An overhead view of five people sitting around a dark table in a meeting room. They are looking at documents and laptops. The image is tinted with a dark blue color.

# Автоматизация UiPath

## Пошаговое руководство

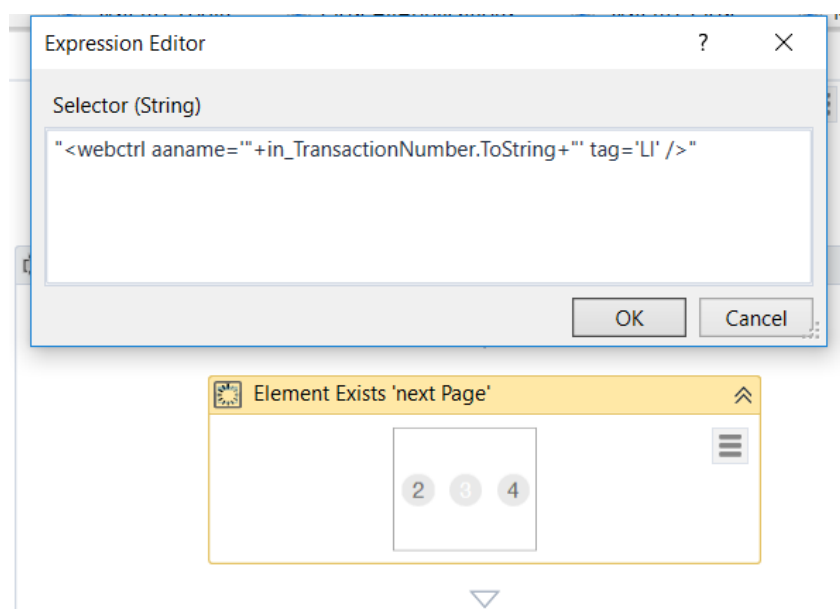
## Пошаговое руководство – Создание ежегодного отчета для поставщика

На этот раз мы используем Orchestrator Queues ("Очереди Orchestrator") для обработки рабочих элементов, чтобы больше узнать о возможностях этой функции. Мы рассмотрим пример того, как использовать нескольких роботов для обработки данных, как предотвратить обработку ими всего списка элементов очереди с самого начала, а вместо этого возобновить работу в случае возникновения системной ошибки и т.д. Мы также разделим процесс при помощи 2 различных процессов. Один создает очередь элементов и называется **Диспетчером** (Dispatcher). Другой обрабатывает ранее созданные элементы очереди и называется **Исполнителем** (Performer). При таком подходе мы можем загружать транзакции с помощью диспетчера только один раз, а затем использовать нескольких роботов-исполнителей для обработки элементов очереди, созданных Диспетчером.

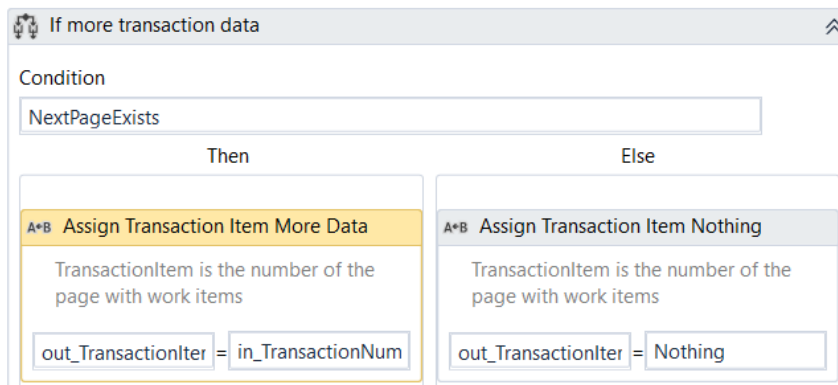
### Процесс Диспетчера

- Начнем с шаблона REFramework.
  - Диспетчер отвечает за загрузку рабочих элементов в очередь. Мы должны загрузить в очередь **WIID**, чтобы присвоить уникальный индекс каждому элементу транзакции.
  - Давайте представим, что стрелка перехода на следующую страницу недоступна для WI 4, поэтому мы не можем считать данные в таблице, чтобы извлечь все рабочие элементы. Более того, если что-то случится с приложением System 1 во время навигации по страницам, Диспетчер восстановится после ошибки и возобновит работу. Он также повторит попытку неудачных транзакций. Мы рассматриваем одну страницу в списке рабочих элементов (WorkItems List) как транзакцию, а номер страницы — как элемент транзакции.
  - Элемент транзакции - это строка, представляющая номер страницы, которая в данный момент обрабатывается.
- Отредактируйте файл Config для текущего процесса.
  - В таблице параметров (**Settings**) присвойте значение **InHouse\_Process4** параметру **QueueName**. Очередь в Orchestrator будет определяться с тем же именем.
  - В **таблице** параметров (Settings) добавьте настройки для параметров **System1 URL** и **System1 Credential**.
  - В списке констант (**Constants**) укажите значение **MaxRetryNumber** как "2".
- Внесите следующие изменения в шаблон:
  - Переменная TransactionItem в главном файле (Main) должна иметь тип System.String. Следует также убедиться, что типы аргументов в файлах **GetTransactionData**, **Process** и **SetTransactionStatus** совпадают с типом элемента транзакции (TransactionItem).

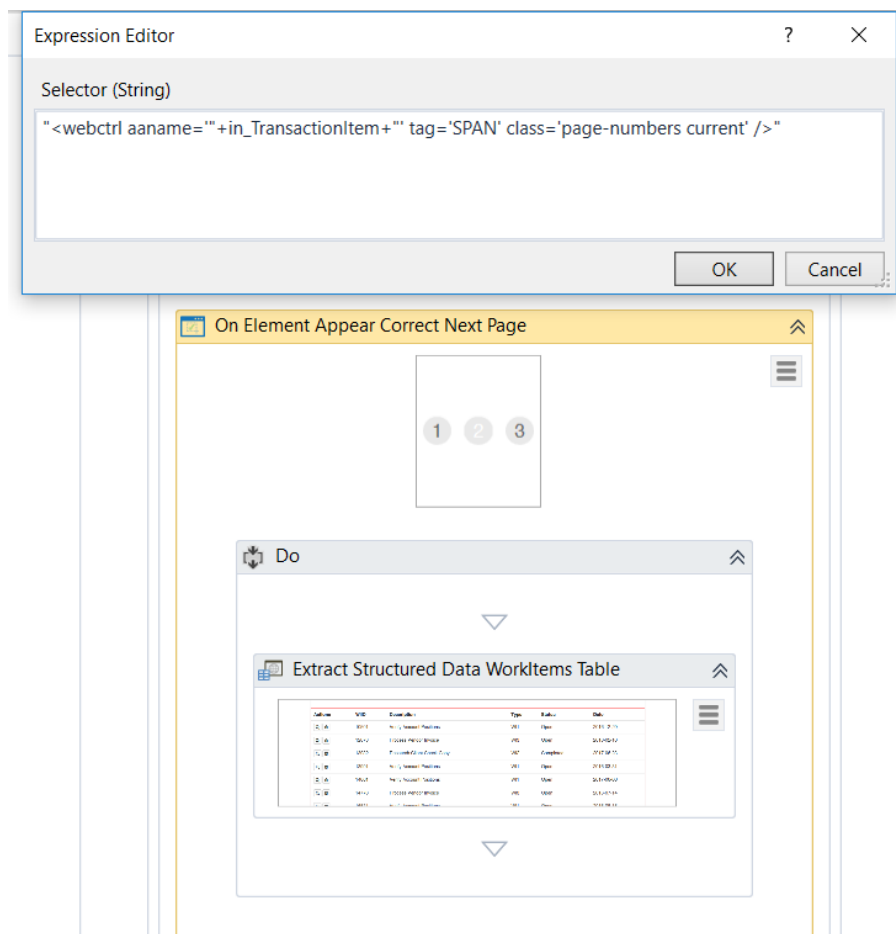
- Мы будем использовать только одно приложение в этом упражнении - ACME System1. Создайте папку с именем **System1** в корневой папке задачи.
  - Следующие компоненты в Процессе 5 можно использовать повторно.
    - Скопируйте файлы **System1\_Login.xaml**, **System1\_Close.xaml** и **System1\_NavigateTo\_WorkItems.xaml** в папку **System1**.
    - Скопируйте файл **SendEmail.xaml** в папку **Common** ("Общие").
- Откройте файл **InitAllApplications**.
  - Вызовите файл **System1\System1\_Login.xaml**.
  - Запустите файл **System1\System1\_NavigateTo\_WorkItems.xaml**
- Откройте файл **CloseAllApplications** .
  - Вызовите файл **System1\System1\_Close.xaml**.
- Откройте файл **KillAllProcesses.xaml** в папке Framework.
  - Добавьте действие **Kill Process** ("Завершить процесс") и переименуйте его в "Kill Process IE".
- Откройте проект **GetTransactionData.xaml** в папке Framework. Его можно найти в состоянии **Get Transaction Data** ("Получить данные транзакции").
  - Удалите действие **Get Transaction Item** («Получить элемент транзакции»), поскольку оно не требуется, так как для передачи данных в очередь используется процесс Диспетчер.
  - Перед последовательностью **Write Transaction info in Logging Fields** ("Записать сведения о транзакциях в поля логирования") добавьте действие **Attach Browser** («Присоединить браузер») и присоедините страницу рабочих элементов.
  - Добавьте действие **Element Exists** для проверки доступности следующей страницы. Укажите номер страницы и измените селектор, чтобы использовать атрибут, относящийся к номеру страницы (напоминание: аргумент **in\_TransactionNumber** является номером страницы для Диспетчера).
  - Вот так должно выглядеть действие **Element Exists**



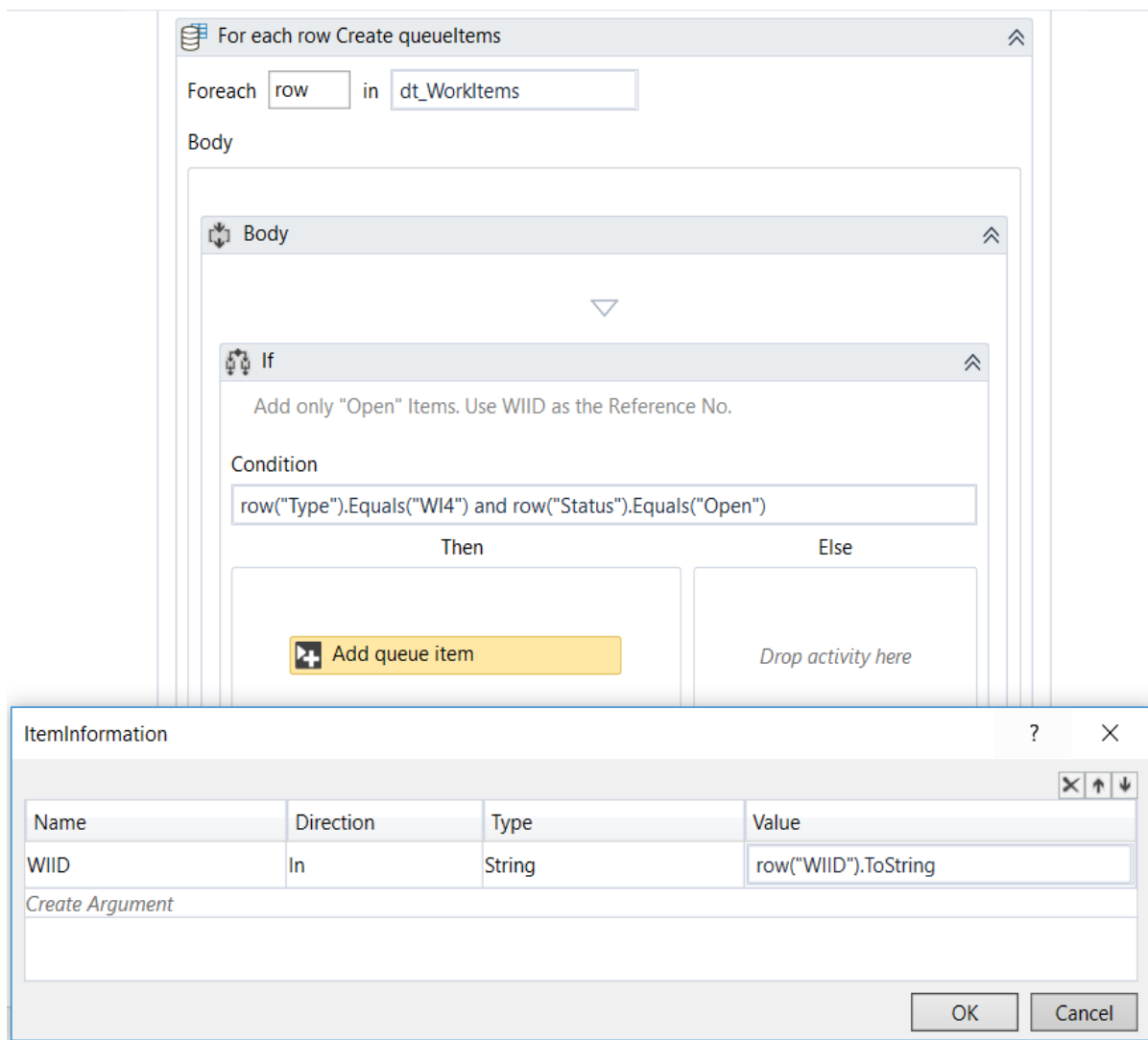
- Используйте действие **If**, чтобы проверить, остались ли еще данные транзакций.
- Если следующая страница существует, припишите выходному аргументу **out\_TransactionItem** значение текущей страницы, а именно **in\_TransactionNumber**.
- Если следующей страницы не существует, укажите для **out\_TransactionItem** значение **Nothing**.
- Вот как должно выглядеть действие **If**:



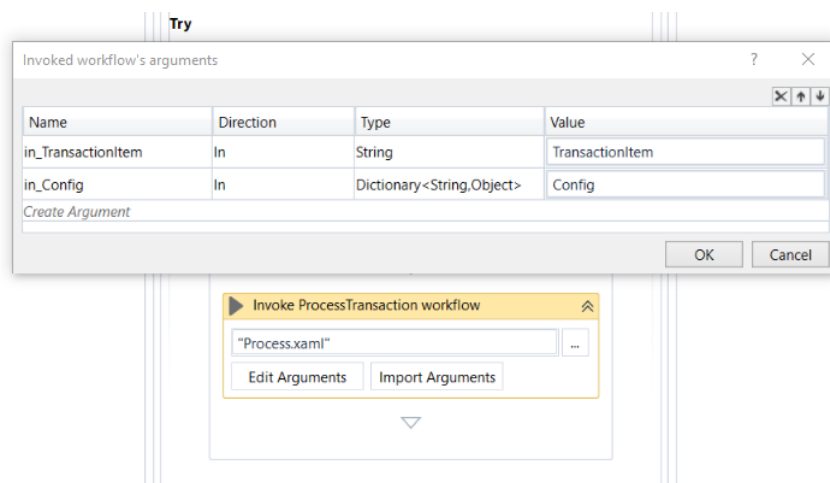
- Откройте файл **Process.xaml** в папке Framework. Его можно найти в состоянии **Process Transaction** ("Получить данные транзакции").
  - Используйте действие **Click**, чтобы выбрать номер обрабатываемой страницы, используя тот же динамический селектор для идентификации текущей страницы (аргумент **in\_TransactionNumber**).
  - Затем добавьте действие **On Element Appear** ("При появлении элемента"), чтобы проверить, открыта ли обрабатываемая страница. Вы можете использовать UiExplorer. Атрибут класса можно использовать для идентификации активных или неактивных страниц, поэтому он подходит и для создания динамического селектора. Внутри действия **On Element Appear** считайте таблицу, содержащую рабочие элементы. Создайте переменную с именем **dt\_WorkItems** в свойстве **Output** для их хранения.
  - Вот так должно выглядеть действие **On Element Appear**:



- Следующим шагом является отправка **WIID** для всех строк внутри считанной таблицы с данными, которые имеют тип WI4 и статус Open в очереди. Чтобы подгрузить значение в очередь, используйте действие **Add Queue Item** ("Добавить элемент очереди"). На панели **Properties** («Свойства») используйте значение из словаря **in\_Config** для заполнения поля **QueueName**. В поле **ItemInformation** создайте аргумент с именем **WIID**. Задайте соответствующее значение для аргумента.
- Так должно выглядеть действие **Add Queue Item** ("Добавить элемент очереди"):



- Откройте файл **Main.xaml**.
  - Убедитесь, что аргументы для проекта, инициированного **Process.xaml**, заданы верно.
  - Аргументы должны выглядеть так:



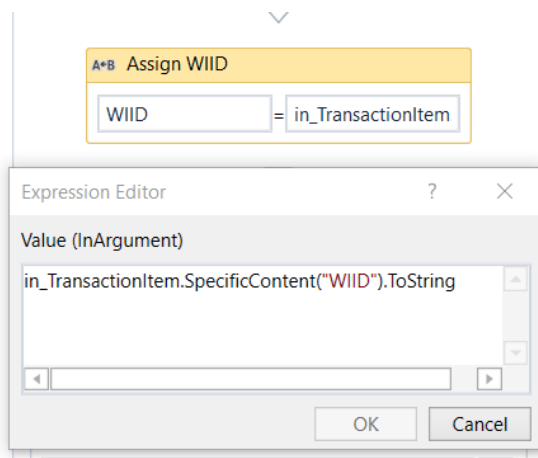
- Мы закончили с реализацией процесса. Далее необходимо протестировать процесс и проверить, правильно ли загружены значения в очередь.

## Процесс Исполнителя

Исполнитель обрабатывает все транзакции, загруженные Диспетчером в очередь. По этой причине тип TransactionItem должен быть элементом очереди (**QueueItem**).

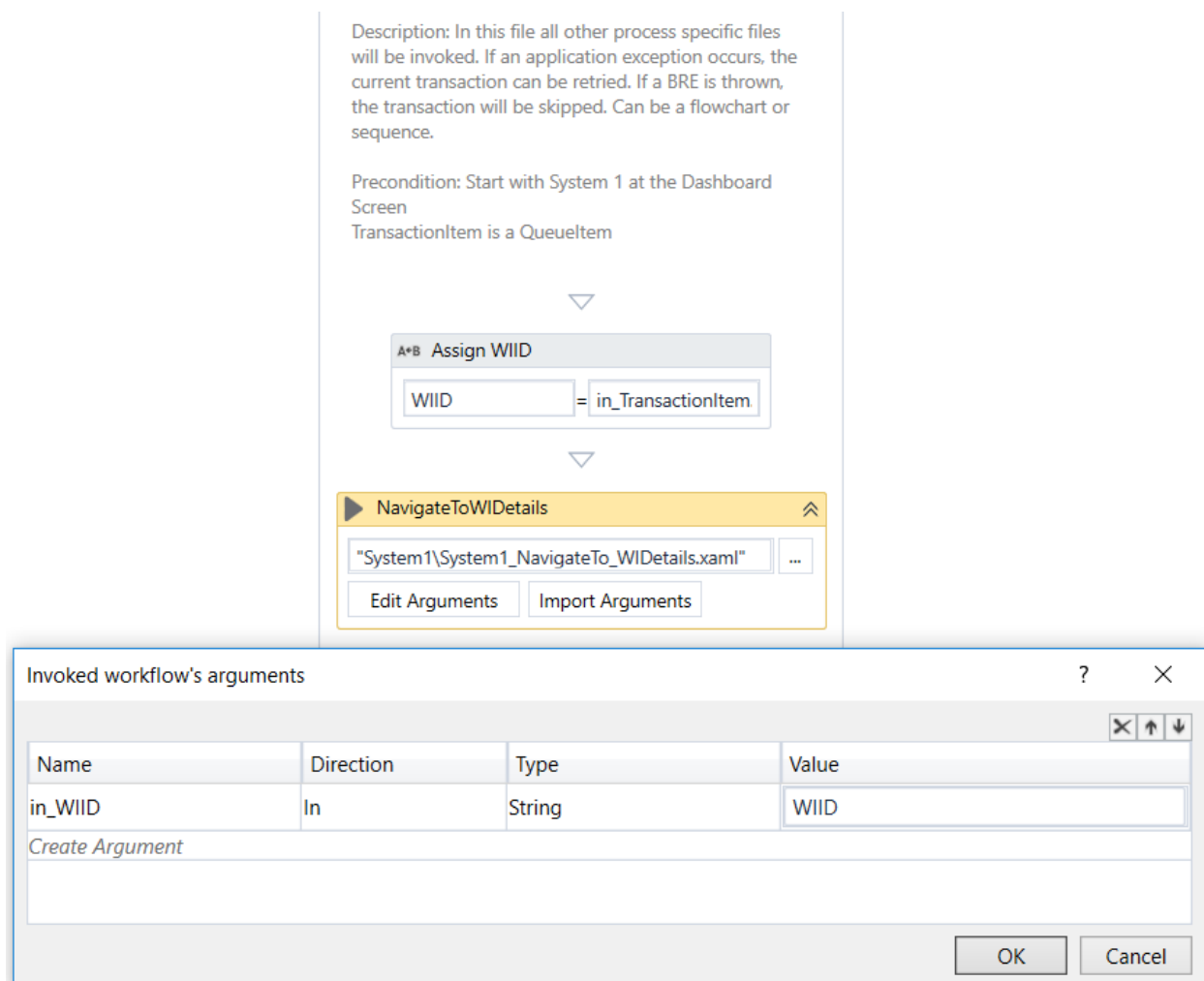
- Начнем с шаблона REFramework.
  - Аргумент **TransactionItem** должен принадлежать к типу "элемент очереди" (**QueueItem**). Этот тип аргумента TransactionItem задан в REFramework по умолчанию.
- Измените **файл конфигурации** для текущего процесса следующим образом:
  - В таблице параметров (**Settings**) добавьте значение InHouse\_Process4 параметру **QueueName**. Очередь в Orchestrator будет определяться с тем же именем.
  - В таблице параметров (**Settings**) добавьте настройки для параметров **System1 URL** и **System1 Credential**.
  - В списке констант (**Constants**) выставьте значение **MaxRetryNumber** на 0, так как мы используем элементы очереди в процессе Исполнителя, а механизм повторных попыток обрабатывается в Orchestrator.
- В этом упражнении мы будем использовать только одно приложение: ACME System1. Создайте папку с именем **System1** в корневой папке задачи.
  - Следующие компоненты в процессе **Исполнителя** могут быть использованы повторно.
    - Скопируйте файлы **System1\_Login.xaml**, **System1\_Close.xaml**, **System1\_NavigateTo\_WorkItems.xaml**, **System1\_NavigateTo\_WIDetails.xaml** и **System1\_UpdateWorkItem.xaml** в папку **System1**.
    - Скопируйте файл **SendEmail.xaml** в папку **Common** ("Общие").
- Откройте файл **InitAllApplications**.
  - Вызовите файл **System1\System1\_Login.xaml**.
- Откройте файл **CloseAllApplications**.
  - Вызовите файл **System1\System1\_Close.xaml**.

- Откройте проект **KillAllProcesses.xaml** в папке Framework. Добавьте действие **Kill Process** ("Завершить процесс") и переименуйте его в **Kill Process IE**.
  - Задайте для свойства **ProcessName** значение **"iexplore"**.
- Откройте проект **Process.xaml** в папке Framework. Его можно найти в состоянии **Get Transaction Data** ("Получить данные транзакции").
  - Создайте строковую переменную для задания идентификатора текущего рабочего элемента (WIID).
  - Добавьте действие **Assign** и присвойте переменную, созданную выше, значению в очереди. Для этого можно использовать метод `SpecificContent` - `in_TransactionItem.SpecificContent("WIID").ToString`
  - Так должно выглядеть действие **Assign**



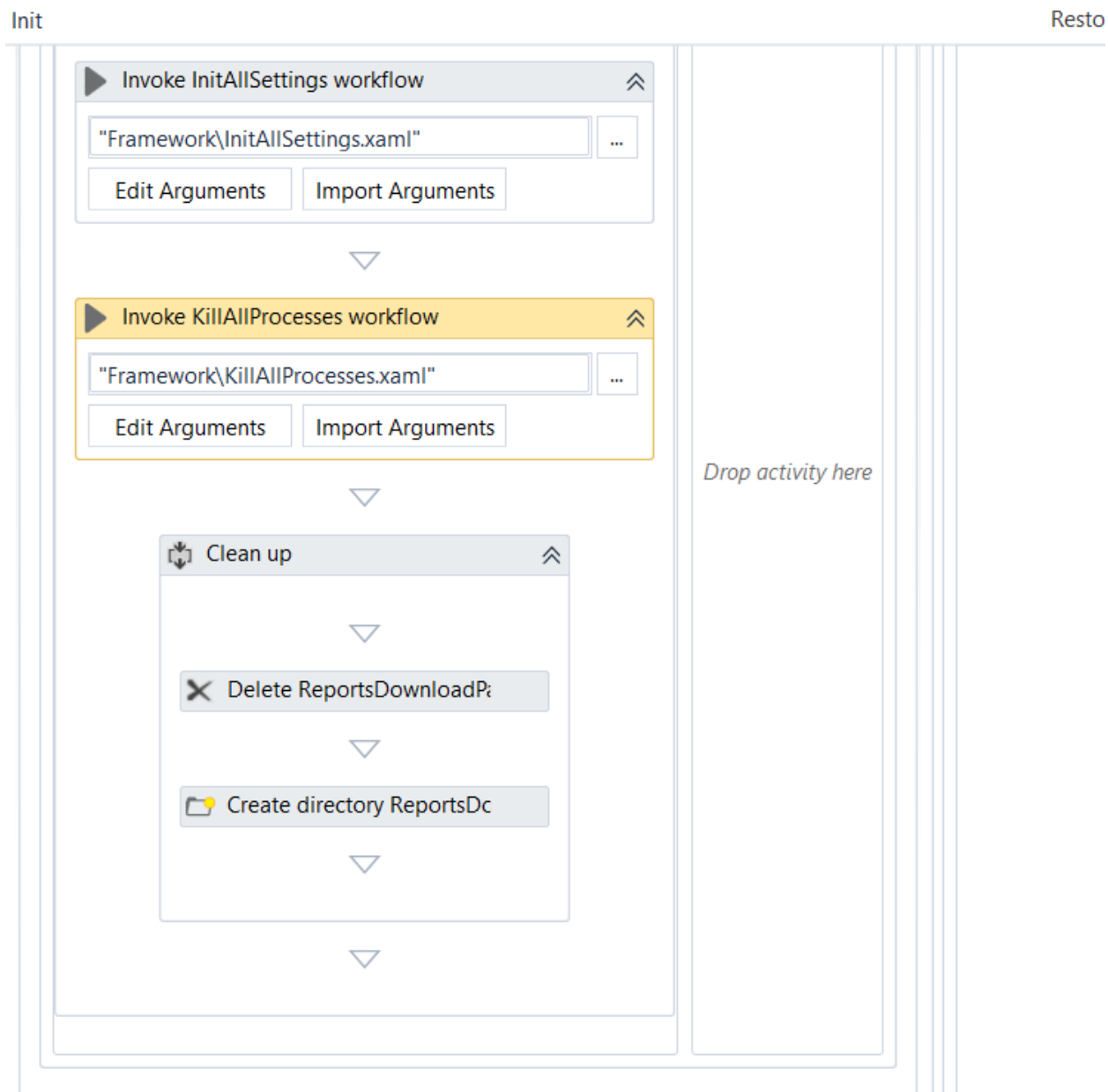
- Откройте файл **Process.xaml** в папке Framework. Вызовите **System1\System1\_NavigateTo\_WIDetails.xaml**. Импортируйте и привяжите аргумент, который должен быть взят из очереди с помощью метода **SpecificContent**, как описано выше.
- Так должны выглядеть действие **Invoke** ("Вызвать") и аргумент WIID:





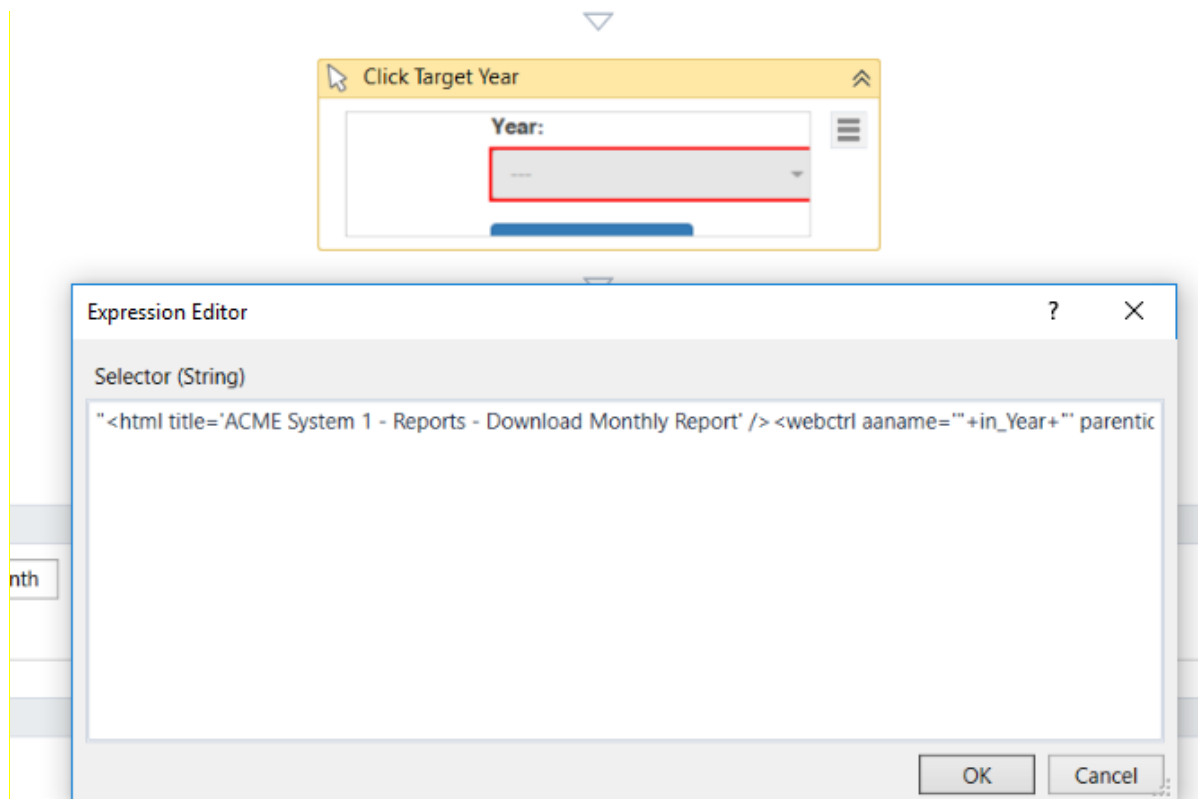
- Затем создайте пустой рабочий процесс в папке System1. Мы воспользуемся им для извлечения значения TaxID из страницы сведений о рабочем элементе (Work Item Details). Мы можем назвать этот процесс System1\_ExtractVendorInformation.xaml. На выходе мы должны получить аргумент TaxID.
- Откройте **Process.xaml** и создайте строковую переменную с именем TaxID для хранения значения аргумента out из предыдущего созданного файла рабочего процесса. Вызовите **System1\System1\_ExtractVendorInformation.xaml** и свяжите аргумент.
- Затем создайте новый рабочий процесс для перехода на страницу Панели управления (Dashboard page). Рабочий процесс должен включать действие **Click** для выбора страницы Панели управления. Назовем этот рабочий процесс **System1\_NavigateTo\_Dashboard.xaml**.
- Вызовите рабочий процесс в файле **Process.xaml**.

- Теперь, когда значение TaxID извлечено, нам нужно перейти на страницу загрузки ежемесячного отчета ( **Download Monthly Report**). Для этого создадим новую пустую последовательность с названием **System1\_NavigateTo\_MonthlyReport**.
- Загрузка существующих инвойсов может вызвать некоторые проблемы. Однако, чтобы избежать возникновения ошибок, мы должны убедиться, что среда - чистая каждый раз, когда запускается робот. Для этого мы можем удалить папку **Download Reports**, упомянутую в **конфигурационном файле**, а затем создать ее заново. Откройте Init State в файле Main.xaml. Добавьте последовательность с именем **Clean Up** ("Очистить") после вызванного файла KillAllProcesses.xaml. Используем два действия: **Delete** ("Удалить") и **Create Directory** ("Создать директорию").
- На панели «Свойства» действия **Delete** задайте в поле **Path** («Путь») значение из **файла конфигурации**. Таким образом, вы всегда можете получить новый пустой каталог Data\Temp, когда робот запущен. Так должна выглядеть последовательность Clean Up:

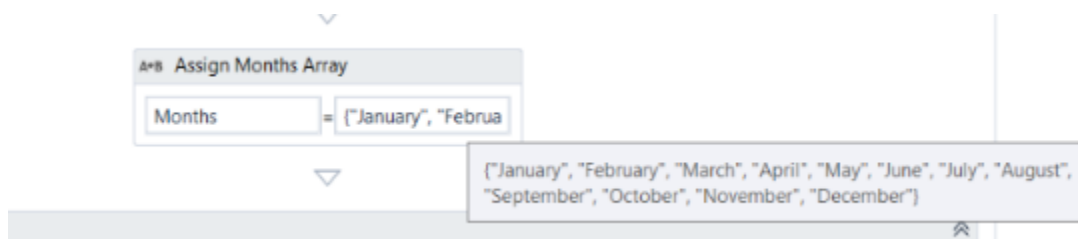


- Далее, возвращаясь к файлу **Process.xaml**, вызовите созданный выше файл **System1\System1\_NavigateTo\_MonthlyReport.xaml**.
- Создайте новую переменную с именем **ReportYear**. Используйте действие **Assign**, чтобы задать его значение для предыдущего года.
- Следующим шагом является создание годового отчета. Для этого нам нужно создать пустой файл последовательности с именем **System1\_CreateYearlyReport.xaml**.
  - Начните с соответствующей аннотации. Предусловием является открытие страницы ежемесячного отчета (Monthly Report Page) в приложении ACME System 1.
  - Создайте три аргумента In, как показано ниже:

- **in\_TaxID** – предоставляется из основного файла. Он хранит значение TaxID .
  - **in\_Year** – предоставляется из основного файла. В нем хранится год, для которого будет создан отчет
  - **in\_ReportsDownloadPath** – папка, в которую будут загружаться ежемесячные отчеты.
- Создайте один выходной аргумент с именем **out\_YearlyReportPath**, чтобы сохранить путь к файлу годового отчета, созданному после объединения всех ежемесячных отчетов.
  - Добавьте новый элемент в конфигурационный файл в таблице Параметров (Settings), чтобы указать путь к папке, в которую загружаются отчеты. Заполните поле Name, набрав **ReportsDownloadPath**, и поле Value, указав **Data\Temp**.
  - Создайте новую переменную таблицы данных с именем dt\_YearlyReport. Мы используем её для объединения всех ежемесячных отчетов. Занесите её значение в **новую таблицу DataTable** с помощью действия **Assign**.
  - Затем добавьте действие **Type Into**, чтобы ввести значение TaxID в приложении ACME System 1. На этом этапе страница ежемесячного отчета уже должна быть открыта .
  - Добавьте действие **Click** для выбора года. Включите свойство **Simulate Click** ("Имитация щелчка"), а затем выберите желаемый год. Таким образом, действие может выполняться в фоновом режиме, даже если раскрывающееся меню не открыто и элемент невидим. Измените атрибут aaname в селекторе на аргумент **in\_Year**. Так должны выглядеть действие **Click** и селектор:

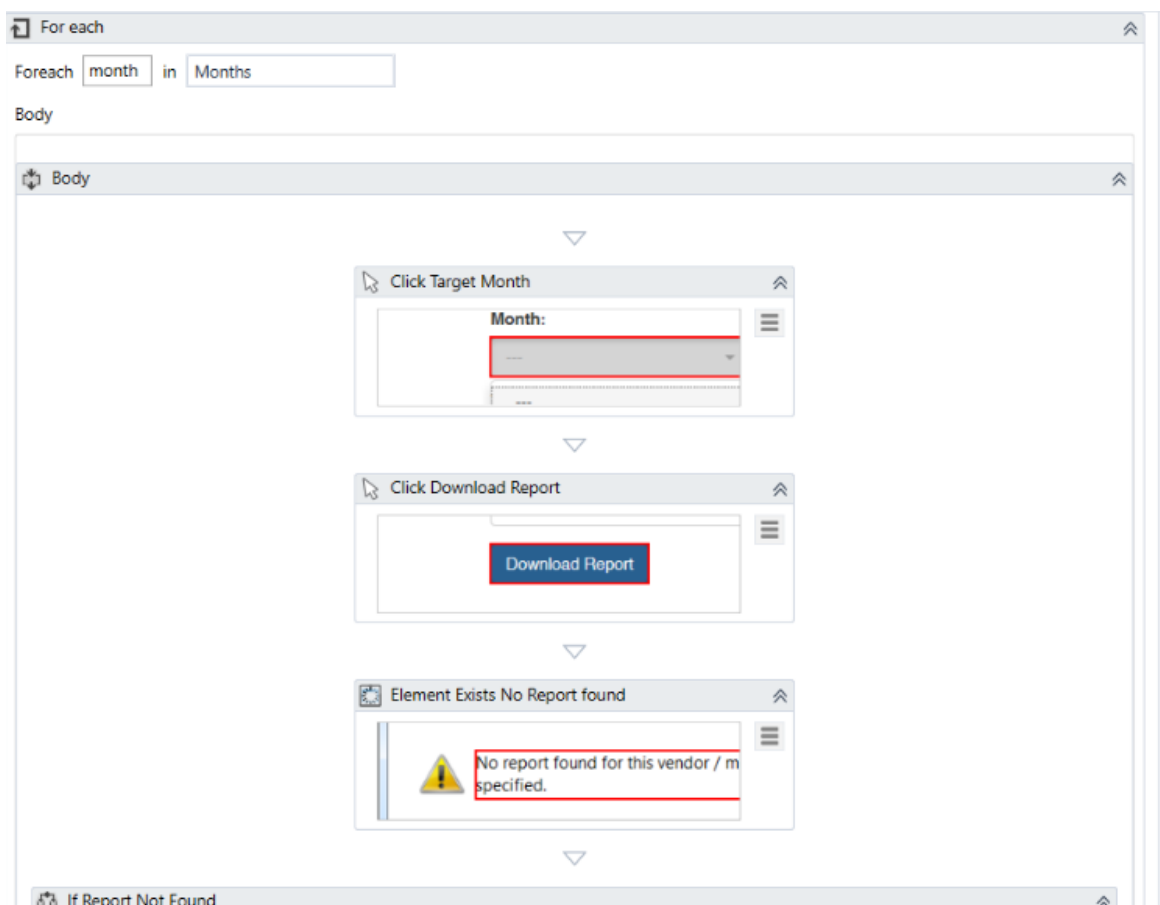


- Добавьте переменную строкового массива с именем **Months** ("Месяцы"). Используйте действие **Assign**, чтобы задать его значения в соответствии с параметрами в раскрывающемся списке **Month** ("Месяц"). Так должно выглядеть действие **Assign**

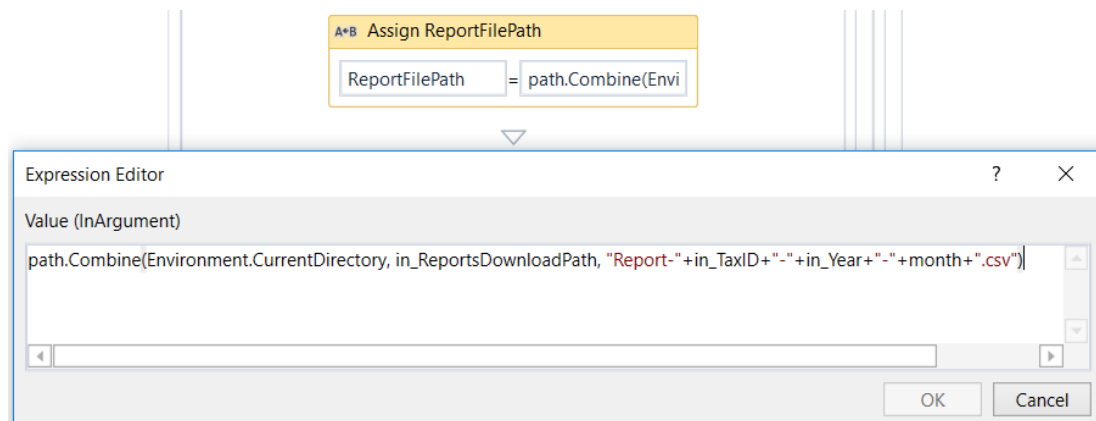


- Далее, чтобы загрузить отчеты за каждый месяц, необходимо добавить действие **For Each** для итерации по массиву Months, выбрать в раскрывающемся списке указанный месяц и загрузить отчет.
- Внутри действия **For Each** добавьте действие **Click**, чтобы выбрать целевой месяц. Отредактируйте селектор так же, как редактируется селектор для выпадающего списка Year ("Год"), с помощью динамического атрибута aaname.
- Затем добавьте действие **Click** и укажите кнопку загрузки (Download button). Выберите свойство **Simulate Click** ("Имитация щелчка").

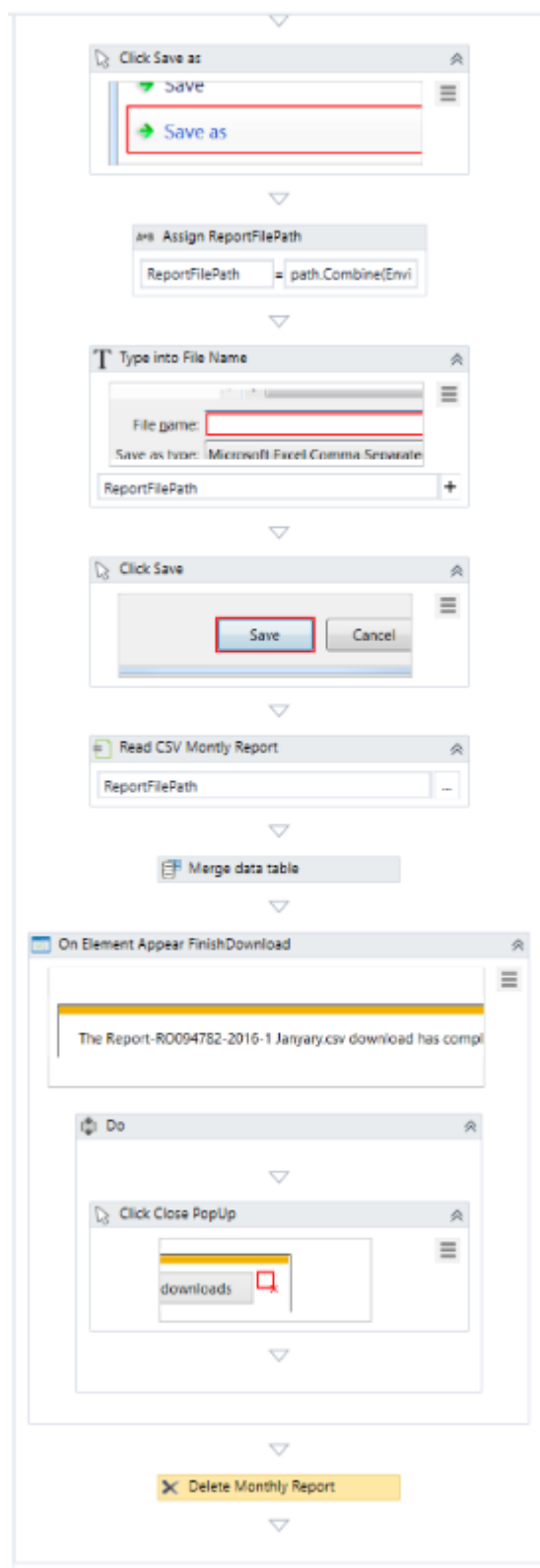
- Поскольку некоторые отчеты не существуют, добавьте действие **Element Exists** и укажите метку всплывающего окна, показанного в данном случае. Создайте логическую переменную с именем ReportNotFound в свойстве **Output**.
- На данном этапе действие **For Each** должно выглядеть следующим образом:



- Следующим шагом является добавление действия **If**, чтобы проверить, найден ли отчет. Используйте в качестве **Condition** ("Условия") переменную ReportNotFound. В разделе **Then** добавьте действие **Click** и установите в качестве его цели кнопку **OK** из всплывающего окна, чтобы перейти к следующему месяцу.
- В разделе **Else** загрузите Отчет, используя следующие действия:
  - Добавьте действие **Click** и нацельте его на кнопку **Save as** («Сохранить как»). Активируйте свойство **Simulate Click** ("Имитация щелчка").
  - Добавьте действие **Assign**. Создайте новую переменную с именем ReportFilePath и установите ее значение в виде пути и имени файла ежемесячного отчета, который загружается в формате .csv.

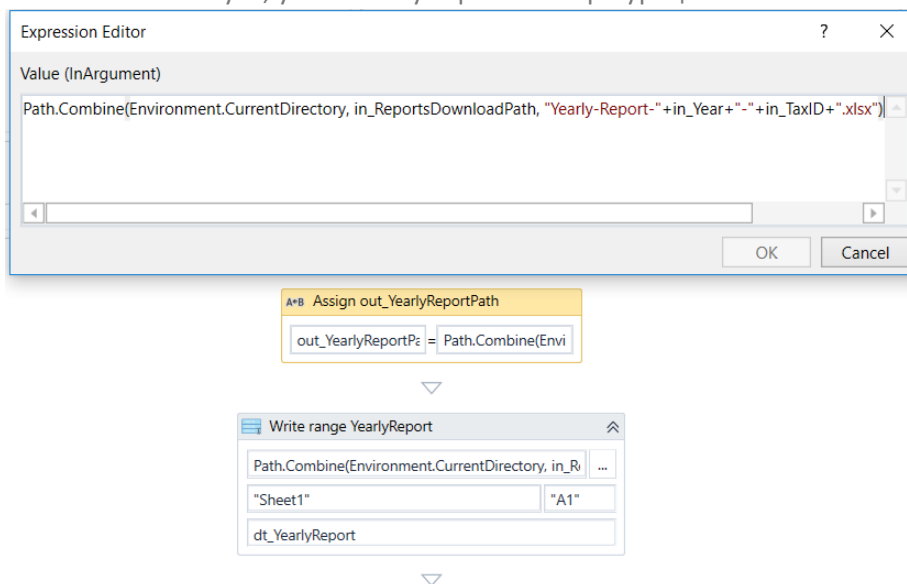


- Используя действие **Type Into**, заполните значение ReportFilePath в поле File Name ("Имя файла") окна Save As ("Сохранить как"). Выберите свойство **Simulate Type** ("Имитировать ввод с клавиатуры").
- Добавьте действие **Click** и направьте его к кнопке **Save** («Сохранить»). Активируйте свойство **Simulate Click**.
- Считайте только что загруженный CSV-файл. В выходном свойстве создайте переменную таблицы данных с именем dt\_MonthlyReport.
- Затем с помощью действия **Merge Data Table** ("Слияние таблицы данных") добавьте значения **dt\_MonthlyReport** в таблицу данных **dt\_YearlyReport**.
- Загрузка ежемесячного отчета может занимать разное количество времени. Чтобы удостовериться в том, что файл был полностью загружен перед загрузкой отчета за следующий месяц, добавьте действие **On Element Appear** и укажите всплывающее окно загрузки. Обновите селектор с подстановочными знаками для значений динамических атрибутов. Активируйте свойство **Wait Visible**.
- В разделе **Do** действия **On Element Appear** добавьте **Click**, чтобы закрыть всплывающее окно. Выберите свойство **Simulate Click** ("Имитация щелчка").
- Добавьте действие **Delete File** ("Удалить файл"), чтобы удалить файл ежемесячного отчета перед загрузкой следующего.
- Так должен выглядеть раздел **Else**:

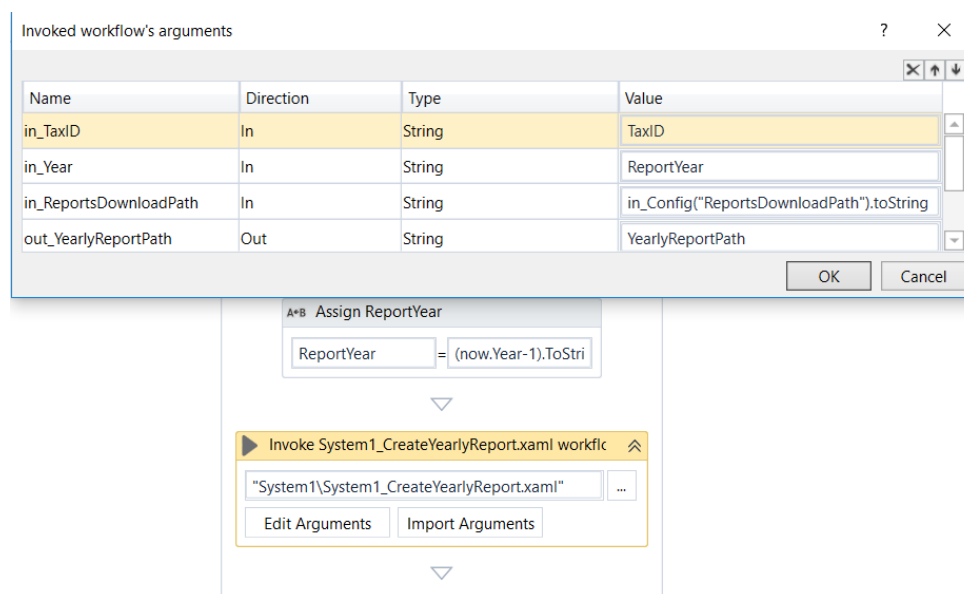




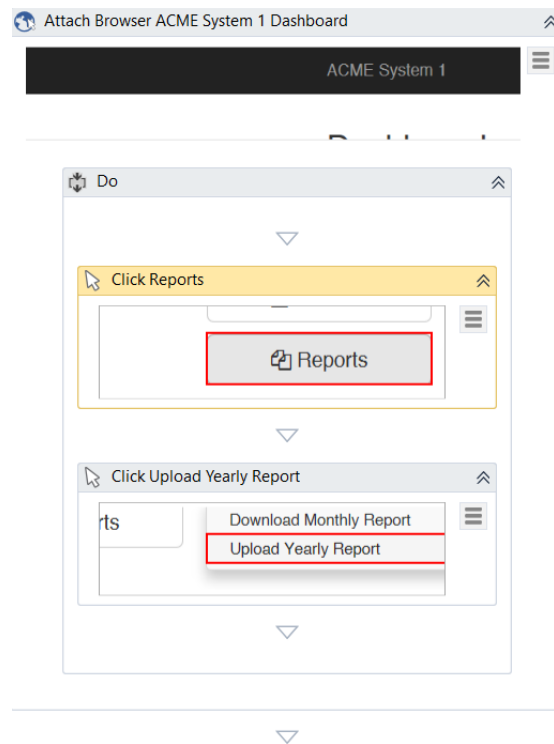
- Следующим действием в последовательности является **Assign**, которое используется для установки значения аргумента **out\_YearlyReportPath**. Убедитесь, что имя файла годового отчета Excel соответствует модели файла PDD и пути, указанному в файле конфигурации.



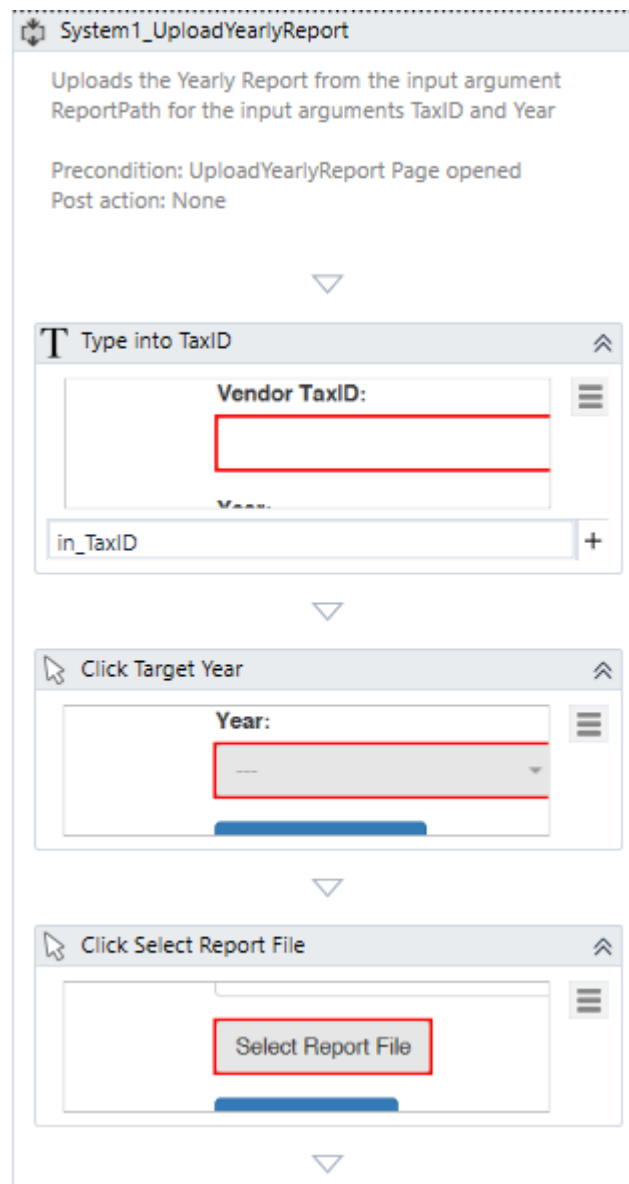
- Задайте значения по умолчанию для аргументов In и протестируйте рабочий процесс.
- Вернитесь к рабочему процессу **Process**.
  - Вызовите файл **System1\System1\_CreateYearlyReport.xaml**, созданный выше. Импортируйте и привяжите аргументы.
  - Создайте строковую переменную с именем **YearlyReportPath**. Она используется для получения значения аргумента **out\_yearlyReportPath** в предыдущем рабочем процессе.
  - Так должны выглядеть действие **Invoke** ("Вызвать") и его аргументы:



- Затем вызовите **System1\System1\_NavigateTo\_Dashboard.xaml** для перемещения обратно на страницу Панели управления.
- Теперь, когда мы создали файл Годового отчета, нам необходимо перейти к странице "Отчеты - Загрузить ежегодный отчет" в приложении ACME System 1. Для этого нам потребуется создать пустую последовательность с именем **System1\_NavigateTo\_UploadYearlyReport**.
  - Начните новый рабочий процесс, добавив аннотацию. Предусловием является открытие страницы Панели управления.
  - Добавьте действие **Attach Browser** и укажите страницу Панели управления.
  - Добавьте действие **Click**, чтобы выбрать кнопку **Reports** ("Отчеты"). Активируйте свойство **SimulateClick**.
  - Затем добавьте новое действие **Click**, чтобы выбрать кнопку **Upload Yearly Report** ("Загрузить годовой отчет"). Используйте **UiExplorer**, чтобы нажать на эту кнопку, как вы уже сделали в рабочем процессе **System1\_NavigateTo\_MonthlyReport.xaml**. Активируйте свойство **SimulateClick**.
  - Так должен выглядеть рабочий процесс:



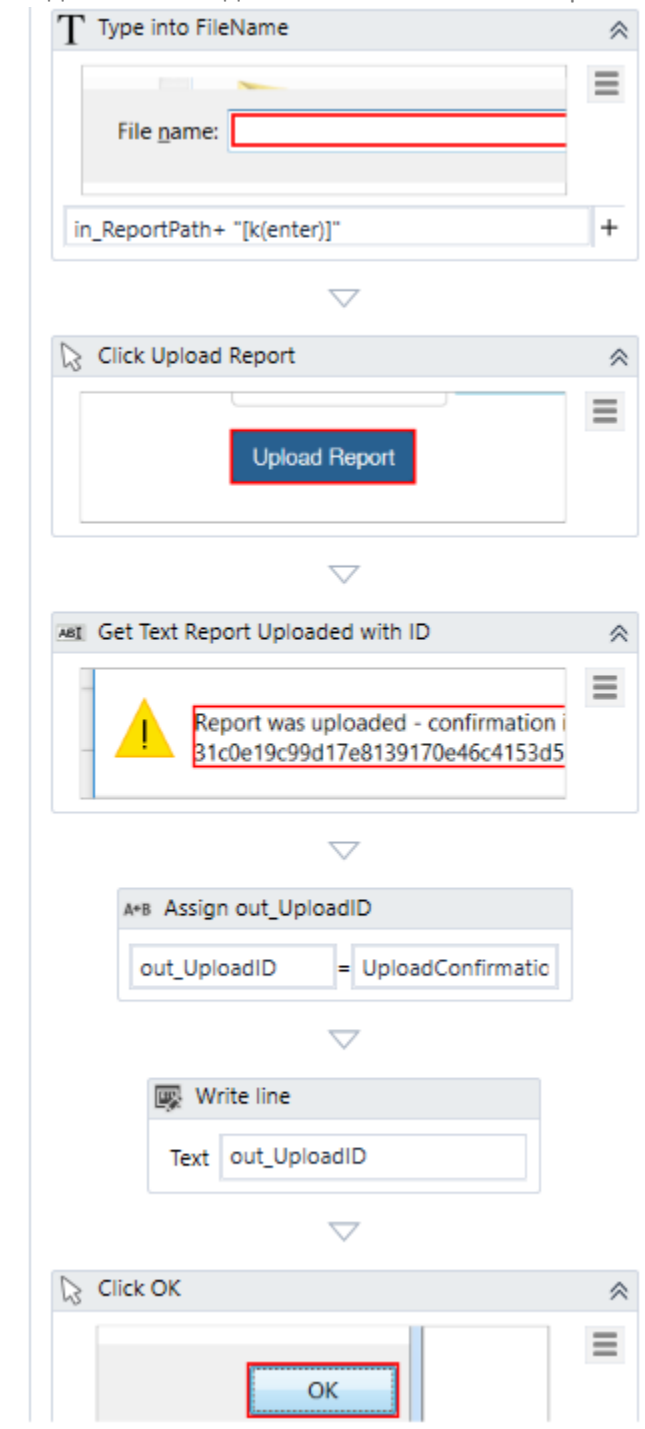
- Вернитесь к рабочему процессу **Process**.
  - Вызовите файл **System1\System1\_NavigateTo\_UploadYearlyReport.xaml**.
  - После перехода на страницу "Отчеты - Загрузить ежегодный отчет", мы должны загрузить годовой отчет. Для этого создадим новую пустую последовательность с названием **System1\_UploadYearlyReport**.
    - Начните новую последовательность, добавив аннотацию. Предусловие заключается в том, что страница "Отчеты - Загрузить ежегодный отчет" открыта.
    - Информация, необходимая для загрузки файла ежегодного отчета, состоит из taxID, пути к файлу и года. При выполнении загрузки генерируется идентификатор подтверждения. Таким образом, в этом рабочем процессе мы должны использовать 3 строковых In аргумента: **in\_TaxID**, **in\_ReportPath**, **in\_Year**, и один строковой выходной аргумент: **out\_UploadID**.
    - Теперь, когда все настроено, добавьте действие **Type Into** и укажите поле **Vendor TaxID**. В качестве текста используйте аргумент **in\_TaxID**.
    - Используйте действие **Click** в выпадающем меню **Year**. Обновите селектор с помощью аргумента **in\_Year**.
    - Используйте действие **Click** на кнопке **Select Report File** ("Выбрать файл отчета").
    - На данный момент последовательность должна выглядеть следующим образом:



- Теперь отображается окно **Choose file to Upload**, поэтому мы должны использовать действие **Type Into**, чтобы установить путь к файлу ежегодного отчета (**in\_ReportPath**) и нажать "Ввод".
- Нажмите кнопку "Загрузить" с помощью действия **Click**.
- Отобразится всплывающее окно с идентификатором подтверждения загрузки, и мы воспользуемся действием **Get Text** для извлечения его значения. В свойстве **Output** создайте переменную с именем **UploadConfirmation**.
- Подставьте значение аргумента **out\_UploadID** в один из идентификаторов подтверждения с помощью действия **Assign**. Используйте метод **Substring** для извлечения значения, как показано ниже:

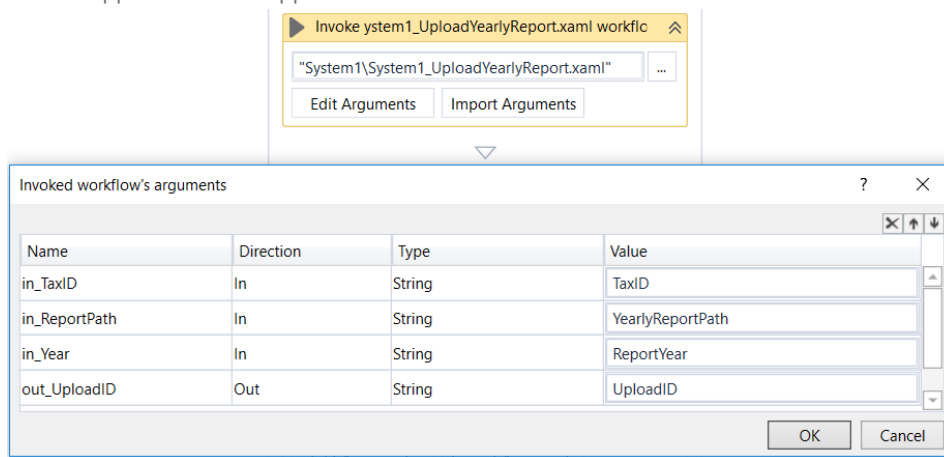
UploadConfirmation.Substring("Отчет был загружен - идентификатор подтверждения равен ".Length)

- Используйте действие **Click** на кнопке **OK**. Готово!
- Так должна выглядеть заключительная часть рабочего процесса.

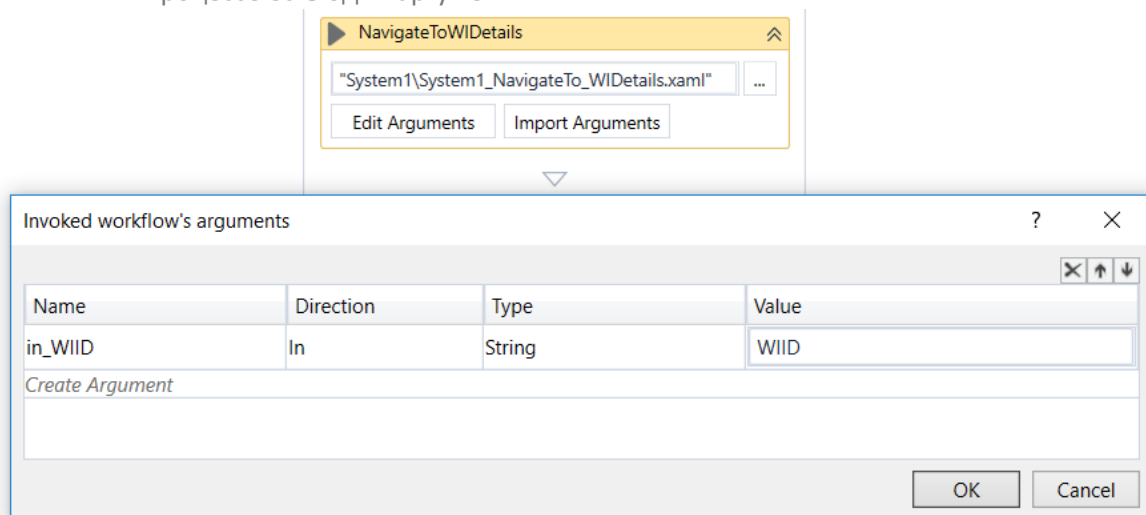


- Вернуться к рабочему процессу **Process**.

- Вызовите файл **System1\System1\_UploadYearlyReport.xaml** и привяжите соответствующие аргументы. Создадим переменную в рабочем процессе **Process** для хранения значения аргумента **out\_UploadID**. Назовите переменную UploadID.
- Так должен выглядеть вызов:



- На этом этапе мы загрузили файл ежегодного отчета, поэтому необходимо обновить статус рабочих элементов.
- Вызовите файл **System1\System1\_NavigateTo\_Dashboard.xaml** для перехода к странице Панели управления.
- Вызовите файл **System1\System1\_NavigateTo\_WIDetails.xaml** для перехода к странице сведений о конкретном рабочем элементе. Как видите, в этом рабочем процессе есть один аргумент - WIID.



- Обновите статус рабочего элемента **Complete**, вызвав файл **System1\System1\_UpdateWorkItem.xaml**. Убедитесь, что заданы правильные аргументы для разделов **Comment** и **Status**. Как упоминалось в файле PDD, присвойте статусу значение **Completed** ("Завершено"), а Комментарию - значение **Uploaded with ID [uploadID]** ("Загружено с идентификатором").

- Наконец, нам нужно вернуть приложение в исходное состояние, чтобы мы могли обработать следующий элемент. Для этого запустите файл **System1\System1\_NavigateTo\_Dashboard.xaml** для возвращения на страницу Панели управления.
- Мы закончили с реализацией процесса. Далее необходимо протестировать весь процесс. Вы уже должны были протестировать каждый отдельный рабочий процесс сразу после разработки, используя значения по умолчанию для аргументов .
  - Запустите рабочий процесс **Main** несколько раз и убедитесь, что каждый раз он выполняется корректно. Если это не так, устраните неполадки и запустите его снова.
  - Используйте параметр "Сброс тестовых данных" (**Reset test data**) в меню Параметры пользователя (User options) для создания нового набора данных для целей тестирования.
  - При необходимости используйте **Диспетчер** для загрузки новых элементов в очередь.

## Примечания по реализации процесса.

Мы начали с процесса **Диспетчера**, который использовался для загрузки элементов очереди в Orchestrator. Затем мы обрабатывали каждый элемент очереди с помощью **Исполнителя**. Обратите внимание, что статус транзакции в очереди изменяется после ее обработки. Все элементы являются независимыми друг от друга и могут быть обработаны параллельно, с использованием нескольких роботов-Исполнителей.