationVerApi.
↪load(idvLastVer);
                                Mct_SpecificationVerApi.setidState(ropLastVer, Mct_
↪SpecificationVerApi.idEditState());
                                Mct_StructureGenPkg.deleteFromStructure(gidvLastVer);
                                flush();
                                Mct_SpecificationApi.delete(rop);
                        //        commit();
                        //}
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                        //@exception
                        // function(exp){
                        //         println('there was an exception');
                        //         rollback();
                        //         }end;
                        }
                        commit();
}
```

Заполнение отсутствующего WBS у позиций спецификаций

Восстанавливает значения WBS у позиций спецификаций, для которых значение не заполнено. Обработка выполняется пакетами с фиксацией изменений после каждой пачки.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var nvCount = sql(`
        select count(distinct s.id) as "nCount"
        from mct_specificationverpos svp
        join mct_specificationver sv on svp.idmctdocumentver = sv.id
        join mct_specification s on s.id = sv.idmctdocument
        where svp.swbs is null
        `).asSingle().nCount;
var nvIter = nvCount / 15 + 1;
for (i : (1 .. nvIter)){
        var select = sql(`
                select distinct s.id
                from mct_specificationverpos svp
                join mct_specificationver sv on svp.idmctdocumentver = sv.id
                join mct_specification s on s.id = sv.idmctdocument
                where svp.swbs is null
                order by s.id
                limit 15
                `);
        select.batchObjLoad(Mct_SpecificationApi, "id");
        select.foreach(function(r){
                Mct_SpecificationVerPosApi.updateWBSBySpec(r.id);
        });
        commit();
}
```

Исправление значений WBS с null у позиций спецификаций

Исправляет значения WBS у позиций спецификаций, в которых строковое значение содержит `null`. Обработка выполняется пакетами с фиксацией изменений после каждой пачки.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var nvCount = sql(`
        select count(distinct s.id) as "nCount"
        from mct_specificationverpos svp
        join mct_specificationver sv on svp.idmctdocumentver = sv.id
        join mct_specification s on s.id = sv.idmctdocument
        where lower(svp.swbs) like '%null%'
        `).asSingle().nCount;
var nvIter = nvCount / 15 + 1;
for (i : (1 .. nvIter)){
        var select = sql(`
                select distinct s.id
                from mct_specificationverpos svp
                join mct_specificationver sv on svp.idmctdocumentver = sv.id
                join mct_specification s on s.id = sv.idmctdocument
                where lower(svp.swbs) like '%null%'
                order by s.id
                limit 15
                `);
        select.batchObjLoad(Mct_SpecificationApi, "id");
        select.foreach(function(r){
                Mct_SpecificationVerPosApi.updateWBSBySpec(r.id);
        });
        commit();
}
```

Генерация ресурсных технологических процессов

Генерирует ресурсные технологические процессы для версий технологических процессов, подходящих под условия отбора. Используется для массового восстановления или создания связанных ресурсных техпроцессов.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
for (i : (1 .. 100)){
        var ropList =  sql(`
                select tpv.id
                from mct_techProc tp
                join btk_objectType ot on tp.idObjectType = ot.id
                join Btk_SubClass sc on ot.idsubclass = sc.id
                join mct_techProcVer tpv on (tpv.idmctdocument = tp.id and tpv.bislast =␣
↪1 and tpv.idstatemc = 300)
                left join mct_techProcVer tpvr on tpvr.idsrctechprocver = tpv.id
                where sc.scode = 'Mct_TechProc_UnitProduct'
                and tpvr.id is null
                and exists (
                        select 1
                        from mct_techproclist tpl
                        join mct_techproclistnorm tpln on tpl.id = tpln.idtechproclist
                        join mct_techprocnorm tpn on tpln.idtechprocnorm = tpn.id
                        where tpl.idtechprocver = tpv.id
                        and tpn.idresource is not null
                        )
                order by tpv.id
                `
                + " limit 500").batchObjLoad(Mct_TechProcVerApi, "id");
        for (rop : ropList){
                Mct_TechProcApi.registerResTechProc(rop);
        }
        commit();
}
```

Генерация проектных спецификаций

Создает или связывает проектные спецификации на основе рабочих спецификаций, проекта и версии проекта. Используется для массового восстановления проектных документов по данным МСТ.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var nvCount = sql(`
        select count(*) as "nCount"
        from (
                select s.sdesignation
                from mct_specification s
                join bs_prj p on s.idprj = p.id
                join bs_prjVer pv on s.idprjver = pv.id
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                join btk_objectType ot on s.idobjecttype = ot.id
                join mct_objecttype mot on mot.idbtkobjecttype = ot.id
                join mct_specificationver sv on sv.idmctdocument = s.id and sv.bislast =␣
↪1
                left join mct_specification sp on sp.scode = s.sdesignation and sp.idPrj␣
↪= s.idPrj
                left join mct_objectType motsp on motsp.idbtkobjecttype = sp.
↪idobjecttype
                left join mct_objectType motp on s.idobjecttype = motp.idwrkdoc
                left join mct_specification sop on sop.scode = s.sdesignation and sop.
↪idPrj != s.idPrj
                left join mct_document d on d.scode = s.sdesignation and getgidclass(d.
↪gidref) != cast(44751 as bigint)
                where coalesce(p.bnotactive, 0) = 0
                        and coalesce(pv.bnotactive, 0) = 0
                        and pv.scode != '03P'
                        and pv.scode != '00000'
                        and coalesce(mot.bisprjdoc, 0) = 0
                        and ot.scode != 'UnitProduct'
                        and s.gidprjdoc is null
                        and sv.idstatemc < 400
                        and s.scode != s.sdesignation
                        and s.sdesignation != ''
                        and (motsp.id is null or motsp.bisprjdoc = 1)
                        and sop.id is null
                        and d.gidref is null
                group by
                        s.sdesignation
                        ,sp.id
                        ,s.idPrj
                        ,s.idDepOwner
        ) t
        `).asSingle().nCount;
var nvIter = nvCount / 100 + 1;
for (i : (1 .. nvIter)){
        var select = sql(`
                select
                        s.sdesignation as "sDesignation"
                        ,sp.id as "idPrjSpec"
                        ,max(motp.idbtkobjecttype) as "idPrjObjectType"
                        ,array_agg(s.id) as "idaWrkDoc"
                        ,s.idPrj as "idPrj"
                        ,s.idDepOwner as "idDepOwner"
                        ,max(s.scaption) as "sCaption"
                from mct_specification s
                join bs_prj p on s.idprj = p.id
                join bs_prjVer pv on s.idprjver = pv.id
                join btk_objectType ot on s.idobjecttype = ot.id
                join mct_objecttype mot on mot.idbtkobjecttype = ot.id
                join mct_specificationver sv on sv.idmctdocument = s.id and sv.bislast =␣
↪1
                left join mct_specification sp on sp.scode = s.sdesignation and sp.idPrj␣
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
↪= s.idPrj
                left join mct_objectType motsp on motsp.idbtkobjecttype = sp.
↪idobjecttype
                left join mct_objectType motp on s.idobjecttype = motp.idwrkdoc
                left join mct_specification sop on sop.scode = s.sdesignation and sop.
↪idPrj != s.idPrj
                left join mct_document d on d.scode = s.sdesignation and getgidclass(d.
↪gidref) != cast(44751 as bigint)
                where coalesce(p.bnotactive, 0) = 0
                and coalesce(pv.bnotactive, 0) = 0
                and pv.scode != '03P'
                and pv.scode != '00000'
                and coalesce(mot.bisprjdoc, 0) = 0
                and ot.scode != 'UnitProduct'
                and s.gidprjdoc is null
                and sv.idstatemc < 400
                and s.scode != s.sdesignation
                and s.sdesignation != ''
                and (motsp.id is null or motsp.bisprjdoc = 1)
                and sop.id is null
                and d.gidref is null
                group by
                        s.sdesignation
                        ,sp.id
                        ,s.idPrj
                        ,s.idDepOwner
                order by s.sdesignation
                limit 100
                `);
        select.foreach(function(r){
                var idvPrjSpec;
                if(r.idPrjSpec == null){
                        var t = Mct_DocPkg.createSpecification(r.idPrjObjectType, r.
↪idPrj, r.idDepOwner, null, null, null, null, null);
                        var rop = Mct_SpecificationApi.load(t);
                        Mct_SpecificationApi.setsCode(rop, r.sDesignation);
                        Mct_SpecificationApi.setsCaption(rop, r.sCaption);
                        idvPrjSpec = t;
                } else {
                        idvPrjSpec = r.idPrjSpec
                };
                var ropPrjSpec = Mct_SpecificationApi.load(idvPrjSpec);
                var gidvPrjLastVer = Mct_DocumentApi.getGidLastVersionByDoc(ropPrjSpec.
↪gid(), null, false, false);
                var ropList = sql(`
                        select
                                s.id
                        from mct_specification s
                        join bs_prj p on s.idprj = p.id
                        join bs_prjVer pv on s.idprjver = pv.id
                        join btk_objectType ot on s.idobjecttype = ot.id
                        join mct_objecttype mot on mot.idbtkobjecttype = ot.id
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                        join mct_specificationver sv on sv.idmctdocument = s.id and sv.
↪bislast = 1
                        where sdesignation = '`+r.sDesignation+`'
                        and idPrj =`+r.idPrj+`
                        and coalesce(p.bnotactive, 0) = 0
                        and coalesce(pv.bnotactive, 0) = 0
                        and pv.scode != '03P'
                        and pv.scode != '00000'
                        and coalesce(mot.bisprjdoc, 0) = 0
                        and ot.scode != 'UnitProduct'
                        and s.gidprjdoc is null
                        and sv.idstatemc < 400
                        and s.scode != s.sdesignation
                `).batchObjLoad(Mct_SpecificationApi, "id");
                for (ropCurWorkSpec : ropList){
                        var gidvWorkSpecVer = Mct_DocumentApi.
↪getGidLastVersionByDoc(ropCurWorkSpec.gid(), null, false, false);
                        Mct_SpecificationApi.setgidPrjDoc(ropCurWorkSpec, ropPrjSpec.
↪gid());
                        //Создадим записи в связанных документах проектной СП
                        Mct_LinkApi.registerByVer(ropPrjSpec.gid(),gidvWorkSpecVer, null,
↪ null);
                        //Создадим записи в связанных документах рабочей СП
                        Mct_LinkApi.registerByVer(ropCurWorkSpec.gid(),gidvPrjLastVer,␣
↪null, null);
                };
        });
        commit();
}
```

Обновление ресурсных техпроцессов покрытий

Обновляет данные ресурсных технологических процессов покрытий по связанным спискам техпроцессов. Используется для массовой корректировки данных МСТ.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
for (i : (1 .. 30)){
        var nvOffset = (i-1) * 500
        var ropList =  sql(`
                select
                        tpl.id
                from mct_techproclist tpl
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                join mct_techprocver tpv on tpl.idtechprocver = tpv.id
                join mct_techproc tp on tpv.idmctdocument = tp.id
                join btk_objecttype ot on tp.idobjecttype = ot.id
                join mct_postype pt on tpl.idpostype = pt.id
                where tpl.idcover is not null
                and ot.scode = 'techProcUnitProd'
                and pt.scode = 'ТМЦ'
                order by tpl.id
                `
                + "offset " + nvOffset + " limit 500").batchObjLoad(Mct_TechProcListApi,
↪"id");
        for (rop : ropList){
                Mct_TechProcListApi.assignCoverResource(rop);
        }
        commit();
}
```

Добавление состояния документа для технологического сборочного узла

Добавляет настройки состояний документа для объектов технологических сборочных узлов. Используется при донастройке документных состояний МСТ.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
for (i : (1 .. 50)){
        //var nvOffset = (i-1) * 500
        var ropList =  sql(`
                select t.id
                from mct_techassemblyunit t
                where not exists (
                        select 1 from mct_documentstatedoc dsd
                        where dsd.gidsrc = t.gid
                )
                order by t.id
                limit 500
                `).batchObjLoad(Mct_TechAssemblyUnitApi, "id");
        for (rop : ropList){
                Mct_DocumentStateDocApi.updateStateDoc(rop.gid(), false);
        }
        commit();
}
```

Обновление дерева элементов спецификации

Обновляет данные дерева элементов спецификации через пакет MCT. Используется для восстановления или актуализации связанных структур спецификаций.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
for (i : (1 .. 200)){
        var nvOffset = (i-1) * 50
        var ropList =  sql(`
                select sv.id
                from mct_specificationver sv
                left join mct_specification s on sv.idmctdocument = s.id
                left join btk_objecttype ot on s.idobjecttype = ot.id
                left join btk_classstate cs on sv.idstate = cs.id
                where cs.norder = 300
                and sv.bislast = 1
                and (ot.scode = 'DraftIVC' or ot.scode = 'DraftWorkIVC')
                `+ "offset " + nvOffset + " limit 50").batchObjLoad(Mct_
↪SpecificationVerApi, "id");
        for (rop : ropList){
                MctPrs_SpecificationPkg.refreshET(rop);
        }
        commit();
}
```

Обновление блока правил автоматического распределения по РС

Обновляет настройки правил автоматического распределения по рабочим структурам. Используется для массовой донастройки правил распределения MCT.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvBlockSormovo = Mct_AutoDistributeWsTypeApi.findByMnemoCode("BlockSormovo");
var ropList =  sql(`
                select t.id
                from Mct_AutoDistributeWsRule t
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                join Mct_AutoDistributeWsType tt on t.idAutoDistributeWsType = tt.id
                where tt.scode = 'Block'
                `).batchObjLoad(Mct_AutoDistributeWsRuleApi, "id");
        for (rop : ropList){
                Mct_AutoDistributeWsRuleApi.setidAutoDistributeWsType(rop,␣
↪idvBlockSormovo);
        }
        commit();
```

CNT

Очистка Contract Tabs

Локальный скрипт модуля **CNT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
Btk_Pkg.setRWSharedUOWEditType();
var ropList = sql(`
select t.id as "id"
from btk_objecttypetab t
left join btk_objectType ot on t.idObjectType = ot.id
where ot.scode = 'Cnt_Contract_Contract'
`).batchObjLoad(Btk_ObjectTypeTabApi, "id");
for (rop : ropList){
        Btk_ObjectTypeTabApi.delete(rop);
}
commit();
```

Удаление Old OT Det

Локальный скрипт модуля **CNT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvTabContract = Btk_ObjectTypeDetApi.findByMnemoCode("Cnt_ContractTypeCard");
if(idvTabContract != null) {
        Btk_ObjectTypeDetApi.delete(Btk_ObjectTypeDetApi.load(idvTabContract));
};
commit();
```

Миграция Curator

Локальный скрипт модуля **CNT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
for (i : (1 .. 200)){
        var nvOffset = (i-1) * 500
        var select = sql(`
                select
                        ss.*
                        ,ost.idResponsibilityCenter as "idResponsibilityCenter"
                from (
                        select
                                s.*
                                ,(
                                select
                                        distinct first_value(oe.
↪idofstructure)over(order by ot.nparentlevel) as idOFS
                                  from bs_ofsemployee oe
                                  left join bs_ofstree ot on oe.idofstructure = ot.
↪idparent
                                  where oe.idemployee=s."idCuratorEmployeeNew"
                                  ) as "idOFSNew"
                        from (
                                select
                                        c.id
                                        ,(select distinct
                                                last_value (e.id) over(order by e.
↪demployment asc ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) as idEmployee
                                        from Bs_Employee e
                                        where e.dDismissal is null
                                        and e.idPerson = c.idCurator) as
↪"idCuratorEmployeeNew"
                                from Cnt_Contract c
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                                where c.idcurator is not null
                                ) s
                        ) ss
                left join bs_ofstructure ost on ss."idOFSNew" = ost.id
                order by ss.id
                offset ` + nvOffset + " limit 500"
                );
        //прогружаем записи
        var ropavContract = select.batchObjLoad(Cnt_ContractApi, "id");
        select.foreach(function(r){
          var rop = Cnt_ContractApi.load(r.id);
          Cnt_ContractApi.dpi().setidCuratorEmployee(rop, r.idCuratorEmployeeNew);
          Cnt_ContractApi.dpi().setidOFSResponsible(rop, r.idOFSNew);
          Cnt_ContractApi.dpi().setidOFSOwner(rop, r.idOFSNew);
          if(r.idResponsibilityCenter != null) {
                Cnt_ContractApi.dpi().setidResponsibilityCenter(rop, r.
↪idResponsibilityCenter);
          }
    });
        commit();
}
```

Обновление Contract OT

Локальный скрипт модуля **CNT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
Btk_Pkg.setRWSharedUOWEditType();
var idvContractType = Btk_ObjectTypeApi.findByMnemoCode("contract");
if(idvContractType != null){
        Btk_ObjectTypeApi.setsCode(Btk_ObjectTypeApi.load(idvContractType), "Cnt_
↪Contract_Contract");
};
var idvContractSubClass = Btk_SubClassApi.findByMnemoCode("contract");
if(idvContractSubClass != null) {
        Btk_SubClassApi.setsCode(Btk_SubClassApi.load(idvContractSubClass), "Cnt_
↪Contract_Contract");
}
var idvStageSubClass = Btk_SubClassApi.findByMnemoCode("stage");
if(idvStageSubClass != null) {
        Btk_SubClassApi.setsCode(Btk_SubClassApi.load(idvStageSubClass), "Cnt_Contract_
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
↪Stage");
}
commit();
```

GDS

Удаление Old Article Contras Tab

Локальный скрипт модуля **GDS**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvTab = Btk_TabApi.findByMnemoCode("Gds_ArticleContrasAvi.List_idGds");
if (idvTab != null) {
        Btk_Pkg.setRWSharedUOWEditType();
        var ropList = sql(`
                select id from btk_objecttypetab
                where idtab = ` + idvTab).batchObjLoad(Btk_ObjectTypeTabApi, "id");
        for (rop : ropList){
                Btk_ObjectTypeTabApi.delete(rop);
        }
        Btk_TabApi.delete(Btk_TabApi.load(idvTab));
        commit();
}
```

Upd Obj Type Goods

Локальный скрипт модуля **GDS**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
Btk_Pkg.setRWSharedUOWEditType();
var idvDevice = Btk_ObjectTypeApi.findByMnemoCode("Device");
var idvGASTypeDevice = Gds_GoodsAndServiceTypeApi.idGASTypeDevice();
if(idvDevice != null && idvGASTypeDevice != null) {
        var rop = Gds_ObjTypeGoodsApi.registerByObjectType(idvDevice);
        Gds_ObjTypeGoodsApi.setidGoodsAndServiceType(rop, idvGASTypeDevice);
}
var idvMaterial = Btk_ObjectTypeApi.findByMnemoCode("Material");
var idvGASTypeMaterial = Gds_GoodsAndServiceTypeApi.idGASTypeMaterial();
if(idvMaterial != null && idvGASTypeMaterial != null) {
        var rop = Gds_ObjTypeGoodsApi.registerByObjectType(idvMaterial);
        Gds_ObjTypeGoodsApi.setidGoodsAndServiceType(rop, idvGASTypeMaterial);
}
var idvMSCH = Btk_ObjectTypeApi.findByMnemoCode("MSCH");
var idvGASTypeMSCH = Gds_GoodsAndServiceTypeApi.idGASTypeMSCH();
if(idvMSCH != null && idvGASTypeMSCH != null) {
        var rop = Gds_ObjTypeGoodsApi.registerByObjectType(idvMSCH);
        Gds_ObjTypeGoodsApi.setidGoodsAndServiceType(rop, idvGASTypeMSCH);
}
var idvAssembly = Btk_ObjectTypeApi.findByMnemoCode("AssemblyUnits");
if(idvAssembly != null && idvGASTypeMSCH != null) {
        var rop = Gds_ObjTypeGoodsApi.registerByObjectType(idvAssembly);
        Gds_ObjTypeGoodsApi.setidGoodsAndServiceType(rop, idvGASTypeMSCH);
}
commit();
```

Заполнение обозначений контрагентов

Заполняет или восстанавливает обозначения контрагентов для данных GDS. Используется для служебной корректировки справочных данных модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
        var ropList =  sql(`
                select distinct cd.id
from(
        with obezl as (
                        select distinct nch21 as sDesignation from sormovo_obezl
                )
                ,spko as (
                        select distinct nsp as sDesignation from sormovo.sormovo_spko
                )
                ,sprkrep as (
                        select distinct nch as sDesignation from sormovo_sprkrep
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                    )
                    select
                            sp.id as idSrcSpec
                     ,sp.idConstructor
                            ,sp1.id as idCurSpec
                            ,sv.id as idCurVer
                            ,sv.idstate as idCurVerState
                            ,case
                                    when tt.sDesignationDraftOrig != tt.sDesignation
                                            and (coalesce(t2.sDesignation, t3.
↪sDesignation, t4.sDesignation) is not null
                                                    or g2.id is not null)
                                    then case when tt.sCaption like '%  05' or  tt.
↪sCaption like '%  5'
                                            then 'СД'
                                            else case
                                                    when tt.sRoutePath like '67%' or tt.
↪sRoutePath like '68%'
                                                    then 'ТМЦ МСЧ'
                                                    else 'МСЧ'
                                                    end
                                            end
                               when  (tt.sRoutePath like '67%' or tt.sRoutePath like '68%')
                                   and tt.sMaterialCode = '00000000000'
                                 then
                                   'ТМЦ NEW'
                                    else case when tt.sRoutePath like '21%'
                                            or tt.sRoutePath like '24%'
                                            or tt.sRoutePath like '26%'
                                            or tt.sMaterialCode = '00000000000'
                                 or tt.sLocatedPlace like 'П%'
                                            then 'ИНФ'
                                            else case when tt.sCaption like '%  05' or ␣
↪tt.sCaption like '%  5'
                                                    then case when tt.bIsShip = 1
                                                              or (tt.nQty = 1
                                       and tt.sDesignationDraftOrig = tt.sDesignation
                                       and tt.bIsWeight = 1)
                                                            then 'ТМЦ СД'
                                                            else 'Д СД'
                                                            end
                                                    else case
                                                            when tt.sRoutePath like '67%'␣
↪or tt.sRoutePath like '68%'
                                                            then 'ТМЦ'
                                                            else 'Д'
                                                            end
                                                    end
                                            end
                                    end as sPosType
                            ,g1.id as idGds
                            ,g2.id as idUnitProd
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                        ,g3.id as idGdsNew
                        ,tt.*
                from (
                        select
                                t.mashnom as sLocatedPlace
                                ,t.nsp as sDesignationDraftOrig
                                ,regexp_replace(t.nsp, '(.+\-\d{6})(\-)(\d*\-\d*)', '\
↪1.\3') as sDesignationDraft
                                ,t.naim as sCaption
                                ,case
                                when t.nsp like '%3621%' then 1
                                else 0 end as bIsShip
                                ,t.poz as sPosition
                                ,t.nch as sDesignation
                                ,cast(t.kol as numeric) as nQty
                                ,cast(t.ves as numeric) as nMassFull
                                ,nullif(t.marka, '') as sMatMarkProj
                                ,t.marshr as sRoutePath
                                ,t.trud as sTechComplect
                                ,regexp_replace(t.zakaz, '(\d)(\d){1}(\d*)', '\1\3')␣
↪as sPrjVers
                                ,t.pokr as sCoverType
                                ,t.shifr as sMaterialCode
                                ,cast(lpad(t.eiv, 1, '0') as numeric) as bIsWeight
                                ,m.id as idMsrNormItem
                                ,cast(t.norma as numeric) as nNorm
                                ,nullif(t.razmer, '') as sSize
                                ,cast(case when t.razmer != '' and t.razmer SIMILAR␣
↪TO '([0-9]*)' and t.razmer not similar to '([0]*)' and t.razmer is not null
                                        then  substring(lpad(t.razmer, 9, '0'),1,5)␣
↪|| '.' || substring(lpad(t.razmer, 9, '0'),6)
                                        else cast (null as varchar(254))
                                end as numeric) as nNormInSize
                        from sormovo.sormovo_spko t
                        left join sormovo_eizm ei on lpad(t.einv, 3, '0') = ei.ed
                        left join msr_measureitem m on upper(ei.ted) = upper(m.
↪sshortname)
                        --where t.nsp = 'RSD59-362313-04-001'
                ) tt
                left join (
                  select distinct on (regexp_replace(scode, 'СБ', ''))
                    id
                    ,regexp_replace(scode, 'СБ', '') as sCode
                    ,idConstructor
                  from mct_specification
                  ) sp on tt.sDesignationDraft = sp.scode
                left join mct_specification sp1 on tt.sDesignationDraft||'(ИВЦ)' =␣
↪sp1.scode
                left join mct_specificationver sv on (sp1.id = sv.idmctdocument and␣
↪sv.bislast = 1)
                left join obezl t2 on tt.sDesignation = t2.sDesignation
                left join spko t3 on tt.sDesignation = t3.sDesignation
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                    left join sprkrep t4 on tt.sDesignation = t4.sDesignation
                    left join bs_goods g1 on tt.sMaterialCode = g1.sarticle
                    left join bs_goods g2 on tt.sDesignation = g2.sarticle
                    left join bs_goods g3 on (tt.sDesignation = g3.sDesignation and tt.
↪sCaption = g3.sname)
                    order by tt.sDesignationDraftOrig, tt.sLocatedPlace
) t
join gds_contrasdesignation cd on cd.sdesignation = t.sDesignation and cd.idGds != t.
↪idUnitProd and cd.idContras = 99206
join bs_goods g on cd.idGds = g.id
where t.sPosType = 'МСЧ'
                `).batchObjLoad(Gds_ContrasDesignationApi, "id");
        for (rop : ropList){
                Gds_ContrasDesignationApi.delete(rop);
        }
        commit();
```

BPM

Миграция Global Subject

Локальный скрипт модуля **BPM**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
//регистрация глобальных субъектов по субъектам, на которые есть ссылки
sql(`
        select
                psps.idPsSubject as "idPsSubject"
                ,pss.sSystemName as "sSystemName"
                ,max(pss.scaption) as "sCaption"
        from Bpm_PrSubjectPersSetting psps
        left join Bpm_PSSubject pss on psps.idpssubject = pss.id
        where psps.idpssubject is not null
        and psps.idsubject is null
        group by psps.idpssubject,pss.ssystemname
`).foreach(function(r){
        Bpm_SubjectApi.register(r.sSystemName, r.sCaption, null);
});
flush();
//проставляем ссылки на глобальные субъекты в субъекты схем
var svSelect = sql(`
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        select
                ps.id as "id"
                ,ps.ssystemname
                ,s.id as "idSubj"
        from bpm_pssubject ps
        join bpm_subject s using (ssystemname)
`);
var ropList =  svSelect.batchObjLoad(Bpm_PSSubjectApi, "id");
svSelect.foreach(function(r){
        Bpm_PSSubjectApi.setidSubject(Bpm_PSSubjectApi.load(r.id), r.idSubj);
});
//проставляем ссылки на глобальные субъекты в настройки пользователей
var ropavPersSettings = sql(`
        select
                psps.id
        from Bpm_PrSubjectPersSetting psps
        where psps.idpssubject is not null
        and psps.idsubject is null
`).batchObjLoad(Bpm_PrSubjectPersSettingApi, "id");
for (rop : ropavPersSettings) {
        var idvPsSubj = rop.copyAro().idPSSubject();
        var idvGSubj = Bpm_PSSubjectApi.load(idvPsSubj).copyAro().idSubject();
        if (idvGSubj != null) {
                Bpm_PrSubjectPersSettingApi.setidSubject(rop, idvGSubj);
        };
};
commit();
```

Обновление Assigment State

Локальный скрипт модуля **BPM**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvStateDirected = Bpm_AssignmentApi.idStateDirected();
var ropListAll =  sql(`
        select a.id
        from bpm_assignment a
        join bpm_process p on a.idprocess = p.id
        where a.idstatemc = 100
        and p.idstatemc is distinct from 100
        `).batchObjLoad(Bpm_AssignmentApi, "id");
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
for (rop : ropListAll){
        Bpm_AssignmentApi.setidState(rop, idvStateDirected);
}
commit();
```

Обновление Setting Use Msg Servises

Локальный скрипт модуля **BPM**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var ida = sql(`
        select
                bs.id
                ,case when bs.idmsgservice is null
                        then cast(0 as numeric)
                        else cast(1 as numeric)
                        end as busemsgservice
                ,case when bs.idmsgservicesms is null
                        then cast(0 as numeric)
                        else cast(1 as numeric)
                        end as busemsgservicesms
        from bpm_setting bs
`).asList();
for(i:ida) {
        var rop = Bpm_SettingApi.load(i.id);
        Bpm_SettingApi.setbUseMsgService(rop, i.busemsgservice);
        Bpm_SettingApi.setbUseMsgServiceSMS(rop, i.busemsgservicesms);
}
commit();
```

LBR

01.author Group

Локальный скрипт модуля **LBR**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var sClassName = "Lbr_Author"
        var list = [["Group1807", "Зарубежная литература", null], ["Group1902",
↪"Отечественная литература", null], ["Group1806", "Русская литература советского периода
↪", "Group1902"], ["Group1803", "Русская литература 18 века", "Group1902"], ["Group1805
↪", "Русская литература серебряного века (конец 19 - начало 20)", "Group1902"], [
↪"Group1804", "Русская литература золотого века  (19 ВЕК)", "Group1902"], ["Top1**",
↪"1** лучших зарубежных автора", "Group1807"]]
        var api = Btk_ClassApi.getApiBySimpleClassName(sClassName);
        var idvRootGroup = Btk_ClassApi.load(api.idClass()).copyAro().idGroupRoot();
        for (t : list){
                var svCode = "Lbr_" + t.0;
                var idv = Btk_GroupApi.findByMnemoCode(svCode);
                if(idv == null){
                        var rop = Btk_GroupApi.insert();
                        Btk_GroupApi.setsSystemName(rop, svCode);
                        Btk_GroupApi.setsCaption(rop, t.1);
                }
        }
        commit();
        for (t : list){
                var svCode = "Lbr_" + t.0;
                var svParentCode = "Lbr_" + t.2;
                var idv = Btk_GroupApi.findByMnemoCode(svCode);
                var idvParent = Btk_GroupApi.findByMnemoCode(svParentCode);
                if(idv != null){
                        var rop = Btk_GroupApi.load(idv);
                        if(idvParent != null){
                                Btk_GroupApi.setidParentGroup(rop, idvParent);
                        } else {
                                Btk_GroupApi.setidParentGroup(rop, idvRootGroup);
                        }
                }
        }
        commit();
```

02.author

Локальный скрипт модуля **LBR**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var sClassName = "Lbr_Author"
        var list = [["0003", "Лотман", "Юрий", "Михайлович", "Лотман Юрий Михайлович",␣
↪null], ["0007", "Пелевин", "Виктор", "Олегович", "Пелевин В.О.", null], ["0001",
↪"Пушкин", "Александр", "Сергеевич", "Пушкин Александр Сергеевич ", "Group1804"], ["0002
↪", "Лермонтов", "Михаил", "Юрьевич", "Лермонтов Михаил Юрьевич", "Group1804"], ["0004",
↪ "Уайльд", "Оскар", "Фингал О'Флаэрти", "Оскар Уайльд", "Group1807"], ["0005",
↪"Достоевский", "Федор", "Михайлович", "Достоевский Ф.М.", "Group1804"], ["0006",
↪"Толстой", "Лев", "Николаевич", "Толстой Л.Н.", "Group1804"]]
        var api = Btk_ClassApi.getApiBySimpleClassName(sClassName);
        var idvRootGroup = Btk_ClassApi.load(api.idClass()).copyAro().idGroupRoot();
        for (t : list){
                var svCode = t.0;
                var idv = api.findByMnemoCode(svCode);
                if(idv == null){
                        var rop = api.insert();
                        api.setsCode(rop, svCode);
                        api.setsLastName(rop, t.1);
                        api.setsFirstName(rop, t.2);
                        api.setsMiddleName(rop, t.3);
                        api.setsFIO(rop, t.4);
                        var svGroupCode = "Lbr_" + t.5;
                        var idvGroup = Btk_GroupApi.findByMnemoCode(svGroupCode);
                        if(idvGroup != null) {
                                Btk_ObjectGroupApi.register(rop, idvGroup, 0B, 1B)
                        } else {
                                Btk_ObjectGroupApi.register(rop, idvRootGroup, 0B, 1B)
                        }
                }
        }
        commit();
```

03.publisher

Локальный скрипт модуля **LBR**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var sClassName = "Lbr_Publisher"
        var list = [["002", "ООО Издательство Эксмо", null], ["003", "Феникс", null], [
↪"001", "ИЗДАТЕЛЬСТВО АСТ", null], ["004", "Азбука", null], ["005", "Эксмо", null]];
        var api = Btk_ClassApi.getApiBySimpleClassName(sClassName);
        for (t : list){
                var svCode = t.0;
                var idv = api.findByMnemoCode(svCode);
                if(idv == null){
                        var rop = api.insert();
                        api.setsSystemName(rop, svCode);
                        api.setsCaption(rop, t.1);
                        api.setsDescription(rop, t.2);
                }
        }
        commit();
```

04.book

Локальный скрипт модуля **LBR**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var sClassName = "Lbr_Book"
        var list = [["978-5-04-121401-2", "Пушкин Александр Сергеевич: Капитанская дочка
↪", "002", "002", "0001", 2022.0000000000000000000В, 384.0000000000000000000В, "«Повести␣
↪покойного Ивана Петровича Белкина» А.С. Пушкин написал болдинской осенью 1830 года,␣
↪находясь на трехмесячном карантине."], ["978-5-04-121401-3", "Портрет Дориана Грея",
↪"002", "001", "0004", 1994.0000000000000000000В, 400.0000000000000000000В, null], ["978-
↪5-04-119061-3", "Анна Каренина", "002", "005", "0006", null, null, null], ["978-5-389-
↪06256-6", "Война и мир", "002", "004", "0006", null, null, null], ["978-5-04-116677-9",
↪ "Идиот", "002", "001", "0005", null, null, null], ["978-5-17-063868-0", "Бесы", "002",
↪ "001", "0005", null, null, null], ["978-5-699-70277-0", "Воскресенье", "002", "005",
↪"0006", null, null, null], ["978-5-4467-0296-1", "Зигмунд в кафе", "002", "003", "0007
↪", null, null, null], ["978-5-17-053221-6", "Братья Карамазовы", "002", "003", "0005",␣
↪null, null, null], ["978-5-389-10222-4", "Лотман Ю.: Александр Сергеевич Пушкин.␣
↪Биография писателя", "007", "003", "0003", 2015.0000000000000000000В, 286.
↪0000000000000000000В, "Юрий Михайлович Лотман - всемирно известный ученый, филолог и␣
↪культуролог, чьи труды переведены на все основные европейские языки. Его наследие␣
↪чрезвычайно обширно и насчитывает около восьмисот научных и научно-популярных статей и␣
↪книг."]];
        var api = Btk_ClassApi.getApiBySimpleClassName(sClassName);
        //var apiCatalog = Btk_ClassApi.getApiBySimpleClassName("Lbr_Catalog");
        var apiPublisher = Btk_ClassApi.getApiBySimpleClassName("Lbr_Publisher");
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        var apiAuthor = Btk_ClassApi.getApiBySimpleClassName("Lbr_Author");
        for (t : list){
                var svCode = t.0;
                var idv = api.findByMnemoCode(svCode);
                if(idv == null){
                        var rop = api.insert();
                        api.setsISBN(rop, svCode);
                        api.setsCaption(rop, t.1);
                        //var idvCatalog = apiCatalog.findByMnemoCode(t.2);
                        //if(idvCatalog != null){
                        //        api.setidCatalog(rop, idvCatalog);
                        //}
                        var idvPublisher = apiPublisher.findByMnemoCode(t.3);
                        if(idvPublisher != null){
                                api.setidPublisher(rop, idvPublisher);
                        }
                        var idvAuthor = apiAuthor.findByMnemoCode(t.4);
                        if(idvAuthor != null){
                                api.setidAuthor(rop, idvAuthor);
                        }
                        api.setnYear(rop, t.5);
                        api.setnColPage(rop, t.6);
                        if(t.7 != null){
                                api.setsDesc(rop, t.7);
                        }

                }
        }
        commit();
```

Регистрация типов объектов документов библиотеки

Регистрирует типы объектов и вкладки для документов библиотечного модуля. Используется при первичной настройке или миграции объектов LBR.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
sql(`
        select regexp_replace(ssystemname, 'lbr', '') as "sNum"
        from btk_module bm
        where ssystemname like 'lbr__'
`).foreach(function(r){
        Btk_Pkg.setRWSharedUOWEditType();
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        var sClassName = "Lbr" + r.sNum + "_InAct";
        var api = Btk_ClassApi.getApiBySimpleClassName(sClassName);
        var idvClass = api.idClass();
        var idvOT = Btk_ObjectTypeApi.register(sClassName, "Приходная накладная",
↪"Приходная накладная",idvClass,null,1B,null,null,null,null,null,null,null,null,null,
↪null,null,null);
        Btk_StateChangeApi.registerForObjectType(idvOT, "Create", "Executed");
        Btk_StateChangeApi.registerForObjectType(idvOT, "Executed", "Create");
        var apiInOrder = Btk_ClassApi.getApiBySimpleClassName("Lbr" + r.sNum + "_InOrder
↪");
        var idvInOrderClass = apiInOrder.idClass();
        var apiOutOrder = Btk_ClassApi.getApiBySimpleClassName("Lbr" + r.sNum + "_
↪OutOrder");
        var idvOutOrderClass = apiOutOrder.idClass();
        var savTab = ["Btk_AttachItemAvi.List_SimpleAttach", "Bpm_ProcessTreeAvi.
↪TreeForDoc", "Bs_ObjectSignAvi.List_Master", "Act_TransAvi.List_gidDoc"]
        for (svTab : savTab){
                var idvTab = Btk_TabApi.findByMnemoCode(svTab);
                if(idvTab != null){
                        var ropTab = Btk_TabApi.load(idvTab);
                        Btk_TabClassApi.register(ropTab, idvClass);
                        Btk_TabClassApi.register(ropTab, idvInOrderClass);
                        Btk_TabClassApi.register(ropTab, idvOutOrderClass);
                }
        }
        Btk_TabApi.register(null, "Характеристики", "gtk-Lbr" + r.sNum + "_OutOrderAvi",
↪"Card_ObjectAttr", apiOutOrder.idClass(), null,  null,  null,  null,  null,  null);
        commit();
})
```

Создание приходных актов библиотеки

Создает приходные акты и позиции документов для библиотечного модуля на основе заданного набора данных.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvDepOwner = Bs_DepOwnerApi.findByMnemoCode("0000");
var idvPerson1 = Bs_PersonApi.findByMnemoCode("0200");
var idvPerson2 = Bs_PersonApi.findByMnemoCode("0201");
sql(`
        select regexp_replace(ssystemname, 'lbr', '') as "sNum"
        from btk_module bm
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        where ssystemname like 'lbr__'
`).foreach(function(r){
        var list = [["1", 567.00000000000000000B, [["1", "978-5-04-121401-2", 3.
↪00000000000000000B, 189.00000000000000000B, 567.00000000000000000B]]], ["2", null,␣
↪[["1", "978-5-389-10222-4", 5.00000000000000000B, 450.00000000000000000B, 2250.
↪00000000000000000B], ["2", "978-5-04-121401-2", 5.00000000000000000B, 358.
↪00000000000000000B, 1790.00000000000000000B], ["3", "978-5-04-121401-3", 10.
↪00000000000000000B, 635.00000000000000000B, 6350.00000000000000000B]]], ["3", null,␣
↪[[null, null, null]]], ["5", 1000000.00000000000000000B, [["2", "978-5-04-119061-3",␣
↪100.00000000000000000B, 100.00000000000000000B, 10000.00000000000000000B], ["3",
↪"978-5-389-06256-6", 100.00000000000000000B, 100.00000000000000000B, 10000.
↪00000000000000000B], ["4", "978-5-04-121401-2", 100.00000000000000000B, 100.
↪00000000000000000B, 10000.00000000000000000B], ["1", "978-5-04-121401-3", 100.
↪00000000000000000B, 100.00000000000000000B, 10000.00000000000000000B], ["5", "978-5-
↪17-063868-0", 100.00000000000000000B, 100.00000000000000000B, 10000.
↪00000000000000000B]]]];
        var apiDoc = Btk_ClassApi.getApiBySimpleClassName("Lbr" + r.sNum + "_InAct");
        var apiDet = Btk_ClassApi.getApiBySimpleClassName("Lbr" + r.sNum + "_InActDet");
        var apiBook = Btk_ClassApi.getApiBySimpleClassName("Lbr" + r.sNum + "_Book");
        for (t : list){
                var svCode = t.0;
                var idv = apiDoc.findByMnemoCode(svCode);
                if(idv == null){
                        var rop = apiDoc.insert();
                        apiDoc.setsNumDoc(rop, svCode);
                        apiDoc.setsNumDocBMs_dz(rop, 1B);
                        apiDoc.setnSum(rop, t.1);
                        apiDoc.setidLibrarian(rop, idvPerson1);
                        apiDoc.setidDepOwner(rop, idvDepOwner);
                        for (det : t.2) {
                                var sDetCode = det.0;
                                if(sDetCode != null){
                                        var ropDet = apiDet.insertByParent(rop);
                                        apiDet.setsNumber(ropDet, sDetCode);
                                        apiDet.setsNumberBMs_dz(ropDet, 1B);
                                        var idvBook = apiBook.findByMnemoCode(det.1);
                                        if(idvBook != null){
                                                apiDet.setidBook(ropDet, idvBook);
                                        }
                                        apiDet.setnQty(ropDet, det.2);
                                        apiDet.setnPrice(ropDet, det.3);
                                        apiDet.setnSum(ropDet, det.4);

                                }
                        }

                }
        }
        commit();
}) ;
```

Создание заявок на поступление книг

Создает документы заявок на поступление и их позиции для библиотечного модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvDepOwner = Bs_DepOwnerApi.findByMnemoCode("0000");
var idvPerson1 = Bs_PersonApi.findByMnemoCode("0200");
var idvPerson2 = Bs_PersonApi.findByMnemoCode("0201");
sql(`
        select regexp_replace(ssystemname, 'lbr', '') as "sNum"
        from btk_module bm
        where ssystemname like 'lbr__'
 `).foreach(function(r){
        var list = [["1", [["1", "978-5-04-121401-2"]]]];
        var apiDoc = Btk_ClassApi.getApiBySimpleClassName("Lbr" + r.sNum + "_InOrder");
        var apiDet = Btk_ClassApi.getApiBySimpleClassName("Lbr" + r.sNum + "_InOrderDet
↪");
        var apiBook = Btk_ClassApi.getApiBySimpleClassName("Lbr" + r.sNum + "_Book");
        for (t : list){
                var svCode = t.0;
                var idv = apiDoc.findByMnemoCode(svCode);
                if(idv == null){
                        var rop = apiDoc.insert();
                        apiDoc.setsNumDoc(rop, svCode);
                        apiDoc.setsNumDocBMs_dz(rop, 1B);
                        apiDoc.setidLibrarian(rop, idvPerson1);
                        apiDoc.setidPerson(rop, idvPerson2);
                        apiDoc.setidDepOwner(rop, idvDepOwner);
                        for (det : t.1) {
                                var sDetCode = det.0;
                                if(sDetCode != null){
                                        var ropDet = apiDet.insertByParent(rop);
                                        apiDet.setsNumber(ropDet, sDetCode);
                                        apiDet.setsNumberBMs_dz(ropDet, 1B);
                                        var idvBook = apiBook.findByMnemoCode(det.1);
                                        if(idvBook != null){
                                                apiDet.setidBook(ropDet, idvBook);
                                        }

                                }
                        }

                }
        }
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        commit();
}) ;
```

Создание заявок на выдачу книг

Создает документы заявок на выдачу и их позиции для библиотечного модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvDepOwner = Bs_DepOwnerApi.findByMnemoCode("0000");
var idvPerson1 = Bs_PersonApi.findByMnemoCode("0200");
var idvPerson2 = Bs_PersonApi.findByMnemoCode("0201");
sql(`
        select regexp_replace(ssystemname, 'lbr', '') as "sNum"
        from btk_module bm
        where ssystemname like 'lbr__'
 `).foreach(function(r){
        var list = [["1", ["1", "978-5-04-121401-2"]]], ["2", [["1", "978-5-04-121401-2
↪"]]], ["6", [["1", "978-5-389-06256-6"]]], ["5", [["1", "978-5-04-121401-2"], ["2",
↪"978-5-389-06256-6"], ["3", "978-5-04-116677-9"], ["4", "978-5-17-063868-0"]]], ["7",␣
↪[[null, null]]], ["4", [[null, null]]]];
        var apiDoc = Btk_ClassApi.getApiBySimpleClassName("Lbr" + r.sNum + "_OutOrder");
        var apiDet = Btk_ClassApi.getApiBySimpleClassName("Lbr" + r.sNum + "_OutOrderDet
↪");
        var apiBook = Btk_ClassApi.getApiBySimpleClassName("Lbr" + r.sNum + "_Book");
        for (t : list){
                var svCode = t.0;
                var idv = apiDoc.findByMnemoCode(svCode);
                if(idv == null){
                        var rop = apiDoc.insert();
                        apiDoc.setsNumDoc(rop, svCode);
                        apiDoc.setsNumDocBMs_dz(rop, 1B);
                        apiDoc.setidLibrarian(rop, idvPerson1);
                        apiDoc.setidPerson(rop, idvPerson2);
                        apiDoc.setidDepOwner(rop, idvDepOwner);
                        for (det : t.1) {
                                var sDetCode = det.0;
                                if(sDetCode != null){
                                        var ropDet = apiDet.insertByParent(rop);
                                        apiDet.setsNumber(ropDet, sDetCode);
                                        apiDet.setsNumberBMs_dz(ropDet, 1B);
                                        var idvBook = apiBook.findByMnemoCode(det.1);
                                        if(idvBook != null){
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                                                apiDet.setidBook(ropDet, idvBook);
                                        }

                                }
                        }

                }
        }
        commit();
}) ;
```

Создание авторов для библиотечных модулей

Создает авторов для всех найденных библиотечных модулей `lbr__` на основе заданного набора данных.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
sql(`
        select regexp_replace(ssystemname, 'lbr', '') as "sNum"
        from btk_module bm
        where ssystemname like 'lbr__'
 `).foreach(function(r){
        var sClassName = "Lbr" + r.sNum + "_Author"
        var list = [["0003", "Лотман", "Юрий", "Михайлович", "Лотман Юрий Михайлович",␣
↪null], ["0007", "Пелевин", "Виктор", "Олегович", "Пелевин В.О.", null], ["0001",
↪"Пушкин", "Александр", "Сергеевич", "Пушкин Александр Сергеевич ", "Group1804"], ["0002
↪", "Лермонтов", "Михаил", "Юрьевич", "Лермонтов Михаил Юрьевич", "Group1804"], ["0004",
↪ "Уайльд", "Оскар", "Фингал О'Флаэрти", "Оскар Уайльд", "Group1807"], ["0005",
↪"Достоевский", "Федор", "Михайлович", "Достоевский Ф.М.", "Group1804"], ["0006",
↪"Толстой", "Лев", "Николаевич", "Толстой Л.Н.", "Group1804"]]
        var api = Btk_ClassApi.getApiBySimpleClassName(sClassName);
        var idvRootGroup = Btk_ClassApi.load(api.idClass()).copyAro().idGroupRoot();
        for (t : list){
                var svCode = t.0;
                var idv = api.findByMnemoCode(svCode);
                if(idv == null){
                        var rop = api.insert();
                        api.setsCodeBMs_dz(rop, 1B);
                        api.setsCode(rop, svCode);
                        api.setsLastName(rop, t.1);
                        api.setsFirstName(rop, t.2);
                        api.setsMiddleName(rop, t.3);
                        api.setsFIO(rop, t.4);
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                        var svGroupCode = "Lbr" + r.sNum + "_" + t.5;
                        var idvGroup = Btk_GroupApi.findByMnemoCode(svGroupCode);
                        if(idvGroup != null) {
                                Btk_ObjectGroupApi.register(rop, idvGroup, 0B, 1B)
                        } else {
                                Btk_ObjectGroupApi.register(rop, idvRootGroup, 0B, 1B)
                        }
                }
        }
        commit();
}) ;
```

Создание групп авторов для библиотечных модулей

Создает группы авторов для всех найденных библиотечных модулей `lbr__` и выстраивает их иерархию.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
sql(`
        select regexp_replace(ssystemname, 'lbr', '') as "sNum"
        from btk_module bm
        where ssystemname like 'lbr__'
 `).foreach(function(r){
        var sClassName = "Lbr" + r.sNum + "_Author"
        var list = [["Group1807", "Зарубежная литература", null], ["Group1902",
↪"Отечественная литература", null], ["Group1806", "Русская литература советского периода
↪", "Group1902"], ["Group1803", "Русская литература 18 века", "Group1902"], ["Group1805
↪", "Русская литература серебряного века (конец 19 - начало 20)", "Group1902"], [
↪"Group1804", "Русская литература золотого века  (19 ВЕК)", "Group1902"], ["Top1**",
↪"1** лучших зарубежных автора", "Group1807"]]
        var api = Btk_ClassApi.getApiBySimpleClassName(sClassName);
        var idvRootGroup = Btk_ClassApi.load(api.idClass()).copyAro().idGroupRoot();
        for (t : list){
                var svCode = "Lbr" + r.sNum + "_" + t.0;
                var idv = Btk_GroupApi.findByMnemoCode(svCode);
                if(idv == null){
                        var rop = Btk_GroupApi.insert();
                        Btk_GroupApi.setsSystemName(rop, svCode);
                        Btk_GroupApi.setsCaption(rop, t.1);
                }
        }
        commit();
        for (t : list){
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                var svCode = "Lbr" + r.sNum + "_" + t.0;
                var svParentCode = "Lbr" + r.sNum + "_" + t.2;
                var idv = Btk_GroupApi.findByMnemoCode(svCode);
                var idvParent = Btk_GroupApi.findByMnemoCode(svParentCode);
                if(idv != null){
                        var rop = Btk_GroupApi.load(idv);
                        if(idvParent != null){
                                Btk_GroupApi.setidParentGroup(rop, idvParent);
                        } else {
                                Btk_GroupApi.setidParentGroup(rop, idvRootGroup);
                        }
                }
        }
        commit();
}) ;
```

Создание книг для библиотечных модулей

Создает записи книг для всех найденных библиотечных модулей `lbr__` и связывает их с каталогами, издателями и авторами.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
sql(`
        select regexp_replace(ssystemname, 'lbr', '') as "sNum"
        from btk_module bm
        where ssystemname like 'lbr__'
 `).foreach(function(r){
        var sClassName = "Lbr" + r.sNum + "_Book"
        var list = [["978-5-04-121401-2", "Пушкин Александр Сергеевич: Капитанская дочка
↪", "002", "002", "0001", 2022.0000000000000000000B, 384.0000000000000000000B, "«Повести␣
↪покойного Ивана Петровича Белкина» А.С. Пушкин написал болдинской осенью 1830 года,␣
↪находясь на трехмесячном карантине. И, по собственному признанию, «он никогда еще так␣
↪не писал» - «просто, коротко и ясно» и так плодотворно. По мнению самого поэта, нужно␣
↪писать прозу, которая одновременно «требует мыслей и мыслей - без них блестящие␣
↪выражения ни к чему не служат». Главный вопрос всех размышлений в «Повестях Белкина»␣
↪был вопрос о счастье и судьбе. Абсолютно новаторский цикл позволил создать широкую␣
↪картину русской жизни с высочайшей степенью человечности.Повесть «Дубровский» была␣
↪задумана автором как роман эпохи пугачёвщины с героем-дворянином, перешедшим на␣
↪сторону бунтовщиков. Работая над «Дубровским», Александр Пушкин задумал «Капитанскую␣
↪дочку», а писал, когда работал над «Историей Пугачева». И Николай Гоголь назвал␣
↪«Капитанскую дочку» «решительно лучшим русским произведеньем в повествовательном роде»,
↪ в котором «выступили истинно русские характеры: простой комендант крепости, капитанша,
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
↪ поручик; сама крепость с единственной пушкой, бестолковщина времени и простое величие␣
↪простых людей».В «Пиковой даме» А. Пушкин соединил признаки модных в то время␣
↪повествований - светской и фантастической повестей и добился широкого признания.␣
↪«Верхом искусства фантастического» позже назвал «Пиковую даму» Ф.М. Достоевский.
↪АннотацияТворчество Александра Сергеевича Пушкина знакомо каждому еще со школы. И␣
↪любой читатель находит в нем именно то, что ему близко: одни восторженно относятся к␣
↪пушкинской лирике, другие учат наизусть Евгения Онегина, третьи любят простоту и␣
↪ясность его прозы. Уникальный язык пушкинской прозы стал той плодородной почвой, на␣
↪которой поднялся великий русский реализм XIX века.В книгу вошли повести Капитанская␣
↪дочка и Пиковая дама, роман Дубровский, Повести покойного Ивана Петровича Белкина,"], [
↪"978-5-04-121401-3", "Портрет Дориана Грея", "002", "001", "0004", 1994.
↪0000000000000000000В, 400.0000000000000000000В, null], ["978-5-04-119061-3", "Анна␣
↪Каренина", "002", "005", "0006", null, null, null], ["978-5-389-06256-6", "Война и мир
↪", "002", "004", "0006", null, null, null], ["978-5-04-116677-9", "Идиот", "002", "001
↪", "0005", null, null, null], ["978-5-17-063868-0", "Бесы", "002", "001", "0005", null,
↪ null, null], ["978-5-699-70277-0", "Воскресенье", "002", "005", "0006", null, null,␣
↪null], ["978-5-4467-0296-1", "Зигмунд в кафе", "002", "003", "0007", null, null, null],
↪ ["978-5-17-053221-6", "Братья Карамазовы", "002", "003", "0005", null, null, null], [
↪"978-5-389-10222-4", "Лотман Ю.: Александр Сергеевич Пушкин. Биография писателя", "007
↪", "003", "0003", 2015.0000000000000000000В, 286.0000000000000000000В, "Юрий Михайлович␣
↪Лотман - всемирно известный ученый, филолог и культуролог, чьи труды переведены на все␣
↪основные европейские языки. Его наследие чрезвычайно обширно и насчитывает около␣
↪восьмисот научных и научно-популярных статей и книг. Особое место в исследованиях Ю. М.
↪ Лотмана занимают труды о Пушкине, за которые в 1993 г. ученый был награжден␣
↪Пушкинской премией РАН. В фундаментальной монографии Александр Сергеевич Пушкин:␣
↪Биография писателя, считающейся классикой пушкинистики, Лотману-биографу удается␣
↪показать не только этапы личностного и творческого роста поэта, но и вписать события␣
↪его жизни в русскую и европейскую историю, а также предложить концепцию␣
↪жизнестроительства - осознанного выстраивания Пушкиным своей биографии как␣
↪оригинального и законченного художественного произведения."]];
        var api = Btk_ClassApi.getApiBySimpleClassName(sClassName);
        var apiCatalog = Btk_ClassApi.getApiBySimpleClassName("Lbr" + r.sNum + "_Catalog
↪");
        var apiPublisher = Btk_ClassApi.getApiBySimpleClassName("Lbr" + r.sNum + "_
↪Publisher");
        var apiAuthor = Btk_ClassApi.getApiBySimpleClassName("Lbr" + r.sNum + "_Author");
        for (t : list){
                var svCode = t.0;
                var idv = api.findByMnemoCode(svCode);
                if(idv == null){
                        var rop = api.insert();
                        api.setsISBN(rop, svCode);
                        api.setsCaption(rop, t.1);
                        var idvCatalog = apiCatalog.findByMnemoCode(t.2);
                        if(idvCatalog != null){
                                api.setidCatalog(rop, idvCatalog);
                        }
                        var idvPublisher = apiPublisher.findByMnemoCode(t.3);
                        if(idvPublisher != null){
                                api.setidPublisher(rop, idvPublisher);
                        }
                        var idvAuthor = apiAuthor.findByMnemoCode(t.4);
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
                        if(idvAuthor != null){
                                api.setidAuthor(rop, idvAuthor);
                        }
                        api.setnYear(rop, t.5);
                        api.setnColPage(rop, t.6);
                        if(t.7 != null){
                                api.setsDesc(rop, t.7);
                        }

                }
        }
        commit();
}) ;
```

Создание каталогов для библиотечных модулей

Создает каталоги для всех найденных библиотечных модулей `lbr__` и выстраивает их иерархию.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
sql(`
        select regexp_replace(ssystemname, 'lbr', '') as "sNum"
        from btk_module bm
        where ssystemname like 'lbr__'
 `).foreach(function(r){
        var sClassName = "Lbr" + r.sNum + "_Catalog"
        var list = [["002", "Художественная литература", "001", 0.0000000000000000000B], [
↪"003", "Проза", "001", 0.0000000000000000000B], ["004", "Классическая русская литература
↪", "001", 0.0000000000000000000B], ["005", "Публицистика", "001", 0.
↪0000000000000000000B], ["006", "Жизнь знаменитых людей", "005", 0.0000000000000000000B],␣
↪["007", "Деятели культуры и искусства", "006", 0.0000000000000000000B], ["001", "Все␣
↪категории", null, 1.0000000000000000000B]];
        var api = Btk_ClassApi.getApiBySimpleClassName(sClassName);
        for (t : list){
                var svCode = t.0;
                var idv = api.findByMnemoCode(svCode);
                if(idv == null){
                        var rop = api.insert();
                        api.setsCode(rop, svCode);
                        api.setsCaption(rop, t.1);
                        api.setnImage(rop, t.3);
                }
        }
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        commit();
        for (t : list){
                var svCode = t.0;
                var svParentCode = t.2;
                var idv = api.findByMnemoCode(svCode);
                var idvParent = api.findByMnemoCode(svParentCode);
                if(idv != null && idvParent != null){
                        var rop = api.load(idv);
                        api.setidParent(rop, idvParent);
                }
        }
        commit();
}) ;
```

Создание издателей для библиотечных модулей

Создает записи издателей для всех найденных библиотечных модулей `lbr__`.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
sql(`
        select regexp_replace(ssystemname, 'lbr', '') as "sNum"
        from btk_module bm
        where ssystemname like 'lbr__'
 `).foreach(function(r){
        var sClassName = "Lbr" + r.sNum + "_Publisher"
        var list = [["002", "ООО Издательство Эксмо", null], ["003", "Феникс", null], [
↪"001", "ИЗДАТЕЛЬСТВО АСТ", null], ["004", "Азбука", null], ["005", "Эксмо", null]];
        var api = Btk_ClassApi.getApiBySimpleClassName(sClassName);
        var idvRootGroup = Btk_ClassApi.load(api.idClass()).copyAro().idGroupRoot();
        for (t : list){
                var svCode = t.0;
                var idv = api.findByMnemoCode(svCode);
                if(idv == null){
                        var rop = api.insert();
                        api.setsSystemName(rop, svCode);
                        api.setsCaption(rop, t.1);
                        api.setsDescription(rop, t.2);
                        Btk_ObjectGroupApi.register(rop, idvRootGroup, 0B, 1B)
                }
        }
        commit();
}) ;
```

MRT

Исправление Change Report Sc

Локальный скрипт модуля **MRT**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvIncorrectSC = Btk_SubClassApi.findByMnemoCode("ChangeReport_WorcCard");
if(idvIncorrectSC != null){
        var rop = Btk_SubClassApi.load(idvIncorrectSC);
        Btk_SubClassApi.setsCode(rop, "ChangeReport_WorkCard");
        commit();
}
```

WF

Восстановление Doc Ver Class Collections

Локальный скрипт модуля **WF**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
Btk_Pkg.setRWSharedUOWEditType();
var idvNewAttrPrDoc = Btk_AttributeApi.findByMnemoCode(Bpm_PrDocApi.idClass(),
↪"gidRefDocVer");
sql(`
        select cc.id
        from Btk_ClassCollection cc
        left join Btk_Class bc on cc.idbtkclass = bc.id
        left join Btk_Class rc on cc.idrefclass = rc.id
        left join Btk_Attribute a on cc.idrefattr = a.id
        where bc.sname = 'Wf_DocVer'
        and rc.sname = 'Bpm_PrDoc'
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
        and a.ssystemname = 'gidRefDocument'
 `).foreach(function(r){
        Btk_ClassCollectionApi.setidRefAttr(Btk_ClassCollectionApi.load(r.id),␣
↪idvNewAttrPrDoc);
}) ;
var idvNewAttrAssign = Btk_AttributeApi.findByMnemoCode(Bpm_AssignmentDocApi.idClass(),
↪"gidRefDocVer");
sql(`
        select cc.id
        from Btk_ClassCollection cc
        left join Btk_Class bc on cc.idbtkclass = bc.id
        left join Btk_Class rc on cc.idrefclass = rc.id
        left join Btk_Attribute a on cc.idrefattr = a.id
        where bc.sname = 'Wf_DocVer'
        and rc.sname = 'Bpm_AssignmentDoc'
        and a.ssystemname = 'gidRefDocument'
 `).foreach(function(r){
        Btk_ClassCollectionApi.setidRefAttr(Btk_ClassCollectionApi.load(r.id),␣
↪idvNewAttrAssign);
}) ;
commit();
```

Обновление Pr Doc Gid Ver

Локальный скрипт модуля **WF**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
sql(`
        select t.id, cast(substring(t.gidrefdocument, '\d*$') as bigint) as "idDoc"
        from bpm_prdoc t
        left join Btk_Class c on cast(substring(t.gidrefdocument, '^\d*') as bigint) = c.
↪id
        left join wf_docVer t1 on t.gidrefdocver = t1.gid
        where t.gidrefdocver is not null
        and t1.gid is null
        and c.sname = 'Wf_Doc'
`).foreach(function(r){
        var idvLastVer = Wf_DocApi.getLastVersion(r.idDoc, null);
        var gidvLastVer = Wf_DocVerApi.getGid(idvLastVer);
        Bpm_PrDocApi.setgidRefDocVer(Bpm_PrDocApi.load(r.id), gidvLastVer);
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
}) ;
commit();
```

BS

Обновление Prj Ver Obj Type And State

Локальный скрипт модуля **BS**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvDefObjectType = Btk_ObjectTypeApi.getDefaultObjType(Bs_PrjVerApi.idClass(), null);
var idvStateOpen = Bs_PrjVerApi.idStateOpen();
var idvStateArchive = Bs_PrjVerApi.idStateArchive();
var select = sql(`
        select
                id
                ,t.bNotActive as "bNotActive"
        from Bs_PrjVer t
        where t.idObjectType is null
        or t.idState is null
        `);
select.batchObjLoad(Bs_PrjVerApi, "id");
select.foreach(function(r){
        var rop = Bs_PrjVerApi.load(r.id);
        Bs_PrjVerApi.setidObjectType(rop, idvDefObjectType);
        if(r.bNotActive == 1B){
                Bs_PrjVerApi.setidState(rop, idvStateArchive);
        } else {
                Bs_PrjVerApi.setidState(rop, idvStateOpen);
        }
});
commit();
```

BTK

Обновление Tech Proc OT Sub Class

Локальный скрипт модуля **BTK**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvScShipYard = Mct_TechProcApi.idScShipYard();
var idvScUnitProduct = Mct_TechProcApi.idScUnitProduct();
Btk_Pkg.setRWSharedUOWEditType();

Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.load(Mct_TechProcApi.idotTechProcSY()),
↪ idvScShipYard);
Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.load(Mct_TechProcApi.idotTechProcUP()),
↪ idvScUnitProduct);
Btk_ObjectTypeApi.setidSubClass(Btk_ObjectTypeApi.load(Mct_TechProcApi.
↪idotTechProcOneTime()), idvScUnitProduct);
commit();
```

Обновление WSTI Tab

Локальный скрипт модуля **BTK**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvTabOld = sql(`
        select id from btk_tab
        where ssystemname = 'Mct_WorkStructureTIAvi.List_idWorkStructure'
        `).asSingle().id;
var idvTabNew = sql(`
        select id from btk_tab
        where ssystemname = 'Mct_ObjectTechInspectTypeAvi.List_gidWorkStructure'
        `).asSingle().id;
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
if(idvTabOld != null && idvTabNew != null){
        Btk_Pkg.setRWSharedUOWEditType();
        sql(`
                select id from btk_objecttypetab
                where idtab = ` + idvTabOld).foreach(function(r){
                        var rop = Btk_ObjectTypeTabApi.load(r.id);
                        Btk_ObjectTypeTabApi.setidTab(rop, idvTabNew);
                });
        sql(`
                select id from Mct_TypePosApplicTab
                where idtab = ` + idvTabOld).foreach(function(r){
                        var rop = Mct_TypePosApplicTabApi.load(r.id);
                        Mct_TypePosApplicTabApi.setidTab(rop, idvTabNew);
                });
        var ropTabOld = Btk_TabApi.load(idvTabOld);
        Btk_TabApi.delete(ropTabOld);
        commit();
};
```

MES

Обновление Route List OT

Локальный скрипт модуля **MES**. Используется для служебной настройки, миграции, очистки или восстановления данных, связанных с объектами модуля.

Место применения: `Сервис > Инструменты > Выполнить JEXL-скрипт`

Внимание

Скрипт привязан к объектам конкретного модуля и требует проверки на целевой базе. Перед запуском проверьте идентификаторы, SQL-запросы, API-классы и условия обработки.

Тип: JEXL-скрипт

```
var idvClass = Mes_RouteListApi.idClass();
var idvDefOT = Btk_ObjectTypeApi.getDefaultObjType(idvClass, null);
var ropav = sql(`
        select
                rl.id
        from Mes_RouteList rl
        where rl.idObjectType is null
`).batchObjLoad(Mes_RouteListApi, "id");
for (rop : ropav) {
        Mes_RouteListApi.setidObjectType(rop, idvDefOT);
};
commit();
```