

# Пакеты

Пакет-сборка

Версия 8.0



Эта документация предоставляется с ограничениями на использование и защищена законами об интеллектуальной собственности. За исключением случаев, прямо разрешенных в вашем лицензионном соглашении или разрешенных законом, вы не можете использовать, копировать, воспроизводить, переводить, транслировать, изменять, лицензировать, передавать, распространять, демонстрировать, выполнять, публиковать или отображать любую часть в любой форме или посредством любые значения. Обратный инжиниринг, дизассемблирование или декомпиляция этой документации, если это не требуется по закону для взаимодействия, запрещены.

Информация, содержащаяся в данном документе, может быть изменена без предварительного уведомления и не может гарантировать отсутствие ошибок. Если вы обнаружите какие-либо ошибки, сообщите нам о них в письменной форме.

# Содержание

<b>Пакет-сборка</b>	<b>4</b>
Структура и зависимости пакета-сборки	4
Операции с пакетом-сборкой	6
Конвертация пакетов	8
Импорт пакета-сборки	13

# Пакет-сборка



Сложный

Возможность использования пакета-сборки доступна для приложений Creatio версии 7.18.3 и выше.

**Пакет-сборка** — пакет, исходный и автогенерируемый код которого компилируется в отдельную сборку.

**Назначение** пакета-сборки:

- Увеличение скорости разработки за счет снижения времени компиляции, поскольку компилируется только пакет-сборка.
- Увеличение скорости поставки функциональности за счет отсутствия необходимости компиляции, поскольку пакет-сборка содержит в себе предварительно скомпилированную часть.

**Особенности** пакета-сборки:

- Не поддерживается доступ к объектам по протоколу OData версии 3. Для доступа к объектам в пакете-сборке используется протокол OData версии 4. Протокол OData описан в статье [OData](#).
- Не допускается реализация бизнес-логики объектов через событийные подпроцессы дизайнера объекта. Для работы с событийной моделью необходимо использовать `EntityEventListener`. Набор событий `EntityEventListener` не полностью соответствует событийной модели, доступной через Creatio IDE. Событийный слой `Entity` описан в статье [Бизнес-логика объектов](#).
- В C#-коде пакетов не допускается использование типов `Entity` (например, приведение к этим типам), создание экземпляров этих типов через оператор `new` или при помощи рефлексии. Не зависимо от того, где создана эта `Entity`.
- Новые колонки и методы объектов `Entity`, которые размещены в пакете-сборке, не доступны через базовый класс `Entity` (например, `ColumnValues.GetByName(nameValues)`).
- Типы текущего пакета-сборки не доступны в конфигурации и доступны для других пакетов-сборок, которые зависят от текущего.

## Структура и зависимости пакета-сборки

Структура пакета-сборки в файловой системе не отличается от структуры простого пакета. Основное **отличие** пакета-сборки от простого пакета — значение свойств `Type` (тип пакета) и `ProjectPath` (относительный путь от корневого каталога пакета к файлу проекта пакета-сборки) файла `descriptor.json`. Структура простого пакета описана в статье [Общие принципы работы с пакетами](#).

Вспомогательные файлы пакета-сборки создаются или актуализируются:

- При выгрузке пакета-сборки в файловую систему, если режим разработки в файловой системе включен.
- При компиляции пакета-сборки, если режим разработки в файловой системе выключен.
- При фиксации пакета-сборки в системе контроля версий, независимо от статуса режима разработки в файловой системе.

## Вспомогательные файлы пакета-сборки:

- `[Имя пакета].csproj` — файл проекта, который содержит компилируемое содержимое пакета-сборки.
- `Directory.Build.targets` — файл, который используется для исключения содержимого пакета-сборки из компиляции основной конфигурации.

Начиная с версии 7.18.5, пакет-сборка позволяет ссылаться на внешние сборки.

Чтобы для пакета-сборки **настроить ссылку на внешнюю сборку**:

1. Откройте файл `[Имя пакета].csproj` пакета-сборки.
2. Добавьте необходимую внешнюю сборку в любую секцию файла. Исключением является секция `Label="Package References"`, содержимое которой генерируется автоматически.

Пример настройки ссылок на Quartz и внешнюю библиотеку, которая содержится в каталоге

`...\Terrasoft.Configuration\Lib` приведен ниже.

### Пример настройки ссылок на внешние сборки

```
<ItemGroup Label="3rd Party References">
  <Reference Include="Quartz">
    <HintPath>$(CoreLibPath)\Quartz.dll</HintPath>
    <Private>false</Private>
  </Reference>
  <Reference Include="ClassLibrary2">
    <HintPath>$(ConfLibPath)/ClassLibrary2.PackageName.dll</HintPath>
    <SpecificVersion>False</SpecificVersion>
    <Private>False</Private>
  </Reference>
</ItemGroup>
```

Содержимое файла `[Имя пакета].csproj` пакета-сборки допускается менять вручную. Исключением является секция `Label="Package References"`, содержимое которой генерируется автоматически.

При компиляции через Creatio IDE `TargetFramework` проекта соответствует `TargetFramework` ядра. При компиляции через внешнюю IDE используется значение по умолчанию "net472", которое при необходимости можно изменить.

В отличие от основного проекта конфигурации, в проект пакета-сборки не включены сторонние библиотеки. Для работы со сторонними библиотеками в файл проекта пакета-сборки подключите абстракции ядра или внешние сборки пакета.

**Секции со ссылками**, которые содержатся в пакете-сборке:



- Системные библиотеки.
- Внешние сборки пакета ( `<ItemGroup Label="Package Assembly References"></ItemGroup>` ).
- Библиотеки ядра ( `<ItemGroup Label="Core References"></ItemGroup>` ).
- Библиотеки зависимых пакетов ( `<ItemGroup Label="Package References"></ItemGroup>` ).

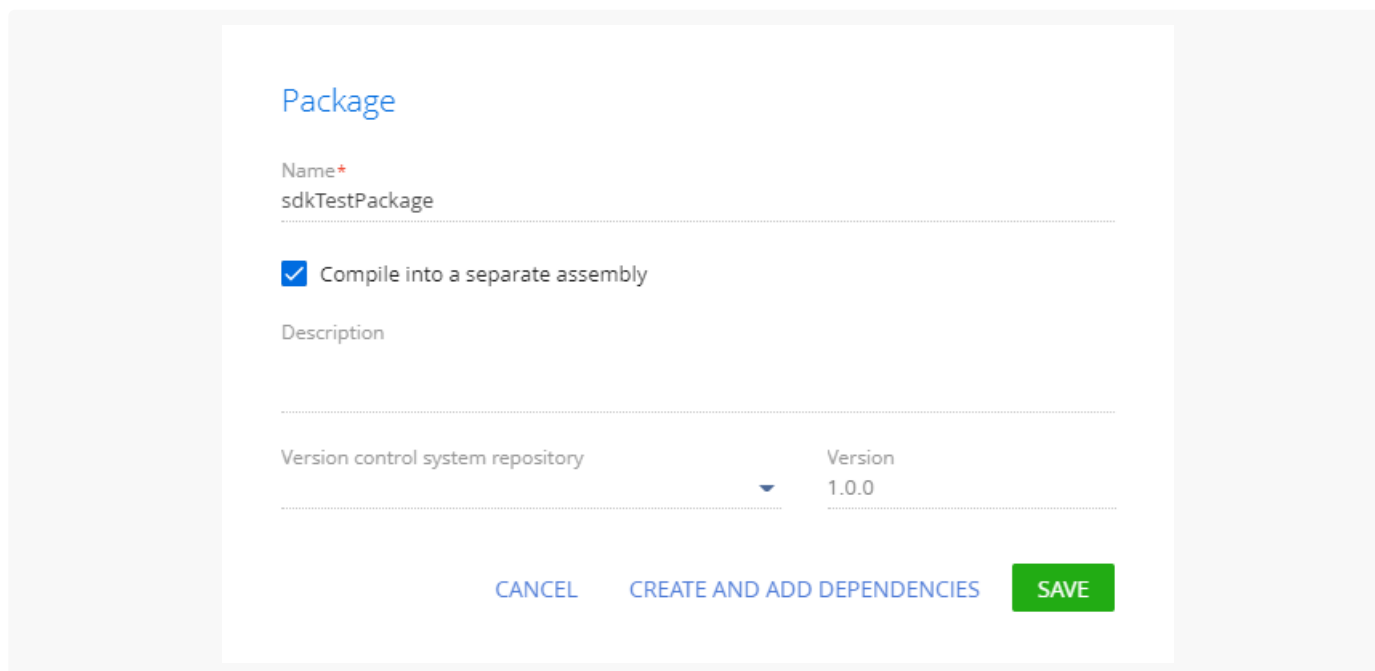
- Внешние сборки ( `<ItemGroup Label="3rd Party References"></ItemGroup>` ).

## Операции с пакетом-сборкой

Creatio позволяет создать, компилировать и удалить пакет-сборку.

### Создать пакет-сборку

1. В приложении на платформе .NET Framework перейдите в дизайнер системы по кнопке .
2. В блоке [ *Конфигурирование разработчиком* ] ([ *Admin area* ]) перейдите по ссылке [ *Управление конфигурацией* ] ([ *Advanced settings* ]).
3. В области работы с пакетами нажмите кнопку .
4. Установите признак [ *Компилировать в отдельную сборку* ] ([ *Compile into a separate assembly* ]).



5. Создайте пакет-сборку и установите его зависимости.
6. Разработайте пользовательскую функциональность.
7. В приложении на платформе .NET Framework выполните компиляцию пакета-сборки. Компиляция пакета-сборки описана в пункте [Компилировать пакет-сборку](#).
8. Перенесите пакет-сборку на другую [рабочую среду](#) на платформе .NET Core.
9. Выполните компиляцию перенесенного пакета-сборки. Компиляция пакета-сборки описана в пункте [Компилировать пакет-сборку](#).

Очередность использования платформ (.NET Framework и .NET Core) для разработки пакета-сборки не имеет значения. Важно выполнить компиляцию пакета-сборки в приложениях на платформах .NET Framework и .NET Core. Это позволит пользователям использовать пакет-сборку в приложениях на платформах .NET Framework и .NET Core.

В результате пакет-сборка будет доступен для поставки конечным пользователям.

## Компилировать пакет-сборку

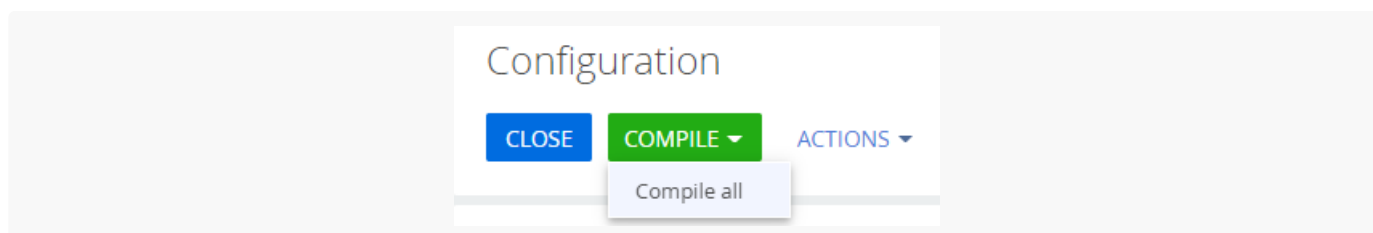
**Назначение** компиляции пакета-сборки — ускорение разработки, поскольку отсутствует необходимость перекомпиляции всей конфигурации.

Начиная с версии 7.18.5, реализована компиляция пакета-сборки, который содержит схемы [КЛИЕНТСКИХ МОДУЛЕЙ](#) (т. е. схемы с JavaScript-кодом).

**Виды компиляции** пакета-сборки:

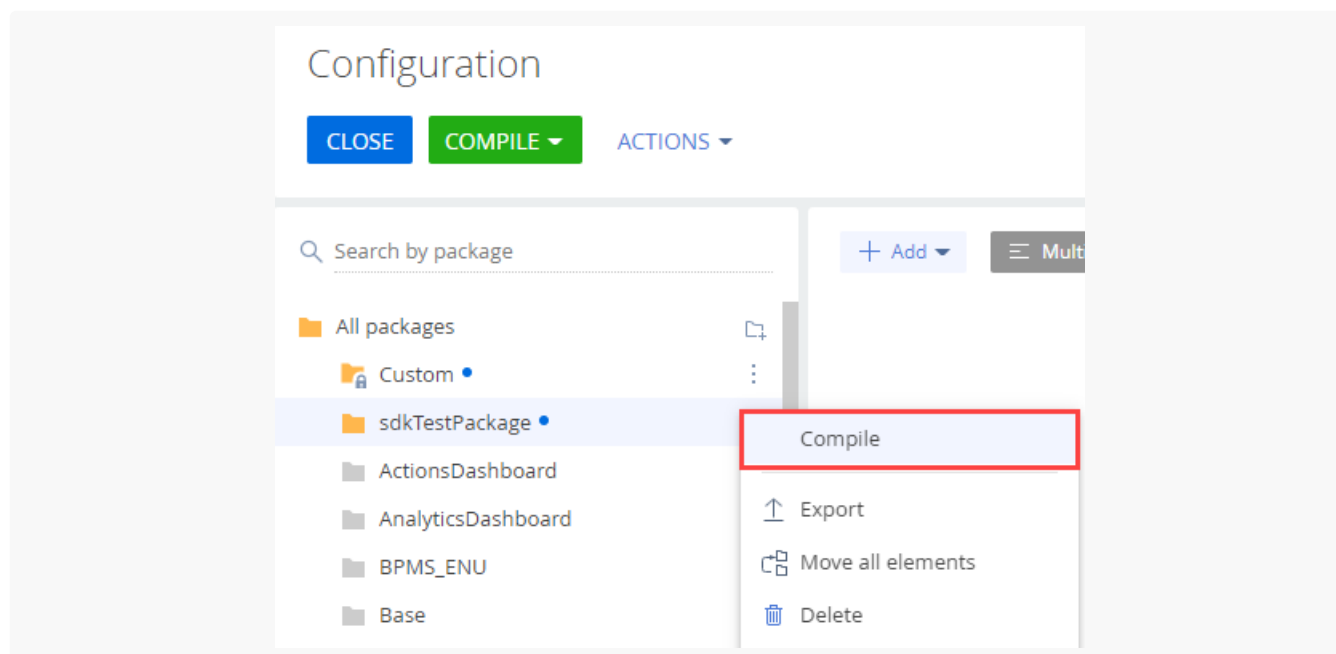
- **Компиляция конфигурации и всех пакетов-сборок.**

Выполняется нажатием кнопки [ Перекомпилировать все ] ([ *Compile all* ]) в выпадающем списке кнопки [ Компилировать ] ([ *Compile* ]) панели инструментов Creatio IDE.

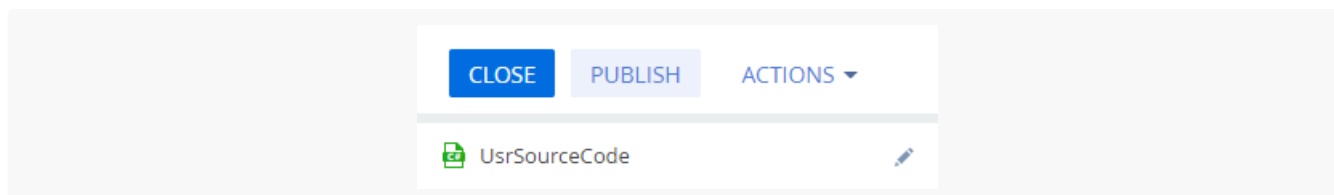


- **Компиляция текущего пакета-сборки.**

- Выполняется выбором пункта [ Компилировать ] ([ *Compile* ]) меню пакета Creatio IDE.



- Выполняется нажатием кнопки [ Опубликовать ] ([ *Publish* ]) в дизайнера исходного кода. Если редактируемая схема типа [ Исходный код ] ([ *Source code* ]) находится в пакете-сборке, то будет выполнена компиляция только пакета-сборки, а не всей конфигурации. В другом случае будет выполнена компиляция всей конфигурации.



В результате пакет-сборка будет скомпилирован. Если при компиляции пакета-сборки будет обнаружено отсутствие пакетов-сборок, от которых зависит текущий пакет-сборка, то приложение также выполнит их компиляцию.

Скомпилированный пакет-сборка будет сохранен в:

- Для приложений на платформе **.NET Framework**

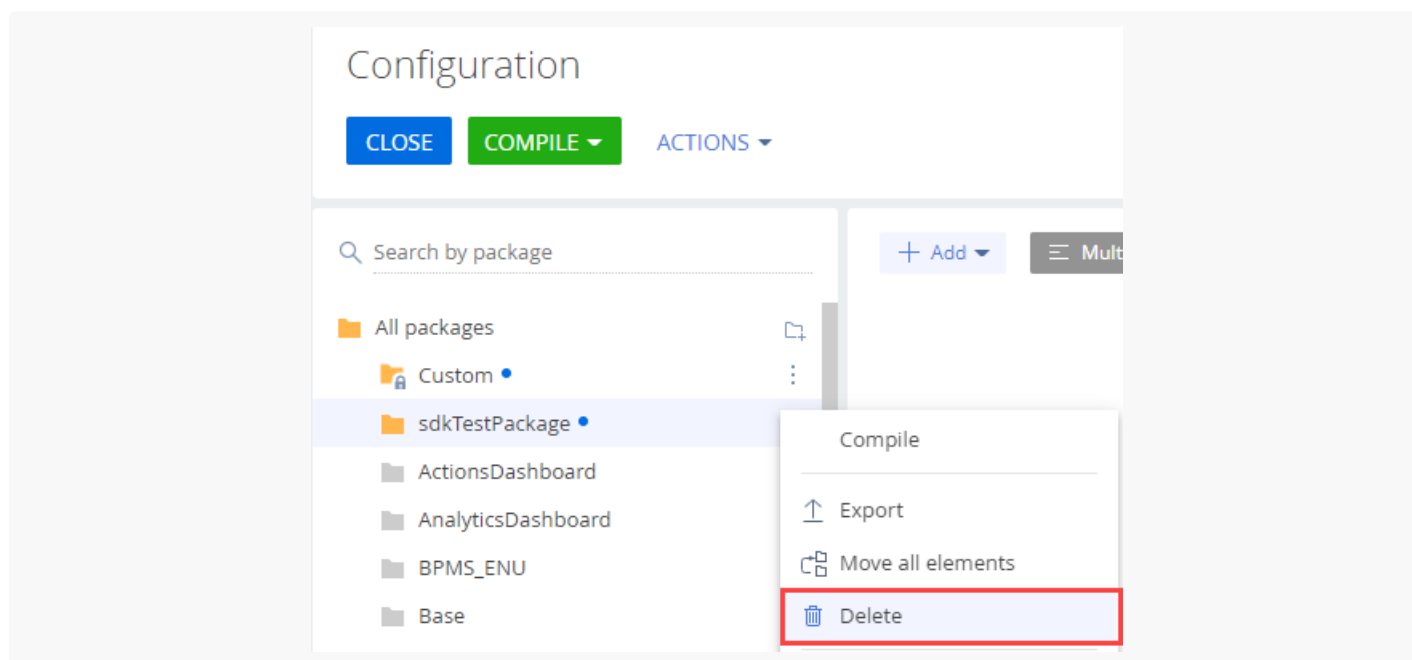
```
...\\Terrasoft.WebApp\\Terrasoft.Configuration\\Pkg\\[Имя пакета]\\Files\\Bin\\[Имя пакета].dll .
```

- Для приложений на платформе **.NET Core**

```
...\\Terrasoft.WebApp\\Terrasoft.Configuration\\Pkg\\[Имя пакета]\\Files\\Bin\\netstandard\\[Имя пакета].dll .
```

## Удалить пакет-сборку

Удаление пакета-сборки не отличается от удаления простого пакета и выполняется в пункте [ Удалить ] ([ Delete ]) меню пакета Creatio IDE.



Об удалении простого пакета читайте в статье [Принципы разработки в Creatio IDE](#).

В результате пакет-сборка будет удален из конфигурации приложения.

## Конвертация пакетов

**Виды конвертации**, которые позволяет выполнять Creatio:

- Конвертация простого пакета в пакет-сборку.



- Конвертация пакет-сборки в простой пакет.

Конвертация пакетов доступна для **on-site приложений**, поскольку требует доступа к базе данных. Для **приложений cloud** можно выполнить перемещение отдельных конфигурационных элементов в пакет-сборку.

## Конвертировать простой пакет в пакет-сборку

1. Подготовьте конфигурационные элементы простого пакета к конвертации, переписав код в соответствии с особенностями пакета-сборки.
2. С помощью SQL-запроса подготовьте простой пакет к конвертации в пакет-сборку.

### MS SQL

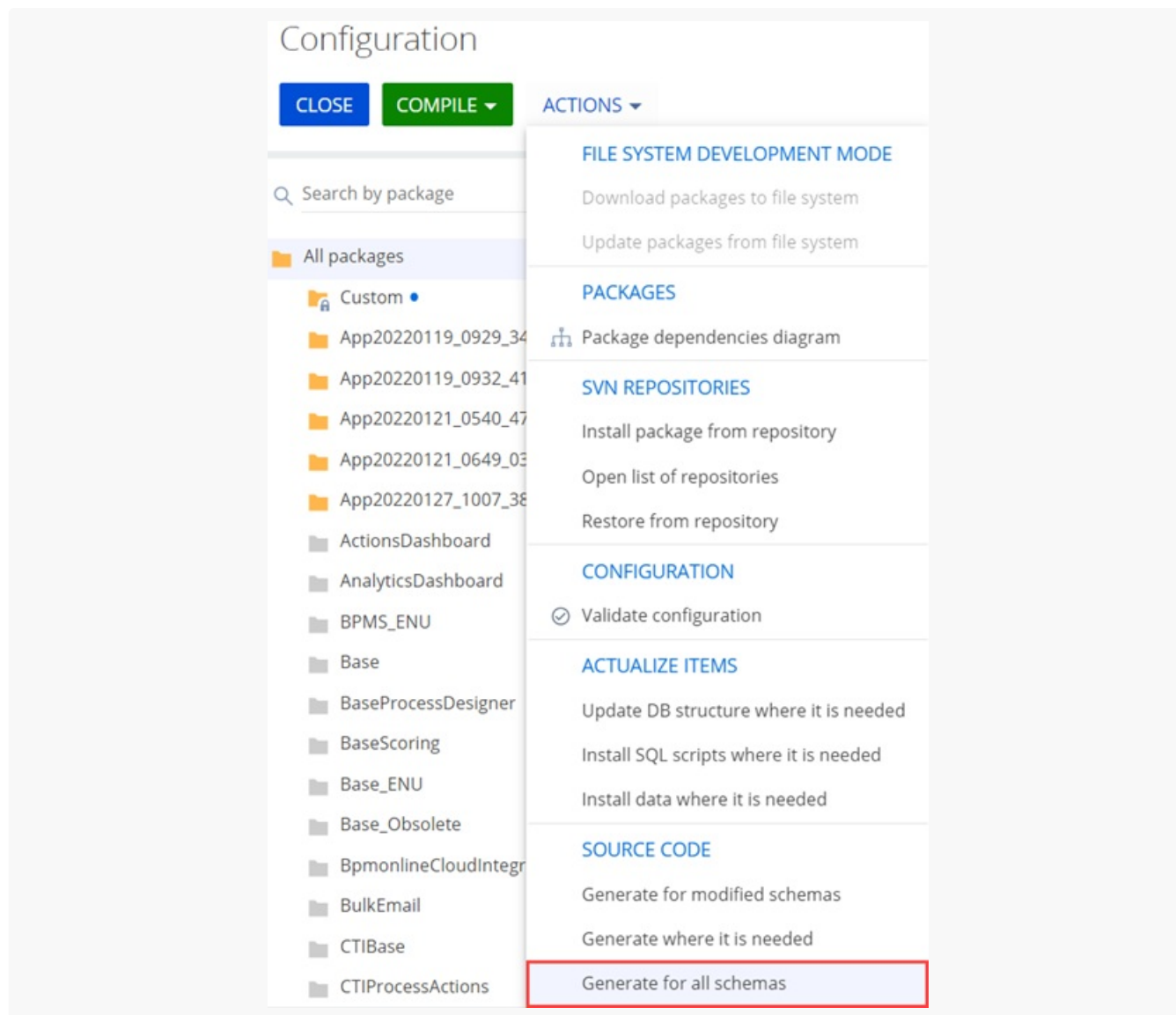
```
update [SysPackage]
set [Type] = 1, [ProjectPath] = '', ModifiedOn = GETUTCDATE()
where [Id] = 'PackageIdGuid'
```

### PostgreSQL

```
update "SysPackage"
set "Type" = 1, "ProjectPath" = '', "ModifiedOn" = current_timestamp
where "Id" = 'PackageIdGuid'
```

Этот запрос выполнит переход в таблицу `[SysPackage]` базы данных. **Действия**, которые выполняются при выполнении запроса:

- Изменение значения колонки `[Type]` с "0" (простой пакет) на "1" (пакет-сборка).
  - Заполнение колонки `[ProjectPath]`. Имя проекта должно совпадать с именем пакета.
  - Актуализация значения колонки `[ModifiedOn]`.
3. При необходимости, [добавьте ссылки на внешние сборки](#).
  4. Перезагрузите приложение. Это необходимо, чтобы сбросить кэш.
  5. Выполните генерацию всех схем простого пакета и его дочерних схем.
    - [Перейдите в раздел \[ Конфигурация \]](#) ([ *Configuration* ]).
    - В меню действий в группе [ *Исходный код* ] ([ *Source code* ]) выберите [ *Сгенерировать для всех схем* ] ([ *Generate for all schemas* ]).



6. Выполните компиляцию конфигурации.

В результате простой пакет будет конвертирован в пакет-сборку.

## Конвертировать пакет-сборку в простой пакет

1. Перейдите в базу данных и в таблице [SysPackage] найдите пакет-сборку, который планируется конвертировать в простой пакет.
2. С помощью SQL-запроса актуализируйте значения колонок [Type], [ProjectPath], [ModifiedOn].

MS SQL

```
update [SysPackage]
set [Type] = 0, [ProjectPath] = '', ModifiedOn = GETUTCDATE()
where [Id] = 'PackageIdGuid'
```

## PostgreSQL

```
update "SysPackage"
set "Type" = 0, "ProjectPath" = '', "ModifiedOn" = current_timestamp
where "Id" = 'PackageIdGuid'
```

3. В каталоге `...\Terrasoft.WebApp\Terrasoft.Configuration\Pkg\[Имя пакета]` пакета-сборки удалите каталоги:

- `Autogenerated`
- `Files\Bin`
- `Files\obj`

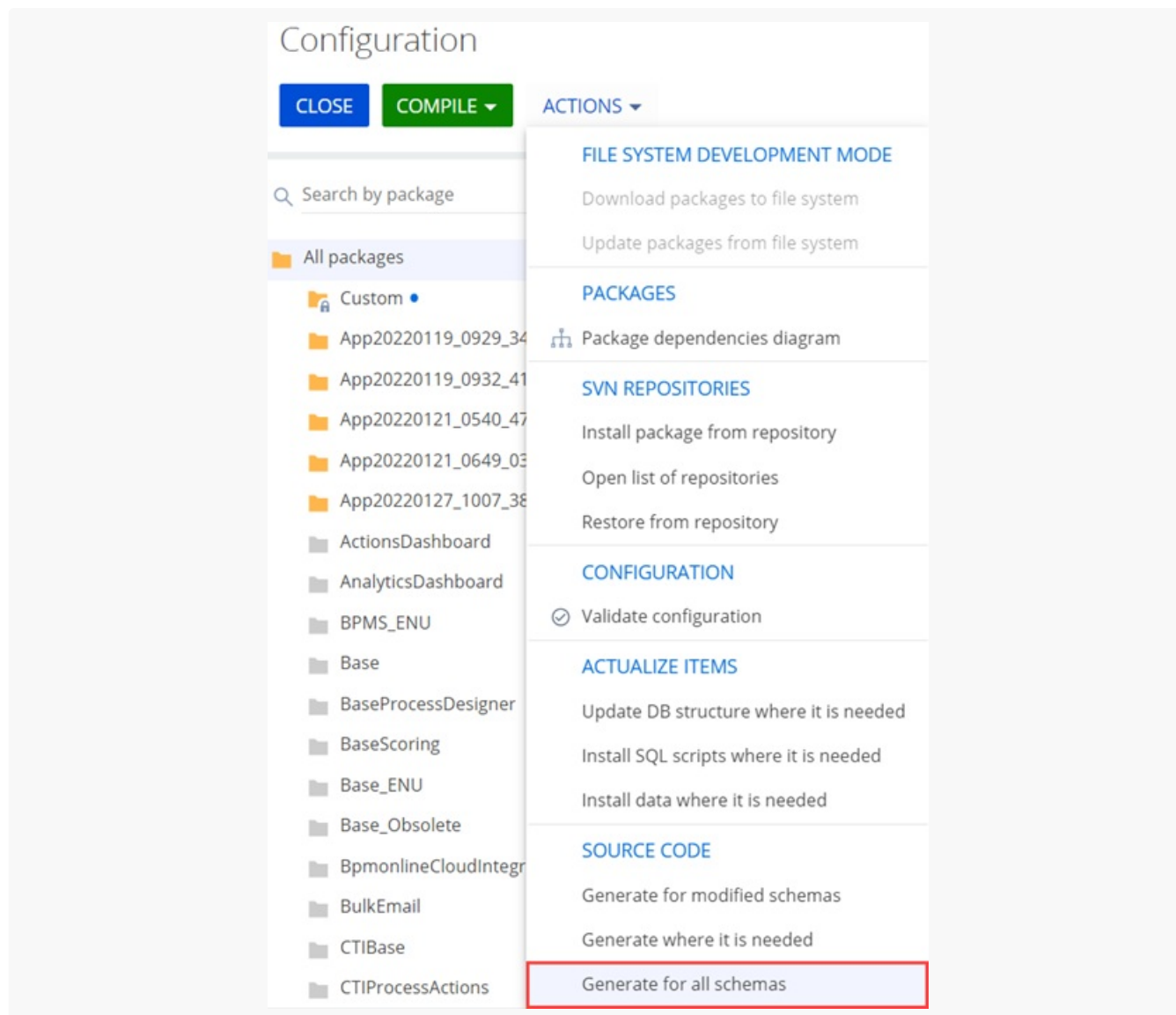
4. В каталоге `...\Terrasoft.WebApp\Terrasoft.Configuration\Pkg\[Имя пакета]` пакета-сборки удалите файлы:

- `Files\[Имя пакета].csproj`
- `Files\Directory.Build.targets`

5. Перезагрузите приложение. Это необходимо, чтобы сбросить кэш.

6. Выполните генерацию всех схем простого пакета и его дочерних схем.

- [Перейдите в раздел \[ Конфигурация \]](#) ([ *Configuration* ]).
- В меню действий в группе [ *Исходный код* ] ([ *Source code* ]) выберите [ *Сгенерировать для всех схем* ] ([ *Generate for all schemas* ]).

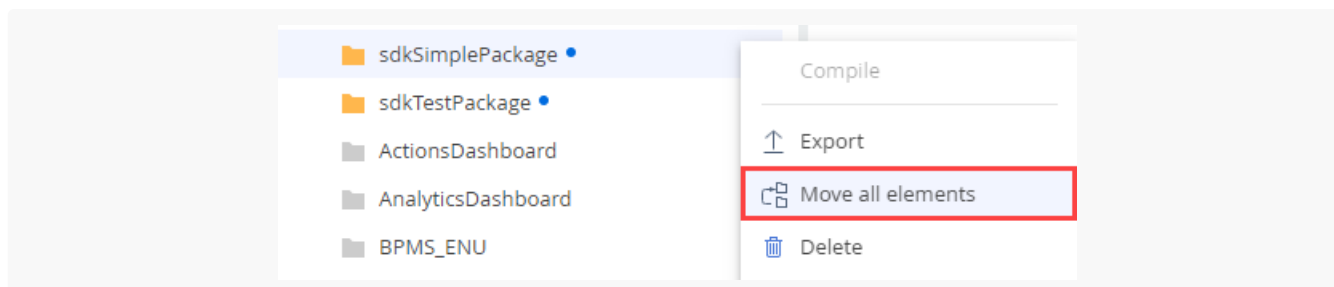


7. Выполните компиляцию конфигурации.

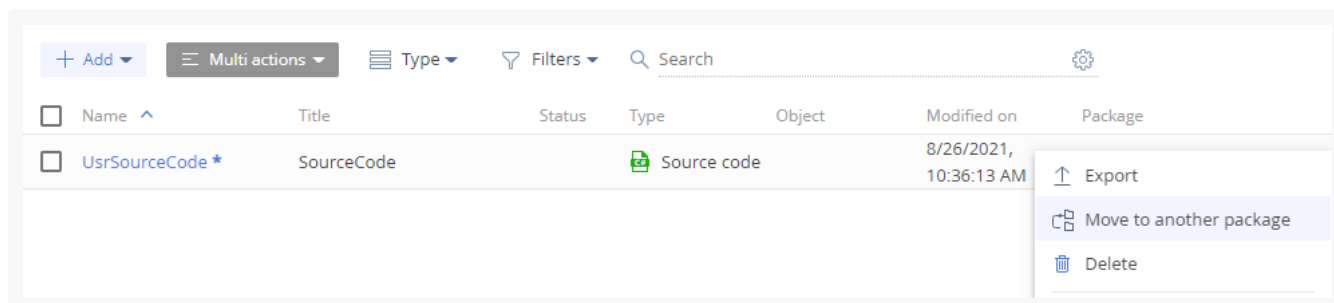
В результате пакет-сборка будет конвертирован в простой пакет.

## Переместить конфигурационные элементы в пакет-сборку

1. Подготовьте конфигурационные элементы к перемещению в пакет-сборку, переписав код в соответствии с особенностями пакета-сборки.
2. Создайте пакет-сборку. Создание пакета-сборки описано в пункте [Создать пакет-сборку](#).
3. Установите [зависимости](#) пакета-сборки. Зависимости пакета-сборки должны совпадать с зависимостями простого пакета, из которого планируется переместить конфигурационные элементы.
4. Выполните перемещение конфигурационных элементов.
  - Чтобы **переместить все конфигурационные элементы**, в меню пакета, из которого планируется переместить конфигурационные элементы, нажмите [ *Переместить все элементы* ] ([ *Move all elements* ]).



- Чтобы **переместить отдельный конфигурационный элемент**, в меню конфигурационного элемента нажмите [ *Переместить в другой пакет* ] ([ *Move to another package* ]).



5. Выполните компиляцию конфигурации.
6. Обновите зависимости дочерних пакетов: замените родительский пакет на пакет-сборку, в который были перенесены конфигурационные элементы.

В результате конфигурационные элементы будут перемещены в пакет-сборку.

## Импорт пакета-сборки

Импорт пакета-сборки не отличается от импорта простого пакета. Об импорте простого пакета читайте в статье [Перенос пакета](#).

**Особенности** импорта пакета-сборки:

- Содержимое пакета-сборки помещается в базу данных.
- Копируется файловый контент пакета-сборки.
- Компиляция конфигурации не запускается, поскольку пакет-сборка был скомпилирован при разработке и содержит в себе runtime-сборку.
- Выполняется перезапуск сайта, инициализация веб-сервисов и фабрики классов, которые содержатся в пакете-сборке.