# BPMN Converter - StormBPMN → Camunda

Описание алгоритма автоматической конвертации BPMN схем из формата **StormBPMN** в формат **Camunda**.

## Описание

Данный документ описывает алгоритм автоматического комплексного преобразование BPMN диаграмм согласно возможностям Camunda Platform 7.

## Выполняемые преобразования:

1. Добавить Camunda namespaces в корневой элемент - `xmlns:camunda`, `xmlns:xsi`.
2. Добавить `camunda:historyTimeToLive="1"` в процесс.
3. Обновить Camunda-специфичные атрибуты процесса - `isExecutable="true"`,
4. Удалить элементы:
   * `<bpmn:collaboration>` со всем содержимым
   * `<bpmn:intermediateCatchEvent>` (все типы)
   * `<bpmn:intermediateThrowEvent>` (все типы)
   * `<bpmn:messageEventDefinition>`
   * `<bpmn:timerEventDefinition>`
   * Связанные `<bpmn:sequenceFlow>`
   * Диаграммные элементы (`<bpmndi:BPMNShape>`, `<bpmndi:BPMNEdge>`)
5. Перенаправить потоки вокруг удаленных элементов.
6. Преобразовать все типы задач (`userTask`, `manualTask`, `callActivity` и др.) в тип `serviceTask`,
7. Добавить ко всем задачам атрибуты: `camunda:type="external"`, `camunda:topic="bitrix\_create\_task"`.
8. Добавить в `<bpmn:conditionExpression>` условные выражения `${result == "ok"} ` к потокам с названиями "да"/"нет".
9. Встроить ответственных в serviceTask как `<camunda:properties>` из отдельного JSON-файла
10. Устранить конфликты `default flow` в эксклюзивных шлюзах
11. Очиcить диаграммные элементы для удаленных объектов

### Логика перенаправления потоков

Когда удаляется промежуточное событие:

1. Находятся входящие потоки (`targetRef` = ID события)
2. Находятся исходящие потоки (`sourceRef` = ID события)
3. Входящие потоки перенаправляются\*\* на цели исходящих потоков
4. Все связанные потоки удаляются

Пример:

Task\_A → IntermediateEvent → Task\_B

Становится:

Task\_A → Task\_B

### Условные выражения

Для потоков с названиями "да" или "нет" добавляются условия:

<!-- Для name="да" -->

<bpmn:conditionExpression xsi:type="bpmn:tFormalExpression**">${result == "ok"}**</bpmn:conditionExpression>

<!-- Для name="нет" -->

<bpmn:conditionExpression xsi:type="bpmn:tFormalExpression**">${result != "ok"}**</bpmn:conditionExpression>

### Исправление конфликтов default flow

В BPMN диаграммах эксклюзивные шлюзы (`exclusiveGateway`) могут иметь `default flow` - поток по умолчанию, который выполняется, если не выполнилось ни одно условие других исходящих потоков.

Проблема:

По стандарту BPMN, `default flow` не должен иметь условие. Это приводит к ошибке:

ENGINE-09005 Exclusive Gateway 'Gateway\_1' has outgoing sequence flow 'Flow\_1'

which is the default flow but has a condition too.

Решение:

Конвертер автоматически убирает атрибут `default` у шлюзов, если соответствующий поток имеет условие.

Было (неправильно):

<bpmn:exclusiveGateway id="Gateway\_1" default="Flow\_YES">

<bpmn:outgoing>Flow\_YES</bpmn:outgoing>

<bpmn:outgoing>Flow\_NO</bpmn:outgoing>

</bpmn:exclusiveGateway>

<bpmn:sequenceFlow id="Flow\_YES" sourceRef="Gateway\_1" targetRef="Task\_1">

<bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">${result == "ok"}</bpmn:conditionExpression>

</bpmn:sequenceFlow>

Стало (правильно):

<bpmn:exclusiveGateway id="Gateway\_1">

<bpmn:outgoing>Flow\_YES</bpmn:outgoing>

<bpmn:outgoing>Flow\_NO</bpmn:outgoing>

</bpmn:exclusiveGateway>

<bpmn:sequenceFlow id="Flow\_YES" sourceRef="Gateway\_1" targetRef="Task\_1">

<bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">${result == "ok"}</bpmn:conditionExpression>

</bpmn:sequenceFlow>

## Что НЕ изменяется:

* ID процессов (остаются оригинальными)
* exporterVersion (остается из исходной схемы)
* default атрибуты в exclusiveGateway (сохраняются)
* endEvent элементы (не удаляются)
* Названия процессов (берутся из исходной схемы)

## Получение списка ответственных

Пример данных:

* "assigneeEdgeId": 15700296,
* "assigneeName": "Рук.отд. арх.сопр. и анализа проектов",
* "assigneeId": 15298311,
* "elementId": "Activity\_1597r5e",
* "elementName": "Поставить задачу",
* "assigneeType": "HUMAN",
* "duration": 900

Пример встраивания:

```xml

<bpmn:serviceTask id="Activity\_14qyrmj" name="Отправить сообщение" camunda:type="external" camunda:topic="bitrix\_create\_task">

<bpmn:extensionElements>

<camunda:properties>

<**camunda:property name="assigneeName" value="Рук.отд. инж.коммуникаций"** />

<**camunda:property name="assigneeId" value="15299256"** />

</camunda:properties>

</bpmn:extensionElements>

<bpmn:incoming>Flow\_1</bpmn:incoming>

<bpmn:outgoing>Flow\_2</bpmn:outgoing>

</b

```

Логика встраивания:

1. Анализируется JSON файл с ответственными
2. Для каждого ответственного находится соответствующий serviceTask по `elementId`
3. Создается (или дополняется) секция `<bpmn:extensionElements>`
4. Добавляется `<camunda:properties>` с двумя свойствами:

* `assigneeName` - имя ответственного
* `assigneeId` - ID ответственного

Примечания:

* Если для элемента назначено несколько ответственных, встраивается только первый
* Если JSON файл не найден, конвертация продолжается без ответственных