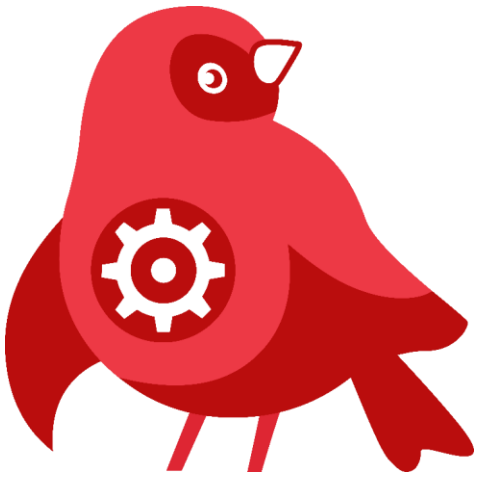
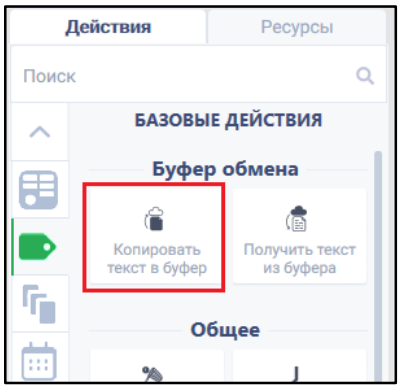


Руководство по созданию пользовательских действий на C# и Python (ROBIN SDK)

Москва, 2022

Оглавление

Определения

Понятие	Описание
Контракт действия	Строковая константа заданного формата. Позволяет однозначно идентифицировать функциональный тип действия в системе
Реализация действия	Программный код, реализующий функцию действия
Версия реализации действия	Реализация действия может иметь несколько версий для каждого язык
Платформа (тип) реализации действия	Язык программирования (Java, C#, Python), на котором реализовано действие
(Пользовательская) Версия действия	Версия действия на пользовательском уровне. Именно такую версию видит пользователь в ROBIN Studio. Скрывает от пользователя версию реализации
.Net-проект	Содержит код действия, реализованный на C#, а также его артефакты
Python-проект	Содержит код действия, реализованный на Python, а также его артефакты
Определение действия	Обязательный артефакт, содержащий описание входных параметров и результата исполнения действия, а также описание всего действия в целом
Супер-действие	Чисто техническая сущность - создается в файле ListActions.xml, используется для регистрации действий, их версий и платформ реализации в системе. В ROBIN Studio для пользователя представляют собой «кубики» действий, расположенные на левой панели на вкладке «Действия» 

Необходимое ПО

Для создания кастомных действий на компьютере разработчика должно быть установлено следующее ПО:

- среда разработки (IDE) для одного из трех языков программирования: C#, Python;
- Robin.SDK в составе:
  - ✓ Robin.ActionEditor - редактор действия;
  - ✓ Robin.ActionLoader - утилита импорта пакета действия в ROBIN Studio;

Для создания действий на Net дополнительно должно быть установлено:

- .net framework 4.8 sdk - https://dotnet.microsoft.com/download/visual-studio-sdks
- .NET Core 2.0 SDK и выше
- Visual Studio 2019 - https://visualstudio.microsoft.com/ru/

Начиная с релиза ROBIN Platform 2.9.0 появилась возможность работать в Low-code режиме, в котором помимо стандартного функционала доступно расширение возможностей платформы за счет написания кода C# во встроенном в редакторе кода и публикации этого кода в виде нового действия в ROBIN Studio.

Ссылка на документацию: http://wiki.rpa-robin.ru/LowCode/index.html.

Для создания действий на Python дополнительно должно быть установлено:

- Python 3.8 32 bit (Intel)

Общий порядок разработки пользовательского действия

При разработке действия и его интеграции в ROBIN Studio следует придерживаться следующего порядка, общего для всех используемых языков программирования:

1. До начала разработки действия или уже в процессе его разработки разработчик должен определиться с его контрактом.
2. С помощью ActionEditor разработчик:
   2.1. заполняет форму для создания определения нового действия;
   2.2. выполняет сохранение определения действия.
3. В IDE разработчик:
   3.1. создает новый проект (C# или Python) и реализует в нем действие;
   3.2. создает файл с описанием реализации действия *.robin-impinfo и размещает его в проекте вместе с кодом;
   3.3. создает пакеты со скомпилированным кодом, с транзитивными зависимостями и исходным кодом (только в Enterprise);
   3.4. импортирует в Проект действия:

- пакет с реализацией действия,
- пакеты с транзитивными зависимостями действия,
- пакет с исходным кодом действия (только в Enterpise).

4. Разработчик добавляет действие через существующее или новое супер-действие в ListActions.xml (вручную или через ActionEditor).
5. Если было создано новое супер-действие, то его необходимо добавить также в ListGroups.xml в подгруппу или группу супер-действий.
6. С помощью ActionEditor создать пакет действия с расширением *.robin-package.
7. С помощью утилиты импорта ActionLoader выполнить публикацию пакета действия в ROBIN Studio.

Создание Проекта действия

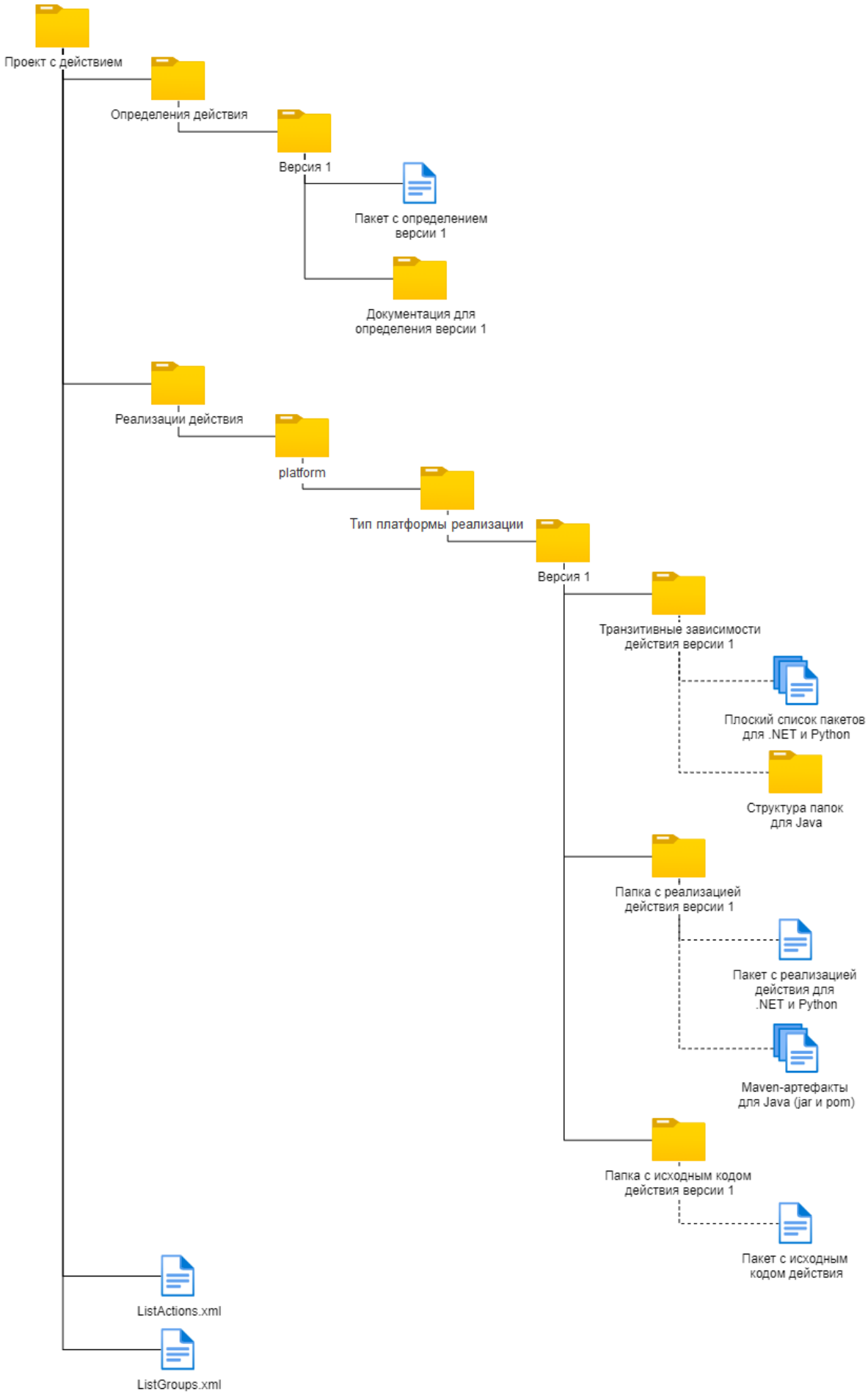
Структура проекта действия

Создание *Проекта действия* включает в себя создание структуры его папок и подготовку следующих пакетов:

- Пакет с определением действия,
- Пакет с реализацией действия,
- Пакеты с транзитивными зависимостями действия,
- Пакет с исходным кодом действия (только в Enterpise),
- ListActions.xml - файл со списком новых супер-действий, используется для регистрации пользовательских действий, их версий и платформ реализации в системе. Редактируется вручную только при необходимости.
- ListGroups.xml - файл со списком групп. Редактируется вручную только при необходимости.

Папка проекта действия, имеющая особую структуру подпапок, создается через ActionEditor. Местоположение по умолчанию - **%USERPROFILE%\Documents\Robin\Robin Actions**.

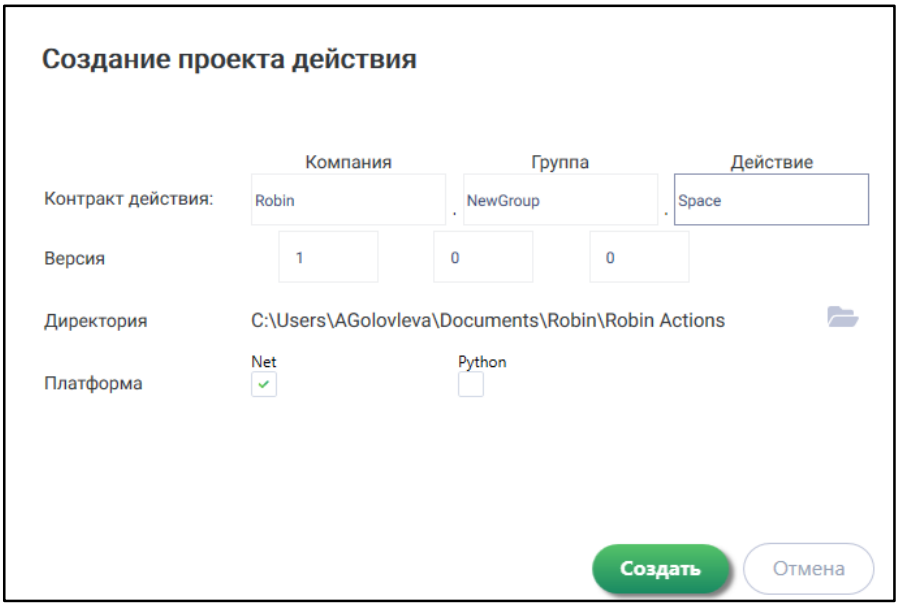
Сам проект имеет следующую структуру папок:


Проект с действием
Определения действия
Версия 1
Пакет с определением версии 1
Документация для определения версии 1
Реализации действия
platform
Тип платформы реализации
Версия 1
Транзитивные зависимости действия версии 1
Плоский список пакетов для .NET и Python
Структура папок для Java
Папка с реализацией действия версии 1
Пакет с реализацией действия для .NET и Python
Maven-артефакты для Java (jar и pom)
Папка с исходным кодом действия версии 1
Пакет с исходным кодом действия
ListActions.xml
ListGroups.xml

Создание структуры папки проекта действия

1. Запустить ActionEditor.
2. Нажать «*Создать проект действия*».
3. Указать контракт действия:
   - наименование компании;
   - придумать или использовать уже существующую группу действий, которой будет принадлежать новое действие;
   - придумать наименование нового действия.

   Каждое слово в названии компании, группы и действия выделяется заглавной буквой, например, MyCustomAction.
4. Указать версию определения действия. По умолчанию, начальная версия – 1.0.0. Версия определения и версия реализации действия между собой не связаны и указываются в разных местах.
5. Указать директорию, где будет создан проект действия. По умолчанию – **%USERPROFILE%\Documents\Robin\Robin Actions** .
6. Указать платформу (язык программирования), на которой будет реализовано действие.



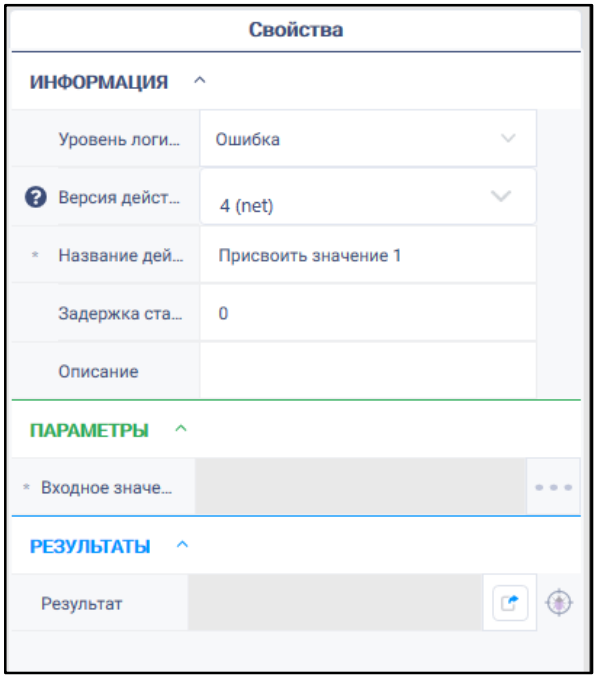
7. Нажать на кнопку «*Создать*». В итоге в папке **%USERPROFILE%\Documents\Robin\Robin Actions\<Контракт действия>** будет создан Проект действия со следующим списком артефактов:
   - **<Контракт действия>.action-project** - файл с проектом действия, который можно открыть на редактирование в ActionEditor;
   - **ListActions.xml** - XML-файл с описанием действия;
   - **ListGroups.xml** - XML-файл с описанием группы действия;

- **<Контракт действия>-1.0.0.robin-def** - шаблон файла с описанием действия.

Создание определения действия

Создание действия лучше начинать с подготовки пакета определения. Создание определения происходит с помощью ActionEditor, который, помимо создания пакета с определением, формирует ещё и весь проект с действием, имеющий особую структуру папок.

При создании определения действия из всех свойств действия разработчик указывает только поля для блоков «Параметры» и «Результаты» (в ROBIN Studio отображаются на правой панели в свойствах действия). Блок «Информация» - это стандартный блок, его нельзя создать через ActionEditor; он будет создан автоматически.



Порядок действий для создания определения:

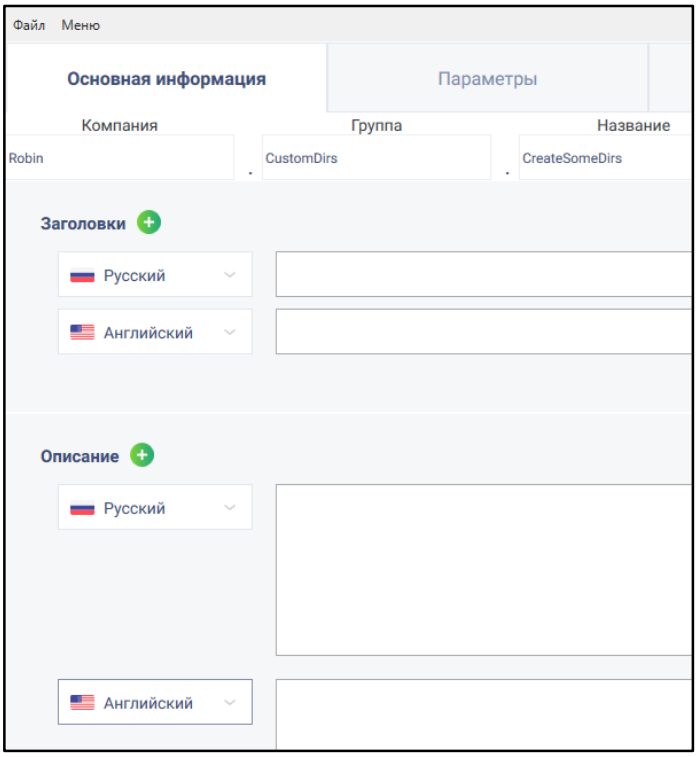
1. Если для данного действия еще не существует папки его проекта, выполнить шаги из предыдущего раздела по созданию Проекта действия. После выполнения этих шагов откроется форма с 4 вкладками для заполнения параметров определения.
2. Если для данного действия ранее уже был создан Проект:

   2.1. Запустить ActionEditor.

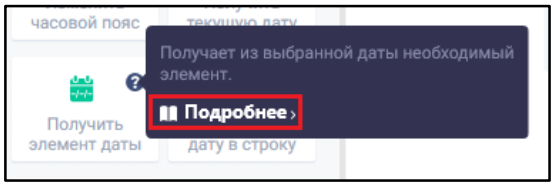
   2.2. Нажать на «Открыть действие» и выбрать через окно проводника соответствующий файл **<Контракт действия>.action-project.**

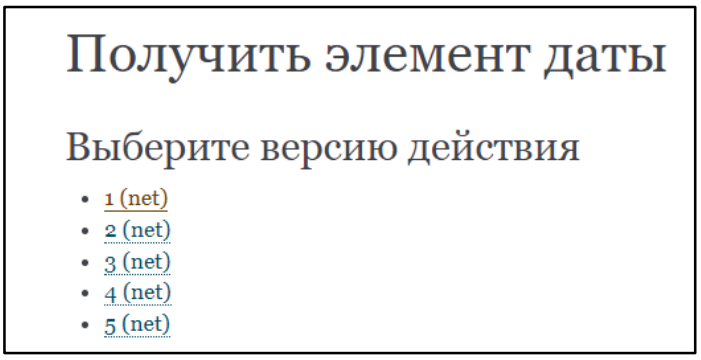
   2.3. Откроется форма с 4 вкладками для заполнения\изменения параметров определения.

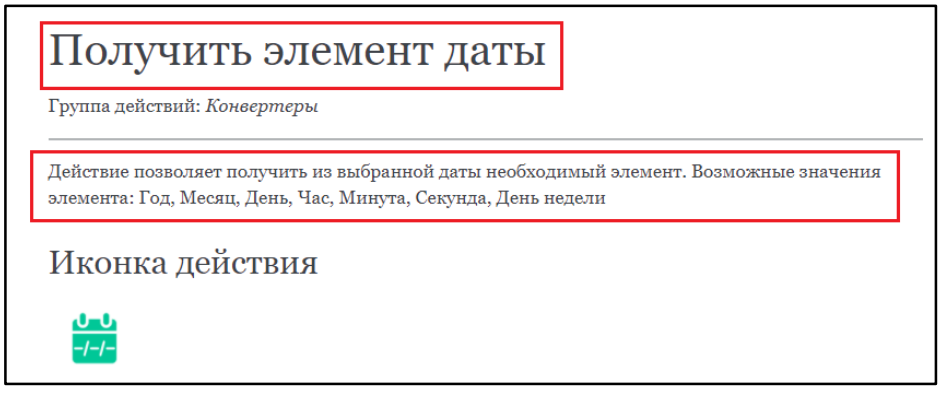
3. На первой вкладке «*Основная информация*» указан контракт действия и кнопки для добавления Заголовков и Описания действия на русском и английском языках. Поля заголовков и описания заполнять необязательно.



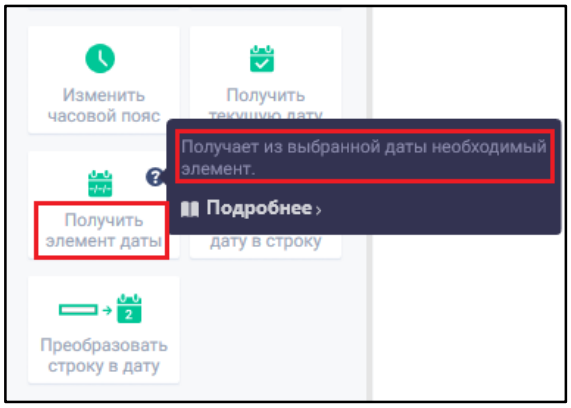
У каждой версии действия может быть свой заголовок и описание, и такая информация в дальнейшем будет отражаться в документации на это действие, которую можно вызвать из ROBIN Studio.



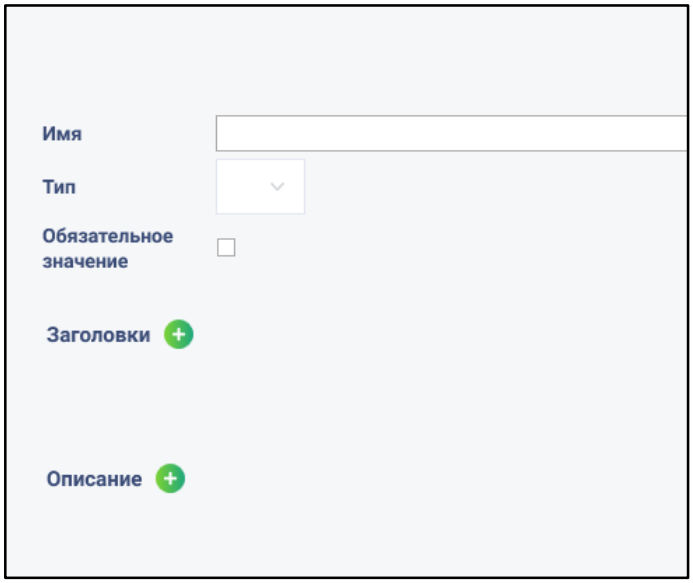




Но это не является названием и описанием, которые пользователь видит в ROBIN Studio на «кубике» действия на левой панели с действиями. Такие название и описание указываются на других вкладках в ActionEditor.



4. На вкладке «*Параметры*» необходимо создать поля (переменные), которые потом будут отображаться в ROBIN Studio пользователю в блоке «Параметры» в свойствах созданного действия.
   Нажать «*Добавить параметр*» и заполнить открывшуюся форму.



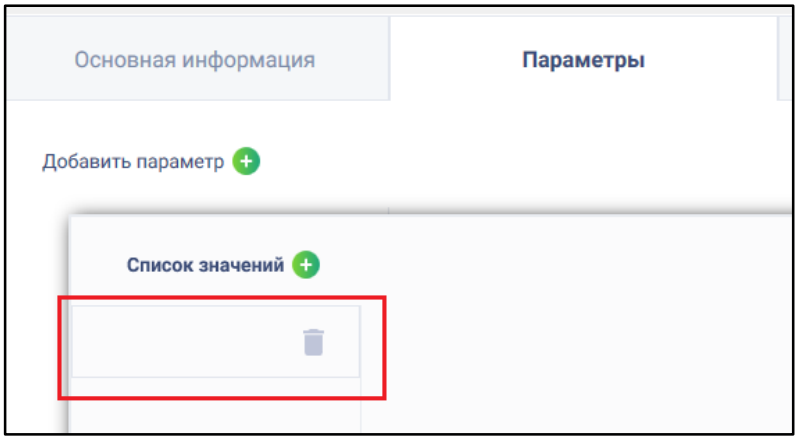
- Указать Имя параметра. Оно указывается только на английском. Это не название поля, которое будет отражено в ROBIN Studio, оно будет использовано в коде действия.
- Указать Тип – тип переменной, которую можно будет использовать в данном поле.
- Отметить, является ли поле обязательным.
- Указать Заголовки – название поля. Именно это название будет отображаться у данного поля в ROBIN Studio. Все заголовки и описания можно указать в 2 вариантах – на русском и на английском, но указывать всегда оба варианта не обязательно, достаточно заполнить хотя бы на одном языке.
- Указать Описание. Это то описание, которое мы видим в ROBIN Studio, когда нажимаем на знак вопроса рядом с полем параметра.
- Нажать «Сохранить».

5. После того, как новый параметр будет создан, при нажатии на его строку отобразится форма с данными, которые заполняли ранее, а также появится кнопка «*Возможные значения*» (зависит от типа переменной данного поля). Если предполагается, что созданное поле должно представлять выпадающий список с заранее определенными значениями (например, как в действиях, где мы выбираем тип клика – ЛКМ, ПКМ и т.д.), то значения этого списка задаются через эту кнопку:

   5.1. Нажать на кнопку «*Возможные значения*».

   5.2. В открывшемся окне нажать на кнопку «*Список значений*». Каждое новое значение выпадающего списка создается через эту кнопку.

   5.3. Нажать на появившийся пустой прямоугольник.



   5.4. В поле «*Значение*» указать название добавляемого элемента списка. Это название нужно только для кода действия.

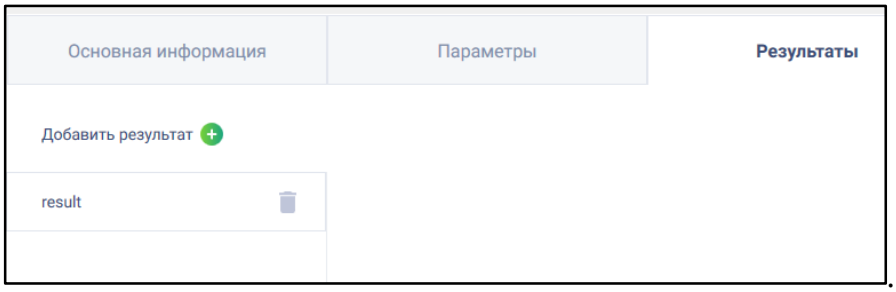
   5.5. В «*Заголовках*» указать название добавляемого элемента списка. Именно оно отобразиться в ROBIN Studio пользователю.

   5.6. В «*Описании*» указать описание данного элемента списка.

   5.7. Повторить шаги 5.2. – 5.6. для добавления других значений выпадающего списка.

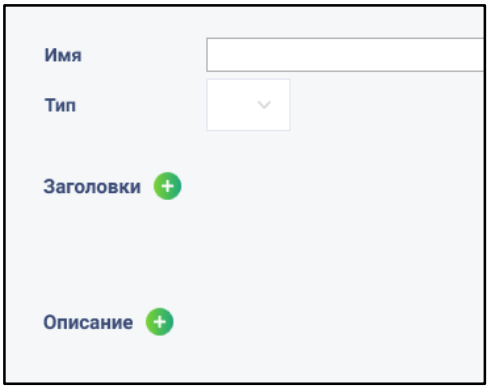
   5.8. Нажать «*Сохранить*».

6. На вкладке «*Результаты*» создать поля, которые потом будут отображаться пользователю в ROBIN Studio в блоке «Результаты» в свойствах созданного действия.



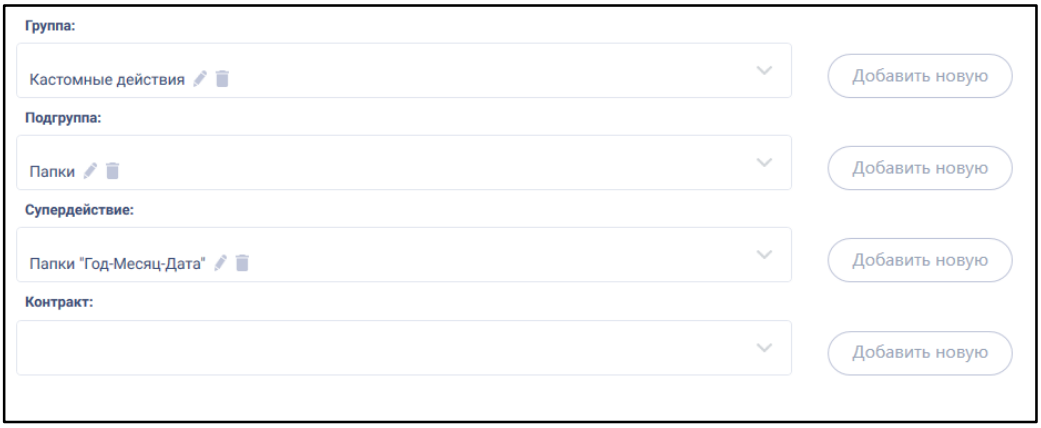
   Нажать на кнопку «*Добавить результат*» и заполнить поля открывшегося окна:

- Указать Имя результата. Оно указывается только на английском. Это не название поля, которое будет отражено в ROBIN Studio, оно будет использовано в коде действия.
- Указать Тип – тип переменной, которая будет сохранять в этот результат действия.
- Указать Заголовки – название поля результата. Именно это название и будет отображаться у данного поля в ROBIN Studio.
- Указать Описание. Это то описание, которое мы видим в ROBIN Studio, когда нажимаем на знак вопроса рядом с полем результата.
- Нажать «Сохранить».

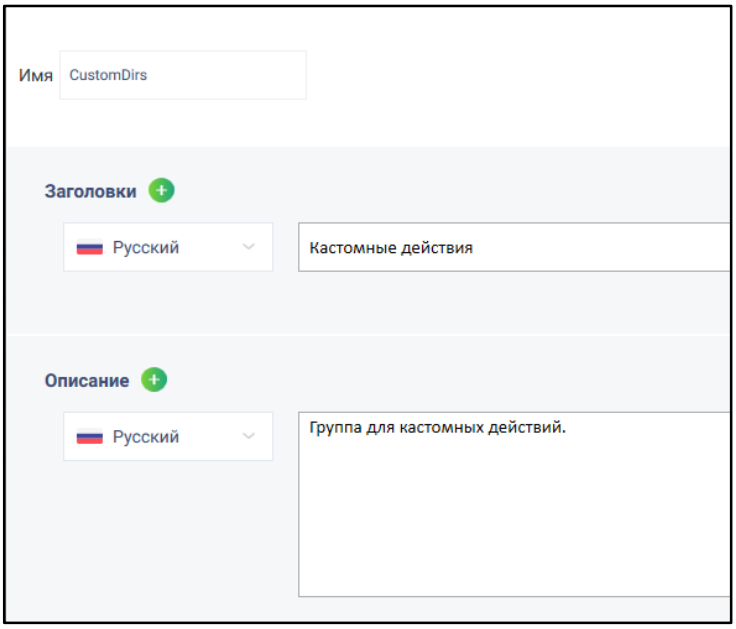


У Результата нет возможности создавать поля с выпадающим списком значений, поэтому кнопка «*Возможные значения*» появляться не будет.

7. На вкладке «*Настройки робота*» заполнить поля:



7.1. ***Группа.*** Указываем группу ROBIN Studio, в которую будет входить действие. Нажать на кнопку *«Добавить новую»* и заполнить поля открывшейся формы:



- в поле *«Имя»* указать название группы на английском. По данному имени группа будет идентифицирована внутри ROBIN Studio. Но такое название не будет отображаться пользователю.

  Если предполагается создать новую группу, можно указать любое значение, например, NewGroup.

  Если добавляем действие в уже существующую группу, то необходимо указать именно ее имя. Узнать имена уже существующих групп можно в файле **ListGroups.xml**, который выгружается из кэша ROBIN Studio через ActionLoader нажатием на кнопку «*Выгрузить списки действий*». Выгруженный файл включает в себя только структуру групп и подгрупп без наполнения (т.е. какие именно супер-действия туда входят).

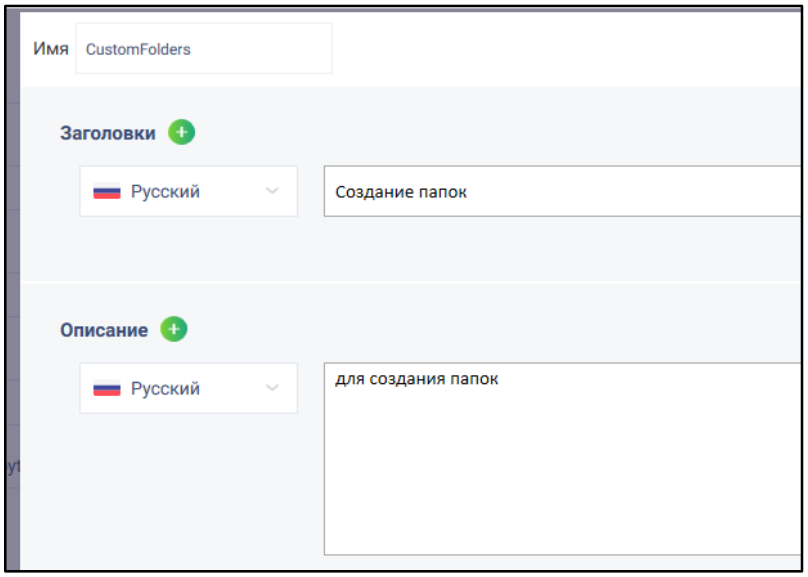
- В поле *«Заголовки»* указать название новой или уже существующей группы. Именно это название будет видеть пользователь в ROBIN Studio.

  Если необходимо добавить действие в уже существующую группу, ввести название, которое соответствует имени, указанному в поле «*Имя*». Такое название существующей группы также можно посмотреть в **ListGroups.xml**.

- В поле «Описание» указать описание группы. В ROBIN Studio оно не будет отражено, но будет добавлено в файл **ListGroups.xml**.

  Если необходимо добавить действие в уже существующую группу, ввести описание, которое соответствует имени, указанному в поле «*Имя*». Такое описание существующей группы также можно посмотреть в **ListGroups.xml**.

7.2. **Подгруппа.** Здесь все аналогично группе: указать подгруппу, которая будет содержаться в выбранной ранее группе, и в которую будет добавлено новое действие. Нажать на кнопку *«Добавить новую»* и заполнить поля открывшейся формы:



- в поле *«Имя»* указать название подгруппы на английском. По данному имени подгруппа будет идентифицирована внутри ROBIN Studio. Но такое название не будет отображаться пользователю.

  Если предполагается создать новую подгруппу, можно указать любое значение, например, NewSubgroup.

  Если добавляем действие в уже существующую подгруппу, то необходимо указать именно ее имя. Узнать имена уже существующих подгрупп можно в файле **ListGroups.xml**.

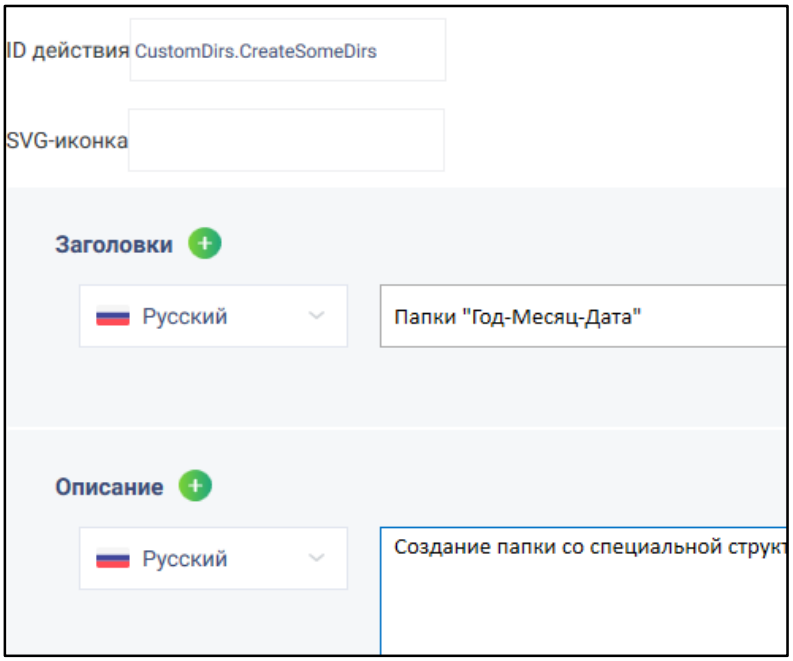
- В поле *«Заголовки»* указать название новой или уже существующей подгруппы. Именно это название будет видеть пользователь в ROBIN Studio.

  Если необходимо добавить действие в уже существующую подгруппу, ввести название, которое соответствует имени, указанному в поле «*Имя*». Такое название существующей подгруппы также можно посмотреть в **ListGroups.xml**.

- В поле «Описание» указать описание подгруппы. В ROBIN Studio оно не будет отражено, но будет добавлено в файл **ListGroups.xml**.

  Если необходимо добавить действие в уже существующую подгруппу, ввести описание, которое соответствует имени, указанному в поле «*Имя*». Такое описание существующей подгруппы также можно посмотреть в **ListGroups.xml**.

7.3. **Супердействие**. Указываем определение (описание) для действия. Нажать на кнопку *«Добавить новую»* и заполнить поля открывшейся формы:



- В поле *«ID действия»* указать любую фразу на английском. Можно фразу из нескольких слов, но они должны быть записаны слитно или через точку.

- В поле *«SVG-иконка»* должно быть указано xml-описание иконки будущего действия. На основе данного описания будет сформировано изображение иконки действия, которое пользователь увидит в ROBIN Studio. Но на данный момент это поле заполнять не нужно, иконка указывается вручную только через файл **ListActions.xml**:

  — Открыть SVG-файл иконки через блокнот и скопировать все элементы, которые являются дочерними для элемента *svg*. Пример svg-файла:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
 "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg version="1.0" xmlns="http://www.w3.org/2000/svg"
 width="245.000000pt" height="135.000000pt" viewBox="0 0 245.000000 135.000000"
 preserveAspectRatio="xMidYMid meet">

<g transform="translate(0.000000,135.000000) scale(0.100000,-0.100000)"
fill="#000000" stroke="none">
<path d="M670 690 l0 -200 520 0 520 0 0 200 0 200 -520 0 -520 0 0 -200z
m990 -5 l0 -155 -465 0 -465 0 0 155 0 155 465 0 465 0 0 -155z"/>
<path d="M370 505 c0 -42 3 -55 15 -55 8 0 15 9 15 20 0 19 7 20 110 20 103 0
110 -1 110 -20 0 -11 7 -20 15 -20 12 0 15 13 15 55 0 42 -3 55 -15 55 -8 0
-15 -7 -15 -15 0 -12 -19 -15 -110 -15 -91 0 -110 3 -110 15 0 8 -7 15 -15 15
-12 0 -15 -13 -15 -55z"/>
<path d="M1740 505 c0 -30 5 -55 10 -55 6 0 10 9 10 20 0 19 7 20 120 20 113
0 120 -1 120 -20 0 -11 5 -20 10 -20 6 0 10 25 10 55 0 30 -4 55 -10 55 -5 0
-10 -7 -10 -15 0 -12 -20 -15 -120 -15 -100 0 -120 3 -120 15 0 8 -4 15 -10
15 -5 0 -10 -25 -10 -55z"/>
</g>
</svg>
```

Необходимо скопировать:

```
<path d="M670 690 l0 -200 520 0 520 0 0 200 0 200 -520 0 -520 0 0 -200z
m990 -5 l0 -155 -465 0 -465 0 0 155 0 155 465 0 465 0 0 -155z"/>
<path d="M370 505 c0 -42 3 -55 15 -55 8 0 15 9 15 20 0 19 7 20 110 20 103 0
```

```
110 -1 110 -20 0 -11 7 -20 15 -20 12 0 15 13 15 55 0 42 -3 55 -15 55 -8 0
-15 -7 -15 -15 0 -12 -19 -15 -110 -15 -91 0 -110 3 -110 15 0 8 -7 15 -15 15
-12 0 -15 -13 -15 -55z"/>
<path d="M1740 505 c0 -30 5 -55 10 -55 6 0 10 9 10 20 0 19 7 20 120 20 113
0 120 -1 120 -20 0 -11 5 -20 10 -20 6 0 10 25 10 55 0 30 -4 55 -10 55 -5 0
-10 -7 -10 -15 0 -12 -20 -15 -120 -15 -100 0 -120 3 -120 15 0 8 -4 15 -10
15 -5 0 -10 -25 -10 -55z"/>
```

— После заполнения последнего поля «Контракт» и сохранения определения в ActionEditor, открыть файл **ListActions.xml**. Файл расположен по пути **%USERPROFILE%\Documents\Robin\Robin Actions\<Контракт действия>\<Контракт действия>\ListActions.xml**

— Вставить после элемента *</ActionContracts>* следующие элементы:

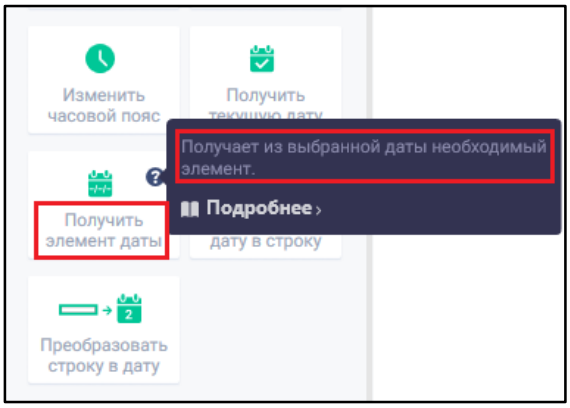
```
<IconSvg>
        <svg width="54" height="29" viewBox="0 0 54 29"
preserveAspectRatio="xMidYMid meet" xmlns="http://www.w3.org/2000/svg">
          <Часть, скопированная из svg-файла>
        </svg>
      </IconSvg>
```

Пример:

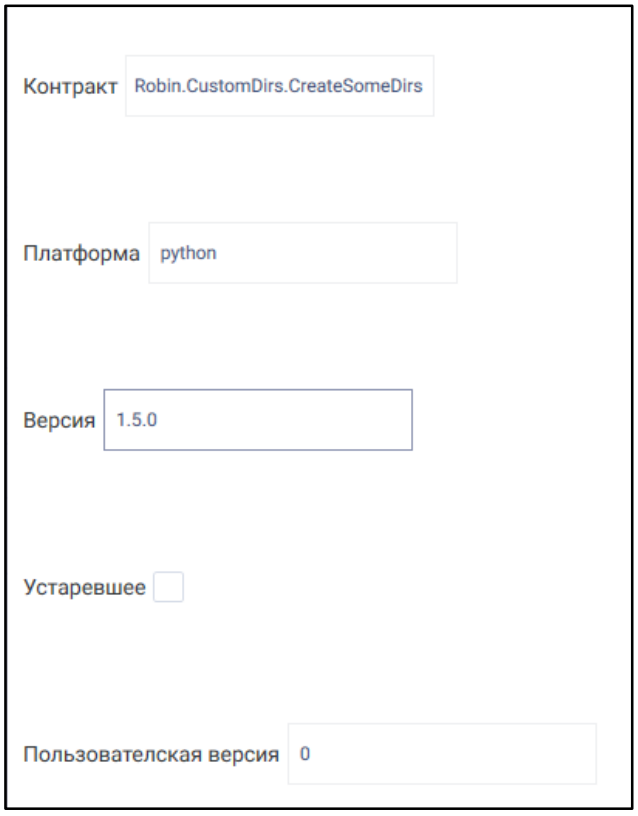
```
<IconSvg>
        <svg width="54" height="29" viewBox="0 0 54 29"
preserveAspectRatio="xMidYMid meet" xmlns="http://www.w3.org/2000/svg">
          <path d="M670 690 l0 -200 520 0 520 0 0 200 0 200 -520 0 -520 0 0 -200z
m990 -5 l0 -155 -465 0 -465 0 0 155 0 155 465 0 465 0 0 -155z"/>
          <path d="M370 505 c0 -42 3 -55 15 -55 8 0 15 9 15 20 0 19 7 20 110 20 103 0
110 -1 110 -20 0 -11 7 -20 15 -20 12 0 15 13 15 55 0 42 -3 55 -15 55 -8 0
-15 -7 -15 -15 0 -12 -19 -15 -110 -15 -91 0 -110 3 -110 15 0 8 -7 15 -15 15
-12 0 -15 -13 -15 -55z"/>
          <path d="M1740 505 c0 -30 5 -55 10 -55 6 0 10 9 10 20 0 19 7 20 120 20 113
0 120 -1 120 -20 0 -11 5 -20 10 -20 6 0 10 25 10 55 0 30 -4 55 -10 55 -5 0
-10 -7 -10 -15 0 -12 -20 -15 -120 -15 -100 0 -120 3 -120 15 0 8 -4 15 -10
15 -5 0 -10 -25 -10 -55z"/>
        </svg>
      </IconSvg>
```

— Сохранить изменения в файле **ListActions.xml**

- В поле *«Заголовки»* указать название нового действия, которое будет отображаться пользователю в ROBIN Studio.
- В поле *«Описание»* указать текст, который будет указан в ROBIN Studio в подсказке к создаваемому действию.

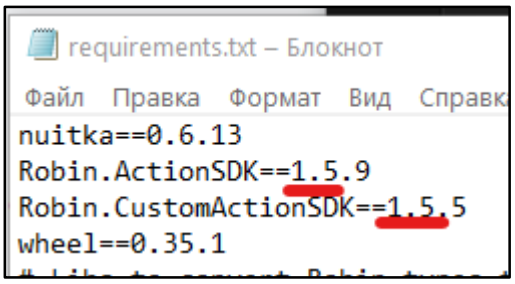


7.4. **Контракт**. Нажать на кнопку *«Добавить новую»* и заполнить поля открывшейся формы:

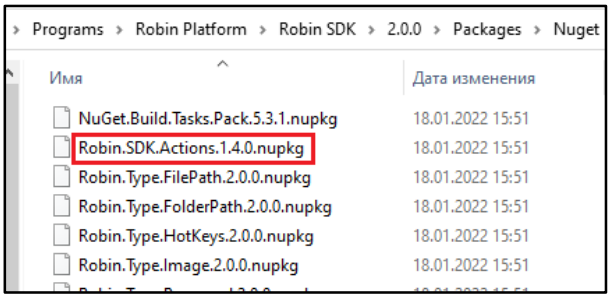


- В поле *«Контракт»* указать контракт действия, который указывали при создании проекта действия. Пример: Robin.CustomDirs.CreateSomeDirs.
- В поле *«Платформа»* указать название платформы, на которой будет реализовано действие: net или python.
- В поле *«Версия»* указать версию реализации действия. Пример: 1.5.0.

  В отличии от версии определения, это не может быть любым номером. Версию реализации для действий на python можно посмотреть в файле **%localappdata%\Programs\Robin Platform\Robin SDK\2.0.0\PythonSDK\requirements.txt**.



  Версию реализации для действий на net можно посмотреть в названии файла **%localappdata%\Programs\Robin Platform\Robin SDK\2.0.0\Packages\Nuget\Robin.SDK.Actions.1.4.0.nupkg**



Мажор.Минор версия реализации действия должна соответствовать Мажор.Минор версии движка. Например, если версия движка 1.5.75, то версия действия должна быть 1.5.*, где * - это число (номер патча). Т.е. по своему усмотрению пользователь выбирает только 3е число в значении версии, которое относится к версии патча (1.5.**1**, 1.5.**2** и т.п).

- Чекбокс *«Устаревшее»* необходимо активировать, если наша цель - не создание нового действия, а просто отключение определенной версии уже существующего в ROBIN Studio супер-действия. Сама версия все равно будет видна пользователю в ROBIN Studio, но ее нельзя будет выбрать, и она будет подсвечена серым.

- Поле *«Пользовательская версия»* сейчас не редактируется. По умолчанию установлено значение 0. Именно этот номер версии пользователь увидит в ROBIN Studio в поле «*Версия действия*» в свойствах этого действия. Но можно вручную поменять это номер: сохранить определение в ActionEditor, открыть файл **ListActions.xml** и указать необходимый номер версии в значении атрибута *superActionVersion*.

8. Нажать «*Файл*» - «*Сохранить определение действия*» для сохранения внесенных изменений и закрыть ActionEditor.

9. Если группа, в которую должно быть добавлено новое супер-действие, уже содержит какие-то другие супер-действия, после выполнения предыдущего шага необходимо в **ListGroups.xml** (в папке проекта действия) указать все эти супер-действия и их подгруппы. Иначе при импорте готового пакета действия в состав ROBIN Studio, все остальные супер-действия из указанной группы не будут отображаться пользователю. Будет доступно только новое действие. Поэтому в **ListGroups.xml** необходимо после элемента *</SubGroup>* на каждое супер-действие, которое уже содержится в выбранной группе в ROBIN Studio, добавить строку:

```
<SubGroup name="Имя подгруппы супер-действия"><Titles><Title lang="ru-RU"><content>Название подгруппы супер-действия</content></Title></Titles><Descriptions><Description lang="ru-RU"><content>Описание подгруппы супер-действия</content></Description></Descriptions><Actions><Action actionID="ID супер-действия" deprecated="false" /></Actions></SubGroup>
```

Значения атрибутов заполнять в соответствии с этими супер-действиями.

Создание Python-действия

Сторонние зависимости (3rd party libraries)

У системы исполнения есть ограничения на использование некоторых сторонних библиотек. Если при создании действия были использованы библиотеки из следующего списка, то версии этих библиотек должны в точности соответствовать версиям, приведённым ниже:

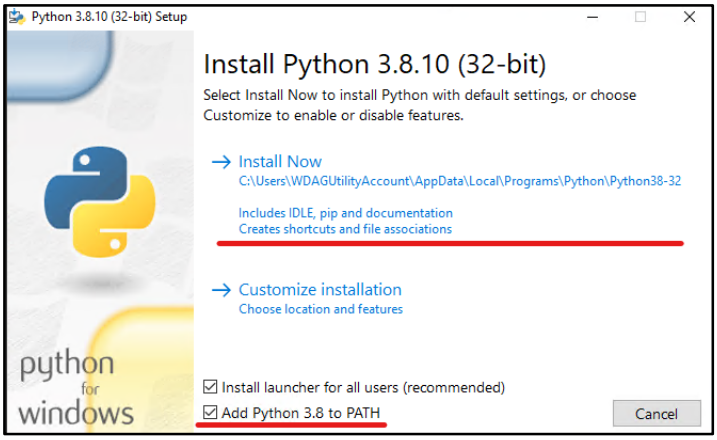
- cffi==1.14.5
- cryptography==3.4.7
- lxml==4.5.0
- Pillow==6.2.2
- protobuf==3.12.2
- pycparser==2.20
- pycryptodome==3.9.8
- pystray==0.16.0
- PyYAML==5.4.1
- pyzmq==19.0.1
- six==1.15.0

Пример:

При создании действия, Вы установили и использовали библиотеку Pillow версии 8.4.0. В то время как система исполнения Python использует библиотеку Pillow версии 6.2.2 (версия библиотеки указана в списке выше). Действие при запуске будет использовать библиотеку той же версии, что и движок. Следовательно, часть функционала данной библиотеки, может стать недоступной действию.

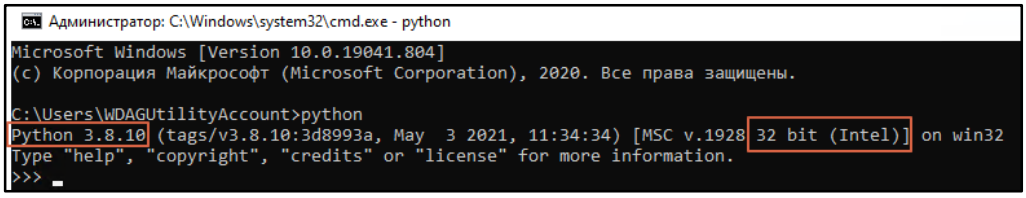
Настройка окружения

Установка Python. Ссылка для загрузки: https://www.python.org/ftp/python/3.8.10/python-3.8.10.exe. После загрузки, запустить python-3.8.10.exe, поставить галочку на "Add Python 3.8 to PATH" и нажать "Install Now"



Проверка версии Python

Открыть командную строку и набрать **python**. Проверить на соответствие версию и разрядность установленного интерпретатора.



Создание и настройка проекта реализации

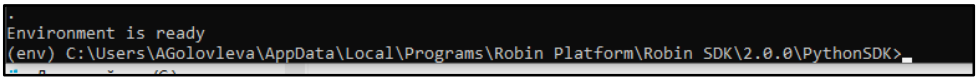
После того, как был установлен "Robin Platform SDK", на рабочем столе должен быть ярлык под названием «**PythonSDK**».

1. Перейти по ярлыку и запустить файл «**PrepareEnvironment.bat**». Подождать пока bat-файл закончит работу и не выходить из консоли после конца его работы. Это нужно для того, чтобы не активировать виртуальное окружение Python повторно. По завершению работы bat-файла, в консоли должна появиться строка "*Environment is ready*". Таким образом, мы создали виртуальное окружение Python и установили в него утилиту для создания Python-проекта действия.

Если закрыли консоль с виртуальным окружением после того, как bat-файл закончил свою работу, нужно перейти по ярлыку PythonSDK и запустить файл **activate.bat**.

Все дальнейшие команды нужно выполнять именно в той консоли, в которой активировано виртуальное окружение.

2. Убедиться, что в cmd указан путь %USERPROFILE%\AppData\Local\Programs\Robin Platform\Robin SDK\2.0.0\PythonSDK.



3. Чтобы вызвать утилиту создания Python-проекта, выполните команду:

```
actionsdk
```

4. Для создания Python-проекта выполните команду:

```
actionsdk create-action
```

После выполнения команды, утилита запросит информацию о проекте действия (имена компании, группы и действия должны соответствовать контракту действия, который указывали при создании проекта действия через ActionEditor). Пример заполнения:

- Название компании*: *Action company name: Robin*
- Название группы*: *Action group name: CustomDirs*
- Название действия*: *Action name: CreateSomeDirs*

- Версия реализации*: *Action version [1.0.0]: 1.5.0*
- ФИО создателя действия, желательно указывать на английском*: Action's author full name []: Golovleva Anna*
- Почта создателя действия*: Action's author email []: cool_email@email.ru*
- Описание создаваемого действия*: Action description []: Действие, которое создает папку со специальной структурой подпапок.*

Пункты, отмеченные * обязательны для заполнения, все остальные можно оставить пустыми.

По итогу будут созданы 2 директории:

- Python-проект действия: …**PythonSDK\Actions\<Контракт действия>**
- Python-
  проект для тестирования действия: …**PythonSDK\tests\<Контракт действия>**

Создание Реализация действия

Особенности кода действия

Класс действия должен наследоваться от базового класса действий (*BaseRobinAction*). Для этого нужно:

1. Импортировать из ActionSDK базовый класс действия и декоратор для валидации параметров действия:

```
from ActionSDK import BaseRobinAction
from ActionSDK.Utils import check_parameters
```

2. Переопределить метод *run_action()*
   - Получить параметры для работы экшена из словаря *self.parameters*
   - Реализовать логику требуемую от действия.
3. Обернуть метод *run_action()* в декоратор *check_parameters*, если требуется сделать проверку входных параметров по "типу" и "обязательности"/"не обязятельности" (обязательность входных параметров смотреть в определении, которое настраивали в ActionEditor).

```
@check_parameters(DIALOG_TITLE=(False, (str, int)), QUESTION_TEXT=(True, str))
      DIALOG_TITLE       Тип: str или int;   Параметр не обязателен для работы экшена.
Ожидаемый тип параметра: str или int

      QUESTION_TEXT      Тип: str;          Параметр не обязателен для работы экшена.
Ожидаемый тип параметра: str
```

Возвращаемое значение действия должно быть словарём следующей структуры:

```
result = dict(
      result_1_name=result_1_value,
      result_2_name=result_2_value,
)
```

Сервисы доступные действию

Движок предоставляет создателю действия несколько сервисов:

1. **Логирование.** Для логирования действия нужно использовать метод *self.logger*. Создателю действия доступно 4 уровня логирования:
   - debug
   - info
   - warning
   - error

   Пример использования логирования:

```
self.logger.error("Сообщение об ошибке в действии")
self.logger.debug("Другое логируемое сообщение")
```

2. **Конвертация Robin-типов** в человекочитаемый формат. Для получения строкового представления Robin-типа или нативного типа Python нужно использовать методом *self.get_display_value***.**

   Пример использования конвертации типов:

```
self.get_display_value(robin_type_object)
```

Типы параметров и результатов можно взять из Таблицы типов в Приложении №1.

Реализация действия

Если мы перейдём в папку Python-проекта действия **%USERPROFILE%\AppData\Local\Programs\Robin Platform\Robin SDK\2.0.0\PythonSDK\Actions\<Контракт действия>\<Название действия>**, то увидим 3 файла:

- *__init__.py* - специальный Python-файл;
- *action.py* - Python-файл, в котором мы будем писать логику нашего действия;
- *<Контракт действия>-Python.robin-impinfo* – файл с описанием реализации действия.

В качестве примера, на обучении будет рассматриваться действие, которое должно:

по пути, указанному во входных параметрах, создать новую папку с названием, которое также указано в параметрах; в созданной папке создать подпапку «XXXX», где XXXX –

текущий год; в папке «XXXX» создать папку «XX. Месяц», где XX – номер текущего месяца, Месяц – наименование текущего месяца на русском языке; в папке «XX. Месяц» создать папку «dd.MM.yyyy», где dd – текущий день, MM – текущий месяц, yyyy – текущий год.

Открыть файл **action.py** и отредактировать код действия. Код, используемый для создания действия из примера:

```
from datetime import datetime
from pathlib import Path

from ActionSDK import BaseRobinAction
from ActionSDK.Utils import check_parameters
from EngineActionInterface.RobinExceptions import ActionException

from RobinFolderPath.RobinFolderPath import RobinFolderPath

MONTHS = {
    1: "Январь",
    2: "Февраль",
    3: "Март",
    4: "Апрель",
    5: "Май",
    6: "Июнь",
    7: "Июль",
    8: "Август",
    9: "Сентябрь",
    10: "Октябрь",
    11: "Ноябрь",
    12: "Декабрь",
}

class CreateSomeDirs(BaseRobinAction):
    @check_parameters(parentDirPath=(True, RobinFolderPath), dirName=(True, str))
    def run_action(self) -> dict:
        parent_dir_path = self.parameters.get('parentDirPath')
        dir_name = self.parameters.get('dirName')
        return self.__some_action_logic(parent_dir_path, dir_name)
```

```
def __some_action_logic(self, parent_dir_path: RobinFolderPath, dir_name: str) -> dict:
    dir_path = Path(parent_dir_path.get_valueOf_()).absolute().joinpath(dir_name)
    today = datetime.now()
    dir_path = dir_path.joinpath(str(today.year))
    dir_path = dir_path.joinpath(f"{today.month}. {MONTHS[today.month]}")
    dir_path = dir_path.joinpath(str(today.strftime('%d.%m.%Y')))
    dir_path.mkdir(parents=True, exist_ok=True)
    root_dir_path = RobinFolderPath(valueOf_=str(dir_path))
    return dict(
        result=root_dir_path
    )
```

Тестирование действия

1. Для тестирования написанного кода действия необходимо перейти в Python-проект с тестами действия **%USERPROFILE%\AppData\Local\Programs\Robin Platform \Robin SDK\2.0.0\PythonSDK \tests\<Контракт действия>**.
2. Открыть файл **tests.py** и отредактировать его для проведения теста действия.
3. Вернуться в окно консоли с настроенным виртуальным окружением, перейти в корень Python-проекта с тестами (**…PythonSDK\tests**) и запустить тесты командой:

```
python runtests.py
```

Если всё было сделано правильно, то в соответствии с описанием задачи действия появится папка, содержащая внутри несколько вложенных подпапок.

Модификация файла *.robin-impinfo

Если в действии были использованы Robin-исключения, то в файле **<Контракт действия>-Python.robin-impinfo** нужно указать id данных исключений.

Наименование исключения	**Название**	**Описание**	**Дополнительная информация, выводимая с исключением**
Robin.Exception.Any	Любая ошибка	Возникает, если в действии произошла любая нестандартная исключительная ситуация	-
Robin.Exception.BrowserNotStarted	Браузер не запустился	Возникает, когда системе не удается запустить	Возникает, если на странице не найден веб-элемент (веб-

Наименование исключения	Название	Описание	Дополнительная информация, выводимая с исключением
		браузере с указанными параметрами	элементы) по идентификатору
Robin.Exception.CellEmpty	Ячейка пустая	Возникает если запрашиваемая ячейка в excel документе пуста	Адрес ячейки, из которой будет взято значение
Robin.Exception.CellNotFound	Ячейка не найдена	Возникает если запрашиваемая ячейка в таблице не найдена	Адрес ячейки, к которой обращается действие.
Robin.Exception.DirectoryNotAvailable	Папка недоступна	Возникает когда системе не удается получить доступ папке по указанному пути	Путь к папке, к которой нет доступа
Robin.Exception.DirectoryNotFound	Папка не найдена	Возникает когда системе не удается найти папку в файловой системе по указанному пути	Путь, по которому не была найдена папка
Robin.Exception.ElementNotFound	Веб-элемент не найден	Возникает, если на странице не найден веб-элемент (веб-элементы) по идентификатору	Элемент для взаимодействия
Robin.Exception.ElementNotReachable	Веб-элемент вне доступа	Возникает, если веб-элемент невидим или заблокирован	Элемент для взаимодействия
Robin.Exception.FileNotAvailable	Файл недоступен	Возникает когда системе не удается получить доступ к файлу по указанному пути	Путь к файлу, к которому нет доступа
Robin.Exception.FileNotFound	Файл не найден	Возникает когда системе не удается найти файл в файловой системе по указанному пути	Путь, по которому не был найден файл
Robin.Exception.PageNotFound	Страница не найдена	Возникает когда браузер не может перейти по запрашиваемому адресу	Адрес страницы

Наименование исключения	Название	Описание	Дополнительная информация, выводимая с исключением
Robin.Exception.ParameterNotFound	Параметр не найден	Возникает, если обязательный параметр не найден в списке передаваемых	Контракт действия Наименование обязательного параметра Идентификатор действия Название действия Номер действия
Robin.Exception.TabNotFound	Вкладка не найдена	Возникает когда системе не удается найти в браузере вкладку с указанным заголовком	Название вкладки
Robin.Exception.TimeExceeded	Время истекло	Возникает, если истекло время выполнения команды	-
Robin.Exception.UnconvertibleFormat	Неконвертируемый формат	Неконвертируемый формат	-
Robin.Exception.Unknown	Неизвестная ошибка	Возникает, если в действии произошла нестандартная исключительная ситуация	-
Robin.Exception.WinElementNotFound	Вин-элемент не найден	Возникает, если не найден вин-элемент	Элемент для взаимодействия
Robin.Exception.WinTimeExceeded	Время истекло (win)	Время ожидания истекло, окно или элемент не найдены	-
Robin.Exception.WindowNotFound	Окно не найдено	Возникает, если не окно не было найдено	-

Обфускация действия

Работает, начиная с версии ROBIN Studio 2.7.3.

Для того, чтобы скрыть исходники действия, можно обфусцировать его. Для обфускации действий требуется Windows 10. Более старые версии Windows не поддерживаются.

1. Для обфускации действия, необходимо перейти в консоли в директорию с действием **%USERPROFILE%\AppData\Local\Programs\Robin Platform\Robin SDK\2.0.0\PythonSDK\Actions\<Контракт действия>\<Название действия>** и выполнить следующую команду:

```
python -m nuitka --module action.py
```

   В процессе обфускации, библиотека nuitka может запросить согласие на загрузку доп. пакетов. Нужно нажать Да/Yes на все подобные запросы. Загрузка дополнительных пакетов выполняется только в том случае, если библиотека nuitka используется на данном компьютере впервые.

   После выполнения команды выше, появится 2 дополнительных файла с расширениями **pyd** и **pyi**. Обфусцированный код действия находится в файле с расширением **pyd**.

2. Создать точку входа для действия. Точкой входа служит файл **cli.py**. Он должен быть создан со следующим содержимым:

```
from <Название действия>.action import <Название действия>
```

3. Удалить файл **action.py**.

4. Если необходимо обфусцировать другие Python-модули, то алгоритм их обфускации аналогичный. Но единственным исключением является то, что файл **cli.py** для этих файлов создавать не нужно.

5. После обфускации действия, для того чтобы убедиться, что оно по-прежнему работает, нужно запустить повторно тесты.

6. Для того, чтобы файлы **pyd** и **pyi** вошли в пакет дистрибуции, рядом с файлом **setup.py** нужно создать файл **MANIFEST.in**. Содержимое файла **MANIFEST.in**:

```
global-include *.pyd
global-include *.pyi
```

Сборка пакета реализации действия

1. В папке **%USERPROFILE%\AppData\Local\Programs\Robin Platform\Robin SDK\2.0.0\PythonSDK\Actions\<Контракт действия>** лежит файл **setup.py**. Это файл, в котором прописана инструкция для сборщика пакетов. В этом файле необходимо указать все использованные пакеты с транзитивными зависимостями.

2. Для сборки реализации действия в пакет, в консоли надо перейти в папку **%USERPROFILE%\AppData\Local\Programs\Robin Platform\Robin SDK\2.0.0\PythonSDK\Actions\<Контракт действия>** и выполнить следующую команду:

```
python setup.py bdist_wheel
```

3. После успешного выполнения команды рядом с файлом **setup.py** появится папка **dist**, в которой и будет пакет с реализацией действия: **<Контракт действия>_Python-1.5.0-py3-none-any.whl**).
4. Перейти через консоль в директорию, в которую надо будет положить пакеты с реализацией **%HOMEPATH%\Documents\Robin\Robin Actions\<Контракт действия>\<Контракт действия>\realization\platform\python\1.5.0**. Изначально в папке **%HOMEPATH%\Documents\Robin\Robin Actions\<Контракт действия>\<Контракт действия>\realization\platform\python** будет лежать подпапка, в названии которой будет указана не версия реализации, а версия определения, которую указывали в ActionEditor при создании проекта действия. Поэтому необходимо вручную изменить название этой папки на версию реализации.
5. Выполнить команды по установке переменных:

set DEPENDENCIES_DIR=%DEPENDENCIES_DIR%

где %DEPENDENCIES_DIR% - это путь до директории с зависимостями PythonSDK. По умолчанию: **%localappdata%\Programs\Robin Platform\Robin SDK\2.0.0\PythonSDK\dependencies**

set ACTION_PACKAGE_PATH=%ACTION_PACKAGE_PATH%

где %ACTION_PACKAGE_PATH% - это путь до директории, в которой лежит созданный нами пакет действия: **%localappdata%\Programs\Robin Platform\Robin SDK\2.0.0\PythonSDK\Actions\<Контракт действия>\dist\<Контракт действия>_Python-1.5.0-py3-none-any.whl**

6. Скопировать Python-пакеты командой:

python -m pip download --no-index -f "%DEPENDENCIES_DIR%" -d transitiveDependencies "%ACTION_PACKAGE_PATH%"

7. Установить Python-пакеты командой:

python -m pip install --no-index -f "%DEPENDENCIES_DIR%" -t actionRealization "%ACTION_PACKAGE_PATH%"

8. Перейти в консоли в директорию **«…realization\platform\python\<Версия реализации>\actionRealization**» и запаковать в zip-архив (БЕЗ СЖАТИЯ) всё

содержимое данной директории. Переименовать получившийся архив на **<Контракт действия>.robin-python-package**. У архива должно поменяться расширение с «*.zip*» на «*.robin-python-package*».

9. Удалить все папки и файлы в **«…realization\platform\python\<Версия реализации>\actionRealization**» кроме архива, который только что создали.

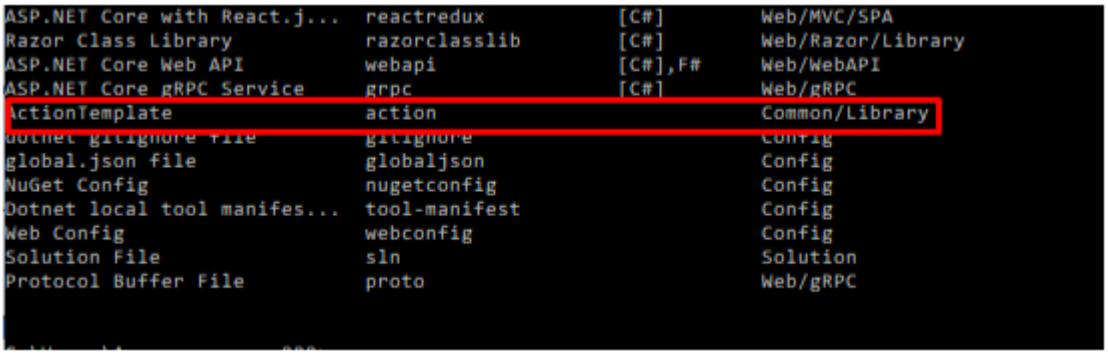
Создание .Net-действия

Создание Net-проекта действия

Перед созданием net-проекта требуется установить пакет с шаблоном действия, для этого нужно в Командной строке выполнить:

```
dotnet new -i "%localappdata%\Programs\Robin Platform\Robin SDK\2.0.0\TemplatePack\Internal.Build.ActionTemplate.1.0.1.nupkg"
```

После этого в списке шаблонов появится новый шаблон:



Для создания net-проекта нужно:

- создать папку, в которой будет создан проект (название и директория не имеют значение);
- выполнить команду:

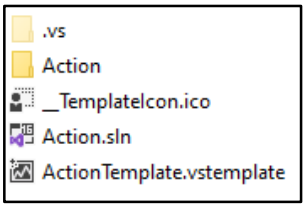
```
dotnet new action --force -o "путь до папки, в которой будет создан проект" --company "пространство" --group "имя группы, к которой принадлежит действие" --action "название действия" --actionVersion "версия действия"
```

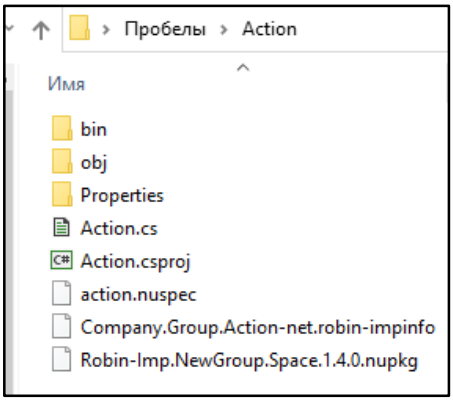
Все параметры команды должны быть написаны латинскими буквами и символами и будут использованы для создания контракта в формате *company.group.action.version*. Это должны быть те же значения, которые указывали при создании определения нового действия.

Пример заполненной команды:

dotnet new action --force -o "C:\Users\AGolovleva\Desktop\Пробелы" --company "Robin" --group "NewGroup" --action "Space" --actionVersion "1.4.0".

После выполнения в указанной директории должны появиться:



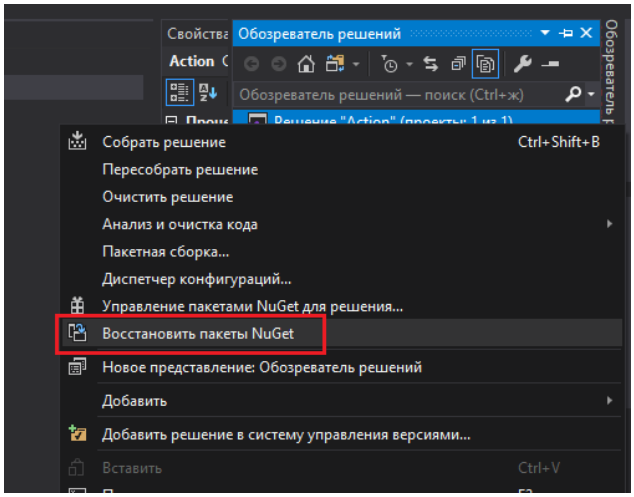


Указанные при создании net-проекта параметры были использованы для генерации содержимого в файлах.

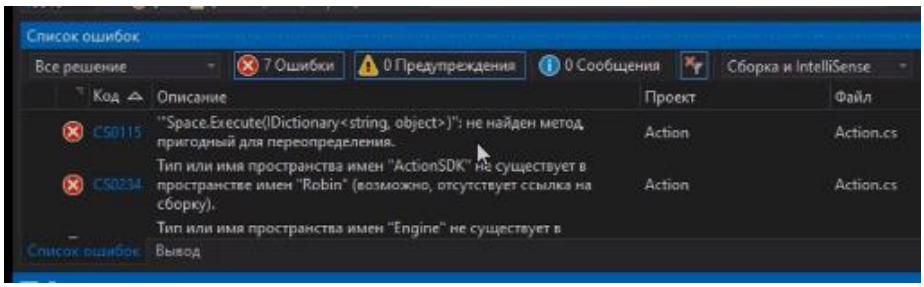
Написание реализации

Для примера будет приведено создание действие по удалению пробелов с начала и конца строкового значения.

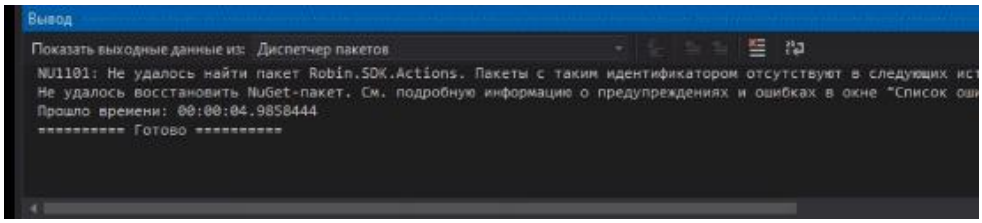
1. Открыть файл **Action.csproj**.
2. Перед редактированием кода необходимо восстановить пакеты Nuget, используемые в проекте. Для этого необходимо нажать в обозревателе решений по строке «*Решение "Action"*» и выбрать в меню «*Восстановить пакеты NuGet*».



3. Если пакеты были восстановлены успешно, то не будет ни ошибок, ни предупреждений.
4. Если возникают такие ошибки:

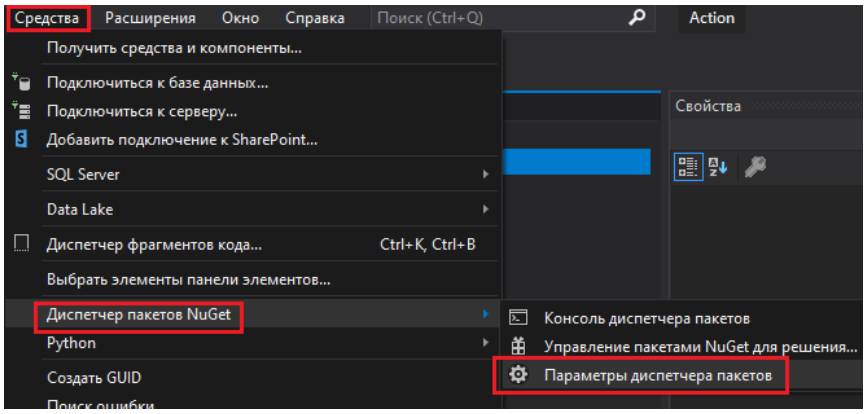


Или



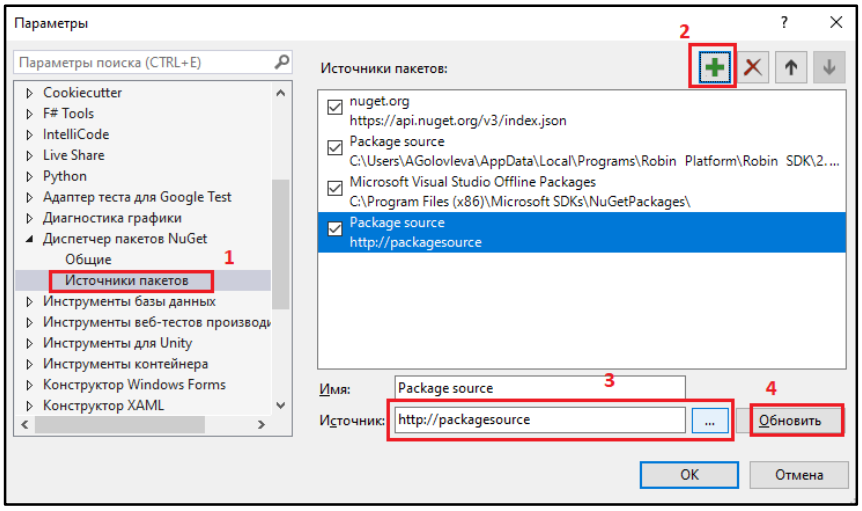
Необходимо сделать следующее:

4.1.Нажать «*Средства*» - «*Диспетчер пакетов NuGet*» - «*Параметры диспетчера пакетов*».



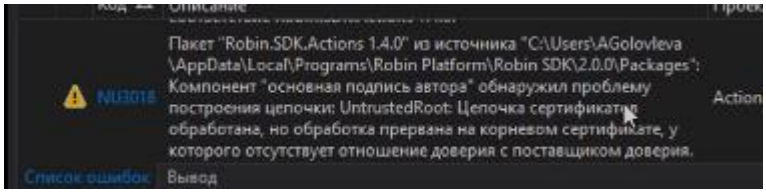
4.2. В открывшемся окне «*Параметры*» выбрать «*Источники пакетов*».

4.3. Добавить новый источник пакетов. В качестве источника указать путь до папки с пакетами **%localappdata%\Programs\Robin Platform\Robin SDK\2.0.0\Packages**. Нажать «*Обновить*» и «*Ок*».



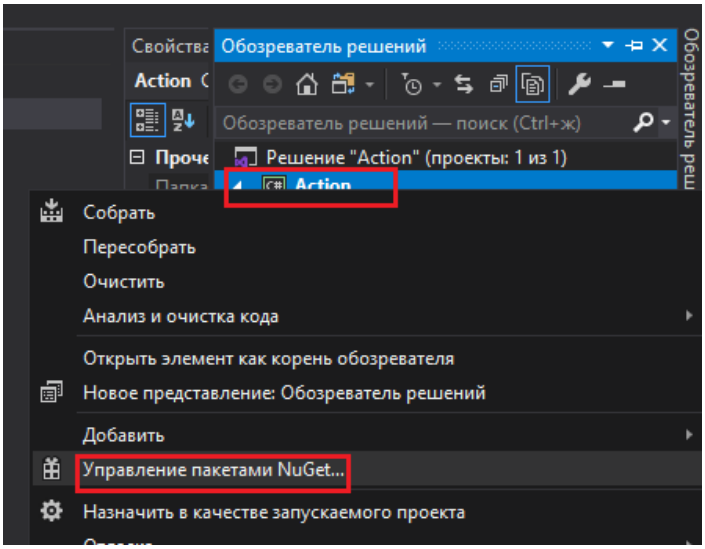
5. Повторить шаг 2.

6. Если возникает такое предупреждение:



Необходимо выполнить следующее:

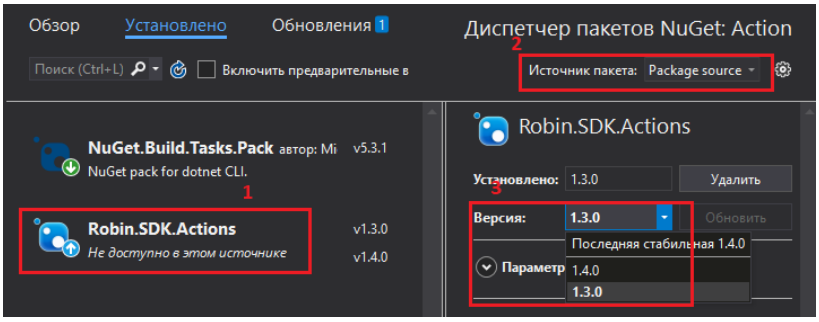
6.1. Нажать в обозревателе решений по строке «Action» и выбрать в меню «Управление пакетами NuGet».



6.2. Выбрать «*Robin.SDK.Action*».

6.3. Выбрать в качестве источника пакетов именно тот, который добавляли на шаге 4.

6.4. В поле «*Версия*» указать версию реализации действия.



6.5. Нажать «*Обновить*»

7. Перейти к редактированию кода действия. Исходный код реализации, полученный после генерации проекта, выглядит следующим образом:

```
using Robin.ActionSDK;
using Robin.Engine.Services.Interfaces;
using System.Collections.Generic;
using System;

namespace Robin.NewGroup
{
```

```
    public class Space : BaseRobinAction
      {
                public Space(IActionLogger logger) : base(logger) { }

                public override IDictionary<string, object> Execute(IDictionary<string, object> parameters)
                {
                        return null;
                }
    }

}
```

Для действий, где несколько действий работают с общими объектами, используются Nuget-пакеты с определением контекстов для их дальнейшей передачи действиям. Такие пакеты, как правило, имеют наименование формата **robin.%group%.commonclasses**.

При разработке действий, работающих с контекстами издателя, необходимо добавить в проект соответствующий пакет. Примеры пакетов указаны ниже:

- robin.xml.commonclasses 1.0.0
- robin.web.commonclasses 2.0.0
- robin.ftp.commonclasses 2.0.5
- и др.

Найти необходимые пакеты можно по стандартному пути **%USERPROFILE%\AppData\Local\Programs\Robin Platform 2.0.0\Agent\Packages\Nuget**

Если тип переменной, используемый в действии, не является нативным типом языка разработки, то необходимо включить в проект пакет с определением этого типа. Типы параметров и результатов можно взять из Таблицы типов в Приложении №1. Примеры пакетов, содержащих определение таких типов, указаны ниже:

- robin.type.datatable 2.0.0
- robin.type.datetime 2.0.0
- robin.type.dictionary 2.0.0
- robin.type.filepath 2.0.0
- robin.type.folderpath 2.0.0
- и т.д.

Найти необходимые пакеты можно по стандартному пути **%USERPROFILE%\AppData\Local\Programs\Robin Platform\Robin SDK\2.0.0\Packages\Nuget**.

Если в указанной директории пакет для нужного типа не был найден, можно найти его по пути **%USERPROFILE%\AppData\Local\Programs\Robin Platform\2.0.0\Agent\Packages\Nuget.**

Класс *BaseRobinAction* - абстрактный класс для действия, от которого необходимо наследовать свое действие при разработке, находящийся в пространстве имен *Robin.ActionSDK*.

Интерфейс *IActionLogger* содержит интерфейс для работы с классами логирования. Интерфейс находится в пространстве имен *Robin.Engine.Services.Interfaces*.

В конструкторе текущего класса необходимо указать все необходимые для работы сервисы, например, для работы с ресурсами необходимо указать в аргументах *IResourcesFolderService*.

При выполнении действия роботом вызывается метод *Execute*, на вход которому передается *Dictionary* с параметрами, указанными в действии, результаты действия возвращаются как результат выполнения метода *Execute*, при его отсутствии возвращается null. Если действие должно возвращать результат, то необходимо написать следующий фрагмент кода:

```
IDictionary<string, object> result = new Dictionary<string, object>
{
   {"Result", resultValue}
};
return result;
```

Пример отредактированного кода действия:

```
using Robin.ActionSDK;
using Robin.Engine.Services.Interfaces;
using System.Collections.Generic;
using System;
namespace Robin.NewGroup
{
   public class Space : BaseRobinAction
          {
                public Space(IActionLogger logger) : base(logger) { }
                public override IDictionary<string, object> Execute(IDictionary<string, object> parameters)
                {
                      var text = (string)parameters["Text"];
                      text = text.Trim();
                      return new Dictionary<string, object>(){
                            { "Result", text }
                      };
                }
   }

}
```

Добавление логирования

В конструкторе класса действия на вход был получен *IActionLogger logger*. Мы можем обратиться к сервису логирования через наследуемый атрибут *Logger*. Сервис логирования имеет несколько уровней:

- Debug
- Info
- Warning
- Error

При указании у действия нужного уровня логирования в ROBIN Studio, при выполнении этого действия будут отображены логи соответствующего уровня. Для создания логов в исходном коде необходимо вызвать:

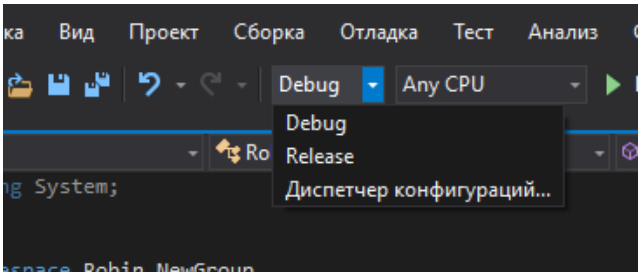
Logger.<Level>(<message>);

Пример:

```
Logger.Error("Возникла ошибка при работе действия");

Logger.Debug("Первый этап действия завершился успешно");
```

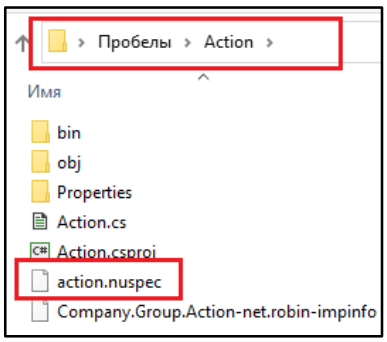
8. После завершения редактирования и проверки кода, нажать «*Release*».



9. Выбрать «*Сборка*» - «*Собрать решение*». Открытую среду разработки на этом можно закрывать.

Сборка Net-проекта

1. Создать файл **action.nuspec** в подпапке «*Action*» net-проекта. Это манифест, содержащий метаданные пакета решения. Описывает содержимое пакета.



Пример содержимого файла:

```
<?xml version="1.0" encoding="utf-8" ?>
<package>
 <metadata>
  <id>Robin-Imp.NewGroup.Space</id>
  <version>1.4.0</version>
  <title>Robin-Imp.NewGroup.Space</title>
  <authors>Robin</authors>
  <owners>Robin</owners>
  <description>Robin-Imp.NewGroup.Space</description>
  <dependencies>
          <group targetFramework="net48">
   <dependency id="Robin.Sdk.Action" version="1.4.0" />
          </group>
  </dependencies>
          <frameworkAssemblies>
   <frameworkAssembly assemblyName="System" targetFramework="net48"/>
   <frameworkAssembly assemblyName="System.Core" targetFramework="net48"/>
   <frameworkAssembly assemblyName="System.Xml.Linq" targetFramework="net48"/>
   <frameworkAssembly assemblyName="System.Data.DataSetExtensions" targetFramework="net48"/>
   <frameworkAssembly assemblyName="Microsoft.CSharp" targetFramework="net48"/>
   <frameworkAssembly assemblyName="System.Data" targetFramework="net48"/>
   <frameworkAssembly assemblyName="System.Net.Http" targetFramework="net48"/>
   <frameworkAssembly assemblyName="System.Xml" targetFramework="net48"/>
  </frameworkAssemblies>
  <summary> A summary of your package </summary>
 </metadata>
 <files>
  <file src="Company.Group.Action-net.robin-impinfo" />
          <file src="bin\Release\Robin-Imp.NewGroup.Space.dll" target="lib\net48" />
 </files>
</package>
```

2. Скачать **nuget.exe** (интерфейс командной строки nuget):
3. Прописать путь до папки, в которой расположен **nuget.exe**, в переменных среды.
4. Перейти в Командной строке в подпапку «*Action*» net-проекта.
5. Выполнить команду:

```
nuget pack action.nuspec
```

```
C:\Users\AGolovleva>cd C:\Users\AGolovleva\Desktop\Пробелы\Action

C:\Users\AGolovleva\Desktop\Пробелы\Action>nuget pack action.nuspec
Попытка выполнить сборку пакета из "action.nuspec".
Successfully created package 'C:\Users\AGolovleva\Desktop\Пробелы\Action\Robin-Imp.NewGroup.Space.1.4.0.nupkg'.

C:\Users\AGolovleva\Desktop\Пробелы\Action>
```

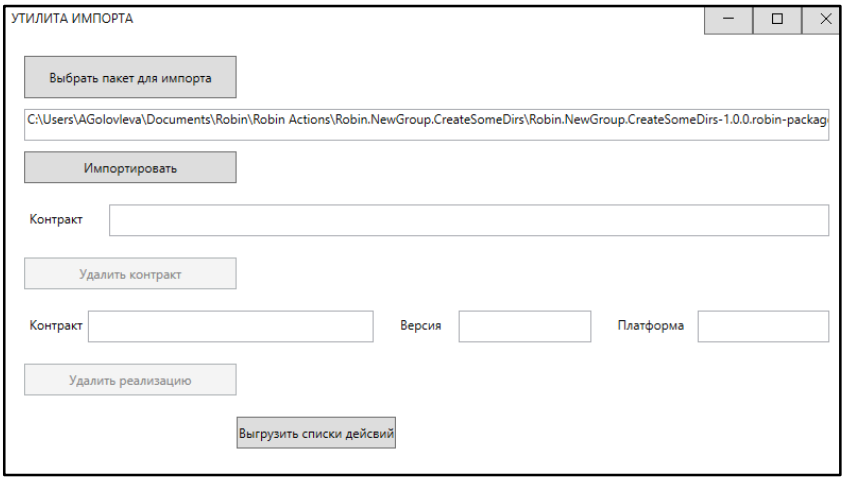
6. В папке **…\Action** сформируется пакет с реализацией net-действия - файл ***.nupkg**, который необходимо перенести в папку **%HOMEPATH%\Documents\Robin\Robin Actions\<Контракт действия>\<Контракт действия>\realization\platform\net\<Версия реализации>\actionRealization**
7. Изначально в папке **%HOMEPATH%\Documents\Robin\Robin Actions\<Контракт действия>\<Контракт действия>\realization\platform\net** будет лежать подпапка, в названии которой будет указана не версия реализации, а версия определения, которую указывали в ActionEditor при создании проекта действия. Поэтому необходимо вручную изменить название этой папки на версию реализации.
8. Положить все использующиеся в действии пакеты с транзитивными зависимостями в папку **%HOMEPATH%\Documents\Robin\Robin Actions\<Контракт действия>\<Контракт действия>\realization\platform\net\<Версия реализации>\transitiveDependencies.**

Сборка готового пакета действия

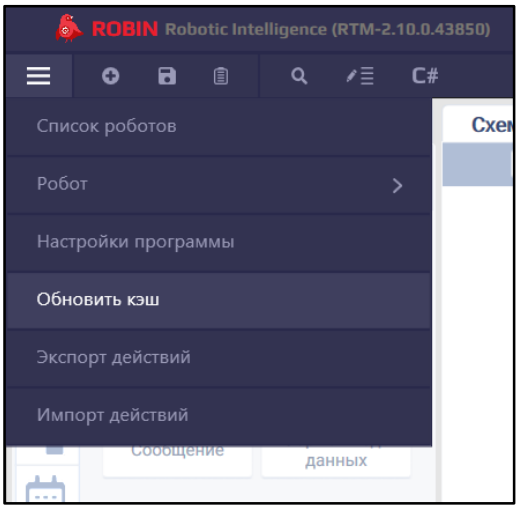
1. Снова запустить ActionEditor.
2. Открыть проект действия.
3. Выбирать «*Файл*» - «*Сохранить как пакет*».

Импорт пакета действия в ROBIN Studio

1. Запустить **ActionLoader.**
2. Через кнопку «*Выбрать пакет для импорта*» выбирать файл пакета действия. Лежит по пути **%USERPROFILE%\Documents\Robin\Robin Actions\<Контракт действия>\<Контракт действия>-1.0.0.robin-package.**



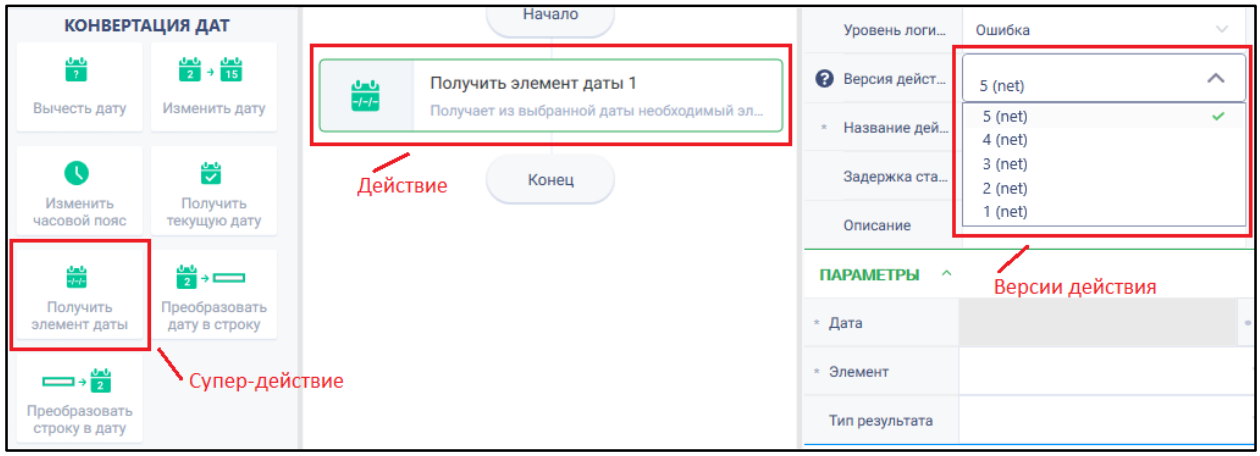
3. Закрыть ROBIN Studio, нажать на кнопку «*Импортировать*» и ждать информационное окошко о завершении импорта действия.
4. Заново открыть ROBIN Studio и открыть любой проект (схему).
5. Нажать «Обновить кэш» в общем меню ROBIN Studio:



6. Проверить наличие и работоспособность нового действия.

Добавление новой версии действия

Супер-действие может содержать несколько различных версий определения действия. Различные версии определения одного и того же действия отличаются друг от друга не только свойствами (название, количество полей входных данных и результата), но и контрактом. При этом к одному и тому же контракту (т.е. фактически определению действия) может относится несколько версий реализации действия. Например, когда при смене версий действия в ROBIN Studio его свойства не изменяются, но на каждой из выбранных версий действие отрабатывает по-разному (на одной версии – ошибка, а на другой - нет).



При рассмотрении добавления новой версии действия здесь учитываем, что в ROBIN Studio уже имеется супер-действие, содержащее как минимум одну версию действия.

Добавление новой версии реализации действия (без изменения определения)

1. Открыть файл **ListActions.xml**, расположенный в папке с проектом действия.
2. Добавить после последнего элемента **<ActionContract>** еще один такой же элемент (по отношению друг к другу элементы будут соседними, иметь один и тот же родительский элемент):

   *<ActionContract deprecated="false" contract="**Контракт действия**" impVersion="**Версия реализации**" superActionVersion="**Пользовательская версия действия**" execType="**тип платформы реализации**" />*

   - *Контракт действия* – необходимо указать контракт того определения, с которым должна быть связана эта реализация.
   - *Версия реализации* – указать версию на номер выше. Пример: предыдущая версия реализации – «1.5.0», значит теперь необходимо поставить «1.5.1».
   - *Пользовательская версия действия* – версия, которая будет отображаться в ROBIN Studio для пользователя.
   - *Тип платформы реализации* – net или python, у действия с одним и тем же определением могут быть различные версии реализации, написанные на различных языках программирования.

3. Сохранить документ.
4. Открыть ActionEditor.
5. В основном меню выбрать файл ***.action-project**, нажав на кнопку «*Открыть действие*». Файл расположен в папке с проектом действия. Т.е. нужно открыть старый проект того действия, которое обновляем.
6. Проверить, что на вкладке «*Настройки робота»* в поле «*Контракт»* в выпадающем списке появилась новая строка, в которой указан один из старых контрактов действия, но с новой версией реализации (той, которую указали в **ListActions.xml**). Выбрать эту новую строку для поля «*Контракт*».
7. Сохранить изменения: *«Файл»* - *«Сохранить определение действия».*
8. Повторить этап «Создание Python-действия» или «Создание Net-действия», указывая при этом везде соответствующую версию реализации и название контракта. Только теперь в папке **«…\realization\platform\python**» или **«…\realization\platform\net**» (в зависимости от типа платформы новой реализации) создаем еще одну папку:
    - название – новая версия реализации;
    - добавить в созданную папку пустые подпапки:
        - actionRealization
        - sourceCode
        - transitiveDependencies.

    Пакеты Python-проекта действия или Net-проекта действия с новой реализацией необходимо устанавливать именно в этих новых подпапках.
9. В ActionEditor нажать «Файл» - «Сохранить как пакет».
10. Повторить этап «Импорт пакта действия в ROBIN Studio».

Добавление новой версии определения действия

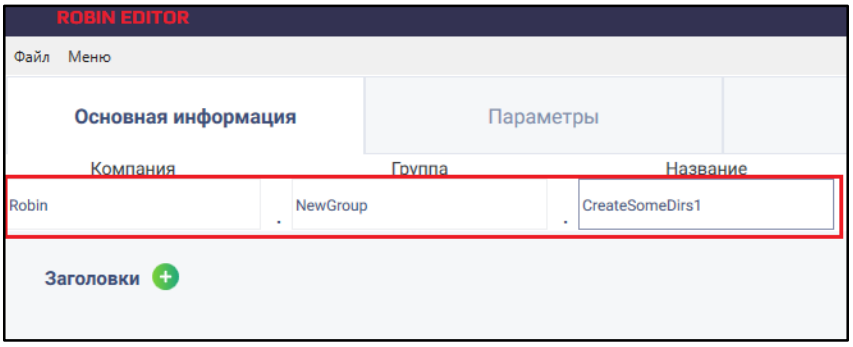
При изменении определения действия, соответственно, придется менять и реализацию действия (чтобы учесть в коде действия эти изменения). Простое изменения, например, описания для подсказок, с сохранением тех же входных параметров и результатов действия, здесь не рассматривается.

1. Открыть файл **ListActions.xml**, расположенный в папке с проектом действия.
2. Добавить после последнего элемента **<ActionContract>** еще один такой же элемент (по отношению друг к другу элементы будут соседними, иметь один и тот же родительский элемент):

    *<ActionContract deprecated="false" contract="**Контракт действия**" impVersion="**Версия реализации**" superActionVersion="**Пользовательская версия действия**" execType="**тип платформы реализации**" />*

- *Контракт действия* – необходимо указать новый контракт. Пример: контракт предыдущего определения – «Robin.NewGroup.CreateSomeDirs», новый контракт – «Robin.NewGroup.CreateSomeDirs**1».**
- *Версия реализации* – указать версию на номер выше. Пример: предыдущая версия реализации – «1.5.0», значит теперь необходимо поставить «1.5.1».
- *Пользовательская версия действия* – версия, которая будет отображаться в ROBIN Studio для пользователя.
- *Тип платформы реализации* – net или python, у действия с одним и тем же определением могут быть различные версии реализации, написанные на различных языках программирования.

3. Сохранить документ.
4. Открыть ActionEditor.
5. В основном меню выбрать файл ***.action-project**, нажав на кнопку «*Открыть действие*». Файл расположен в папке с проектом действия. Т.е. мы должны открыть старый проект того действия, которое обновляем.
6. Проверяем, что на вкладке «*Настройки робота*» в поле «*Контракт*» в выпадающем списке появилась новая строка с новым контрактом, который указывали в **ListActions.xml**. Выбираем сроку с новой версией контракта в поле «*Контракт*».
7. На вкладке «*Основная информация*» указываем новый контракт.



8. Вносим необходимые изменения на вкладке *«Параметры»* и «*Результаты*».
9. Сохраняем изменения: *«Файл» - «Сохранить определение действия».*

11. Повторить этап «Создание Python-действия» или «Создание Net-действия», указывая при этом везде соответствующую версию реализации и название контракта. Только теперь в папке «**…\realization\platform\python**» или «**…\realization\platform\net**» (в зависимости от типа платформы новой реализации) создаем еще одну папку:
    - название – новая версия реализации;
    - добавить в созданную папку пустые подпапки:
        - actionRealization
        - sourceCode

— transitiveDependencies.

Пакеты Python-проекта действия или Net-проекта действия с новой реализацией необходимо устанавливать именно в этих новых подпапках.

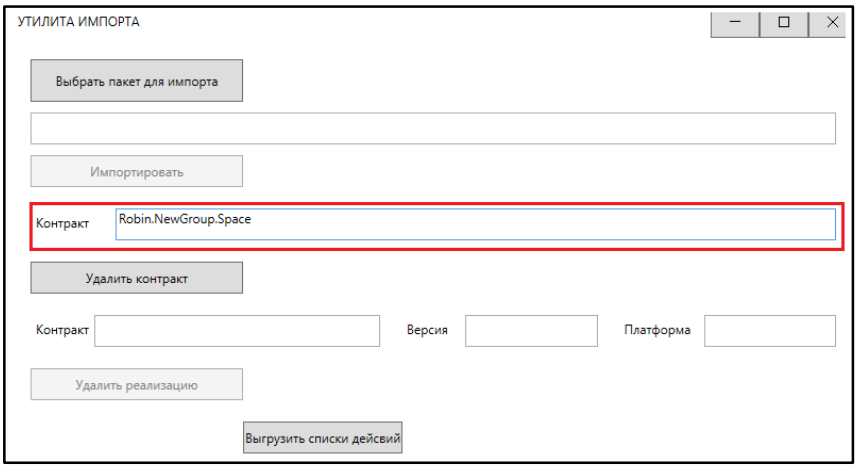
10. В ActionEditor нажать «*Файл*» - «*Сохранить как пакет*».

11. Повторить этап «Импорт пакта действия в ROBIN Studio».

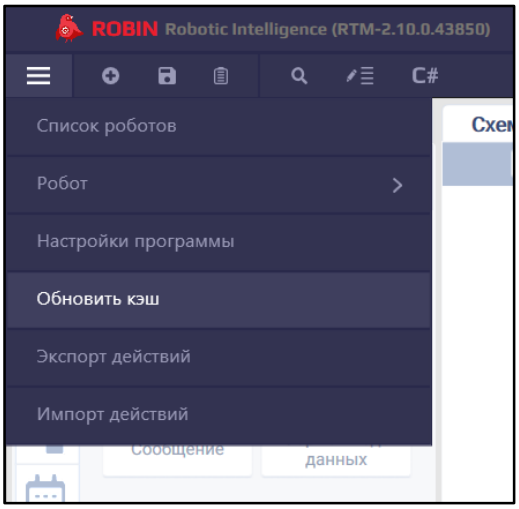
Удаление действия из ROBIN Studio

Удаление версии определения

1. Запустить ActionLoader .
2. Закрыть ROBIN Studio, если была открыта.
3. В верхнем поле *«Контракт»* указать контракт действия. Пример: *Robin.CustomDirs.CreateSomeDirs.*



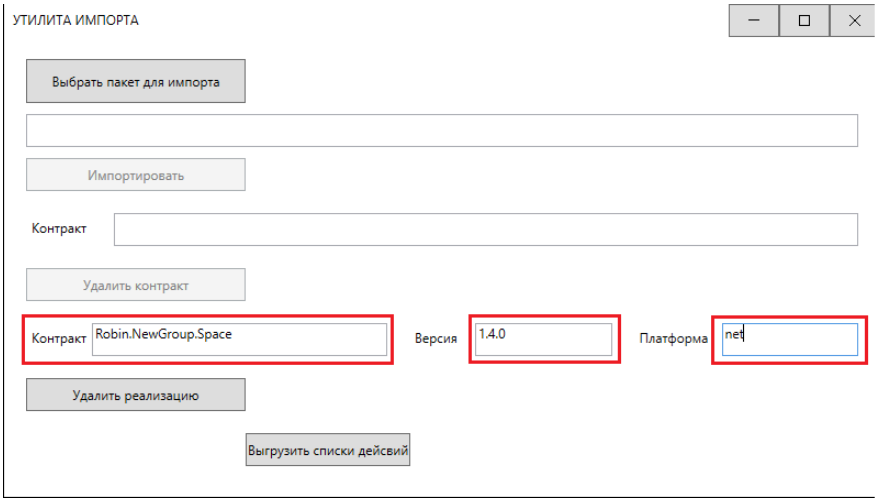
4. Нажать кнопку *«Удалить контракт».* Т.к. в рамках одного контракта действия (одной версии определения) в супер-действии может содержаться несколько различных версий реализации, то при удалении одной версии определения, будут удалены и все версии реализации данного действия, которые относились к контракту этого определения.
5. Заново открыть ROBIN Studio и открыть любой проект (схему).
6. Нажать «Обновить кэш» в общем меню ROBIN Studio:



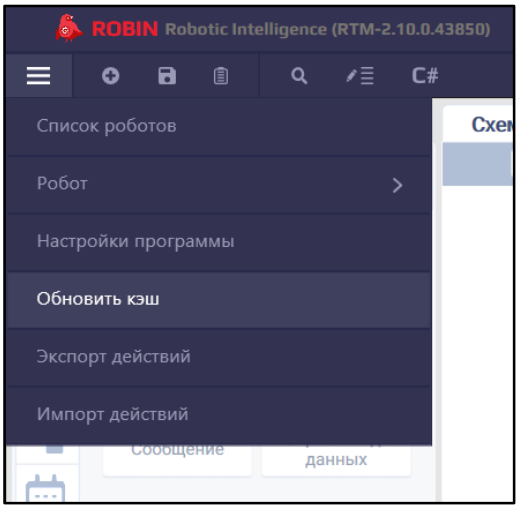
7. проверить отсутствие необходимых версий действия.

Удаление версии реализации

1. Запустить ActionLoader .
2. Закрыть ROBIN Studio, если была открыта.
3. В нижнем поле *«Контракт»* указать контракт действия.
4. В поле *«Версия»* указать версию реализации действия.
5. В поле *«Платформа»* указать название платформы реализации: net или python.



6. Нажать на кнопку «Удалить реализацию». Так из ROBIN Studio удалена только одна из версий действия.
7. Заново открыть ROBIN Studio и открыть любой проект (схему).
8. Нажать «Обновить кэш» в общем меню ROBIN Studio:



9. проверить отсутствие необходимой версии действия.

Приложение 1

Каждая платформа реализации оперирует нативными типами своих данных. Для обеспечения кроссплатформенности и передачи сложных объектов между разыми системными компонентами выполняется приведение нативных типов к кроссплатформенным DTO Robin-типам - типам объектов передаваемых данных.

Ниже представлен список Robin-типов, возможность их сериализации и соответствие нативным типам для каждой платформы. Приведение нативных типов к Robin-типам выполняется с помощью Robin-конверторов.

В колонке "Чем является нативный тип внутри …" есть 3 варианта типов:

1. *Нативный тип*. Пример (net): System.Collections.Generic.List, string и др.

   В этом случае необходимо приводить к тому типу, который указан в столбце. Примеры (net):

   - Для коллекций System.Collections.Generic.List:

     ```
     List<object> Services = (List<object>)parameters["Services"];
     ```

   - Для словарей System.Collections.Generic.Dictionary<string, object>:

     ```
     Dictionary<string, object> options= (Dictionary<string, object>)parameters["Options"];
     ```

2. *DTO*. Пример (net):

   Robin.Type.RobinFilePath

   Nuget

   Robin.Type.FilePath.2.0.0

   В этом случае необходимо сначала добавить указанный Nuget-пакет в проект. Далее выполнить приведение в следующем формате (пример для net):

   - Путь к файлу:

   ```
   var pathToFile = ((RobinFilePath) parameters["Path2File"])
   ```

- Web-элемент:

```
var webElement = ((RobinWebElement) parameters["CoolElementName"])
```

3. *Если указан прочерк*, значит используемый тип – контекст. В этом случае необходимо также сначала добавить указанный Nuget-пакет в проект. Далее выполнить приведение в следующем формате (пример для net):

```
var xmlContext = (XmlContext)parameters["XML_INSTANCE"];
```

Robin-тип	**Сериализуется для передачи между платформами**	**Чем является нативный тип внутри .NET**	**Чем является нативный тип внутри Python**	**Презентационное значение объекта**
Boolean	Да	Нативный тип bool	Нативный тип bool	Получается с помощью toString() Пример: *true*
Collection	Да	Нативный тип System.Collections.Generic.List	Нативный тип list	Записей = N Пример: *Записей = 5*
DataTable	Да	Нативный тип System.Data.DataTable	DTO RobinDataTable.RobinDataTable wheel пакет Robin.Type.DataTable==1.0.0	Столбцов = N, Строк = Y Пример: *Столбцов = 45, Строк = 74*
DateTime	Да	Нативный тип System.DateTime	Нативный тип datetime.datetime	Дата и время в формате dd-MM-yyyy HH:mm:ssXXX Пример:

Robin-тип	Сериализуется для передачи между платформами	Чем является нативный тип внутри .NET	Чем является нативный тип внутри Python	Презентационное значение объекта
				22-07-2020 10:17:34 +03:00
Dictionary	Да	Нативный тип System.Collections.Generic.Dictionary<string, object>	Нативный тип dict	Записей = N Пример: *Записей = 68*
Email	Нет	-	-	
Excel	Нет	-	-	"Экземпляр Excel"
Exception		DTO Robin.SDK.Types.Interfaces.RobinException Nuget Robin.SDK.Actions.1.3.0	Нативный тип EngineActionInterface.RobinExceptions.ActionException wheel пакет Robin.ActionSDK~=1.5.0	Поля структуры ошибки. выведенные через перевод строки Тип: ... Сообщение: ... Источник: ... Дополнительная информация: ключ=значение, ключ=значение Вложенное исключение: есть/нет
FilePath	Да	DTO Robin.Type.RobinFilePath Nuget Robin.Type.FilePath.2.0.0	DTO RobinFilePath.RobinFilePath wheel пакет Robin.Type.FilePath==1.0.0	Строка внутри обёртки FilePath Пример: *C://Temp/1.txt*

Robin-тип	Сериализуется для передачи между платформами	Чем является нативный тип внутри .NET	Чем является нативный тип внутри Python	Презентационное значение объекта
FolderPath	Да	DTO Robin.Type.RobinFolderPath Nuget Robin.Type.RobinFolderPath.2.0.0	DTO RobinFolderPath.RobinFolderPath wheel пакет Robin.Type.FolderPath==1.0.0	Строка внутри обёртки FolderPath Пример: *C://Temp*
FTP	Нет		-	
HotKeys	Да	DTO Robin.Type.RobinHotKeys Nuget Robin.Type.RobinHotKeys.2.0.0	DTO RobinHotKeys.RobinHotKeys wheel пакет Robin.Type.HotKeys==1.0.0	
Image	Да	DTO Robin.Type.RobinImage Nuget Robin.Type.RobinImage.2.0.0	DTO RobinImage.RobinImage wheel пакет Robin.Type.Image==1.0.0	Строка внутри обёртки Image Пример: *C://Temp/1.png*
Internal.SequenceId	Не передаётся между платформами	Нативный тип string	DTO RobinInternalSequenceId.RobinInternalSequenceId	нет

Robin-тип	Сериализуется для передачи между платформами	Чем является нативный тип внутри .NET	Чем является нативный тип внутри Python	Презентационное значение объекта
			wheel пакет Robin.Type.InternalSequenceId==1.0.0	
LocationPoint	Да	Нативный тип System.Drawing.Point	DTO RobinLocationPoint.RobinLocationPoint wheel пакет Robin.Type.LocationPoint==1.0.0	Координаты точки в формате (x, y) Пример: *(24, 32)*
MQ	Нет		-	
Numeric	Да	Нативный тип double	Нативный тип float	Получается с помощью toString() Пример: *28.5*
Object	Нет	Нативный тип object	Нативный тип Object	
Password	Да	DTO Robin.Type.RobinPassword Nuget Robin.Type.RobinPassword.2.0.0	DTO RobinPassword.RobinPassword wheel пакет Robin.Type.Password==1.0.0	

Robin-тип	Сериализуется для передачи между платформами	Чем является нативный тип внутри .NET	Чем является нативный тип внутри Python	Презентационное значение объекта
PDF			-	
RectangleSize	Да	Нативный тип System.Drawing.Size	DTO RobinRectangleSize.RobinRectangleSize wheel пакет Robin.Type.RectangleSize==1.0.0	Размер прямоугольной области в формате W x H Пример: *240 x 320*
Selenium	Нет	Нативный тип Object	-	"Экземпляр Web-драйвера"
SpreadSheets	Нет		-	
String	Да	Нативный тип string	Нативный тип str	Строка и является презентационным значением объекта Пример: *Строка*
WebElement	Да	DTO Robin.Type.RobinWebElement Nuget	DTO RobinWebElement.RobinWebElement wheel пакет Robin.Type.WebElement==1.0.0	Выводится одно из полей структуры. Какое поле выводить указано в поле activeAttribute этой же структуры.

Robin-тип	Сериализуется для передачи между платформами	Чем является нативный тип внутри .NET	Чем является нативный тип внутри Python	Презентационное значение объекта
		Robin.Type.RobinWebElement.2.0.0		
WinElement	Да	DTO Robin.Type.RobinWinElement Nuget Robin.Type.RobinWinElement.2.0.0	DTO RobinWinElement.RobinWinElement wheel пакет Robin.Type.WinElement==1.0.0	
Word	Нет	-	-	"Экземпляр Word"
XML	Нет	-	-	
XmlDoc	Да	Нативный тип System.Xml.XmlDocument	Нативный тип lxml.etree._Element	XML представление xml документа(при презентации пользователю в виде одной строки - обрезается (программой презентующей), но есть возможность посмотреть полный текст.