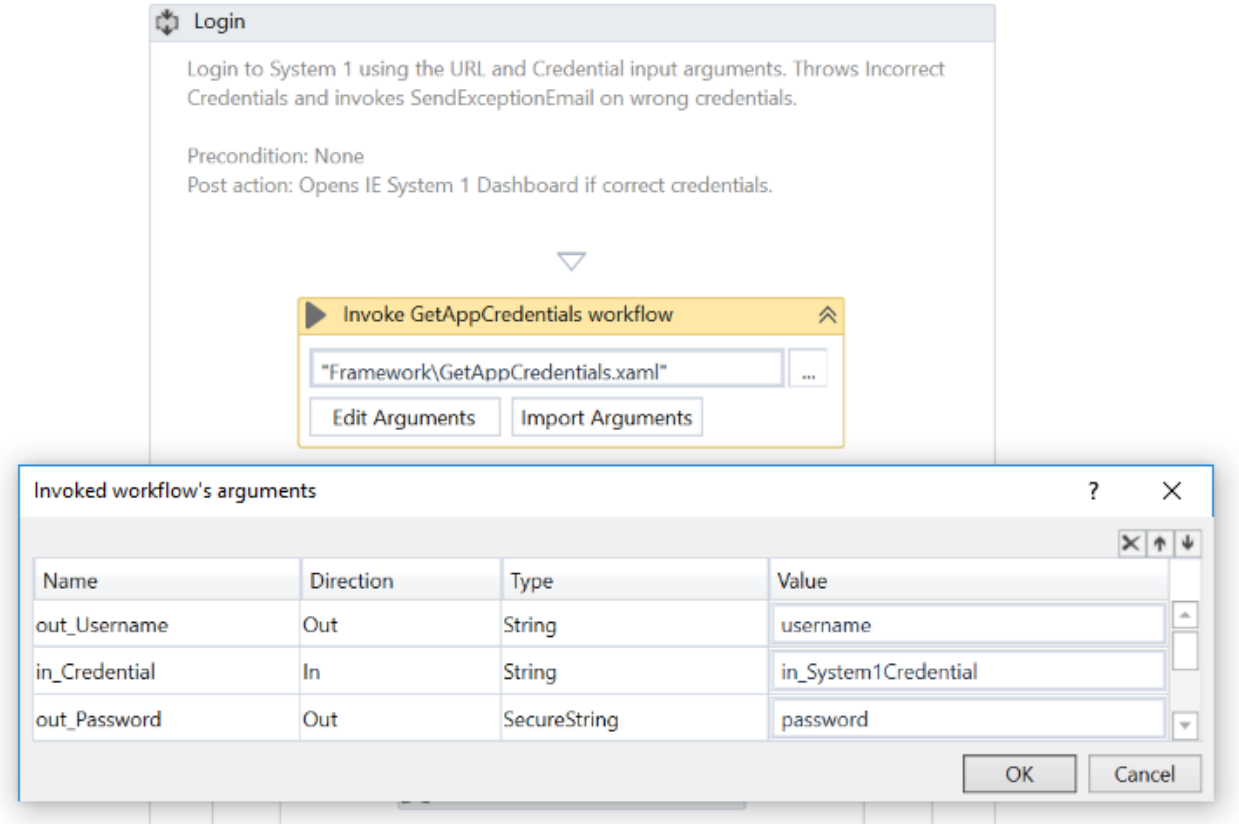
Автоматизация UiPath

Пошаговое руководство

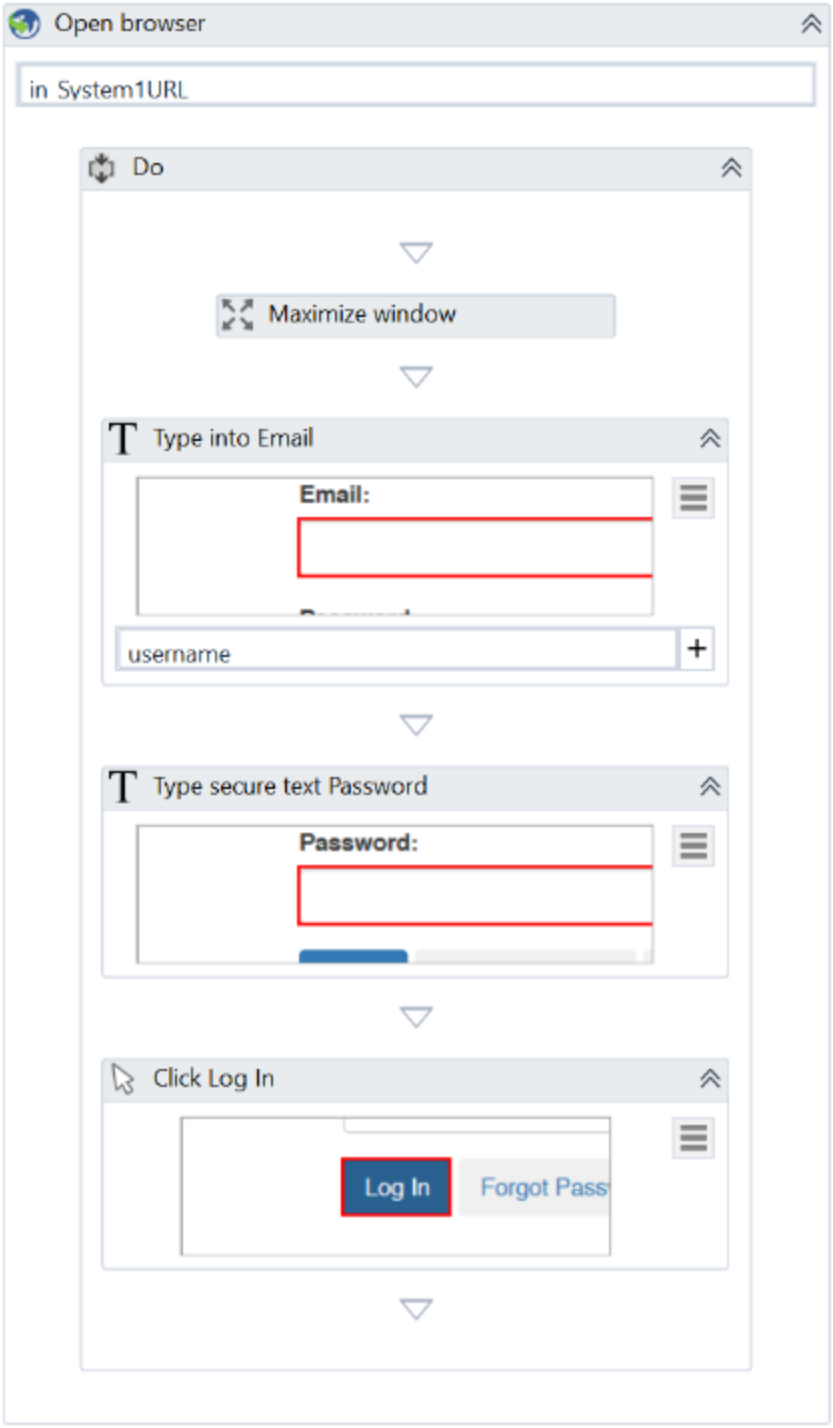
Пошаговое руководство – Вьнисление контрольной суммы безопасности клиента

- Начнем с шаблона REFramework.
    - Мы начинаем с простой задачи, чтобы продемонстрировать шаблон REFramework без использования очередей Orchestrator.
    - Пожалуйста, взгляните на входные данные (список элементов очереди); мы можем извлечь всю таблицу при помощи Мастера считывания данных (Data Scraping wizard).
    - Элемент транзакции является строкой данных из списка. Такой же подход применяется при вводе данных, извлеченных из таблиц Excel, CSV-файлов и баз данных.
- Рекомендуется сохранять значения, подверженные изменениям, в файле конфигурации. Откройте рабочую книгу Data\Config.xlsx и переключитесь на страницу Настроек (Settings).
    - Добавьте настройки для **System1 URL** и **SHA1 Online URL**.
    - Приложение System1 требует аутентификации; Мы будем использовать Orchestrator Assets для хранения учетных данных для System1. Добавьте еще один параметр, System1_Credential, чтобы сохранить имя из учетных данных.
    - Добавьте ресурс (Asset) типа учетных данных (Credential) и создайте имя пользователя и пароль для System1. Убедитесь, что имя ресурса совпадает со значением параметра System1_Credential.
- Переключитесь на лист констант (**Constants**) в рабочей книге **Config** и установите значение MaxRetryNumber на 2. Этот параметр определяет, сколько раз шаблон попытается обработать рабочий элемент при сбое с исключением приложения, прежде чем перейти к следующему элементу.
- Внесите в шаблон следующие изменения.
    - Переменная **TransactionItem** в файле **Main** должна принадлежать к типу System.Data.DataRow, так как мы извлекаем всю таблицу, чтобы затем обработать ее построчно. Также следует изменить тип аргументов в рабочих процессах **GetTransactionData**, **Process** и**SetTransactionStatus**, чтобы они соответствовали типу TransactionItem.
    - Удалите три действия **SetTransactionStatus** из рабочего процесса SetTransactionStatus, поскольку мы не используем функциональные возможности транзакций, предоставляемые Orchestrator.
- Мы используем два приложения в этом упражнении, **ACME System1** и **SHA1-Online.com**. Создайте две папки "System1" и "SHA1Online" в корневом каталоге задачи и используйте их для рабочих процессов, созданных для двух приложений.

- Создайте пустую последовательность в папке System1 для процесса входа в System1. Мы хотим создать многократно используемый компонент, который можно использовать с множеством различных учетных данных.
    - Рекомендуется начинать новую последовательность с короткой аннотации, чтобы объяснить цель рабочего процесса. Это то, что мы собираемся делать во всех последующих файлах. Аннотация начинается с описания, включающего используемые аргументы, предусловие и действие POST.
    - Создайте два In аргумента - один для **System1 URL,**, а другой для **System1 Credential**.
    - Запустите файл рабочего процесса Framework\GetAppCredential.
    - Так должна выглядеть последовательность:



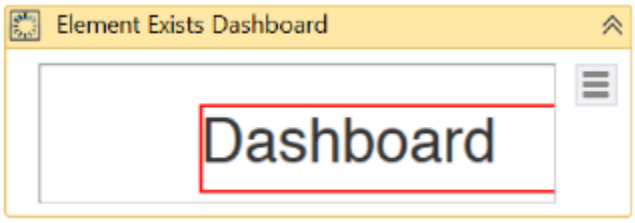
    - Заполните раздел входа в систему. Это должно выглядеть так:



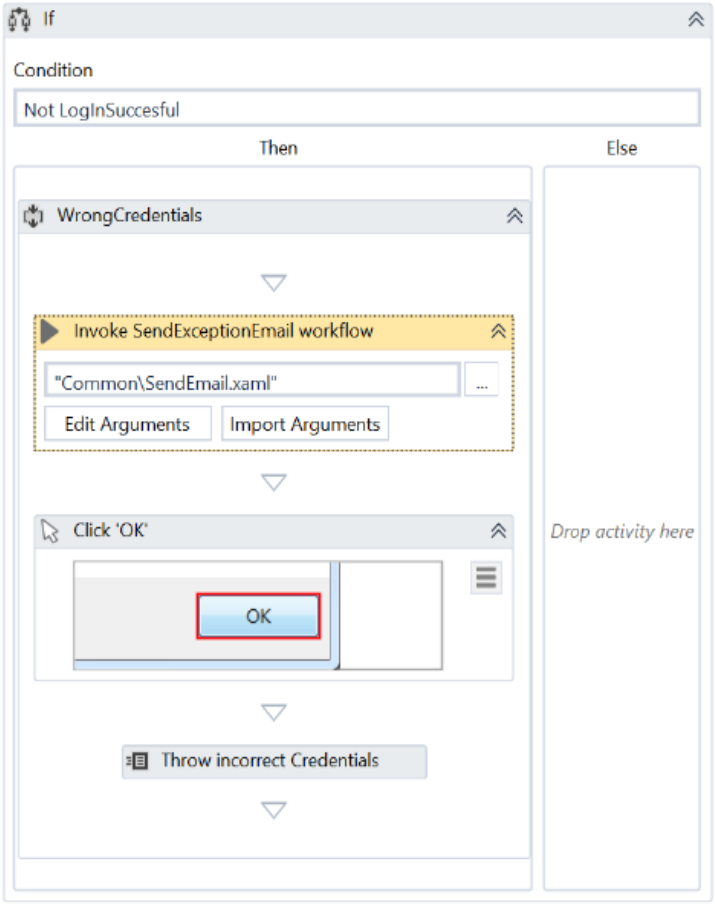
- Постарайтесь представить себе исключения, которые могут возникнуть во время выполнения нашего процесса. В данном случае, мы должны проверить, был ли успешным вход в систему. Это требование также представлено в PDD файле. Попробуйте войти в систему с неверными учетными данными и обратите внимание на разницу.

Пошаговое руководство – Вычисление контрольной суммы безопасности клиента 3

- Используйте действие **Element Exist** , чтобы проверить, удался ли вход в систему, выполнив поиск элемента, который отображается только в данном случае.



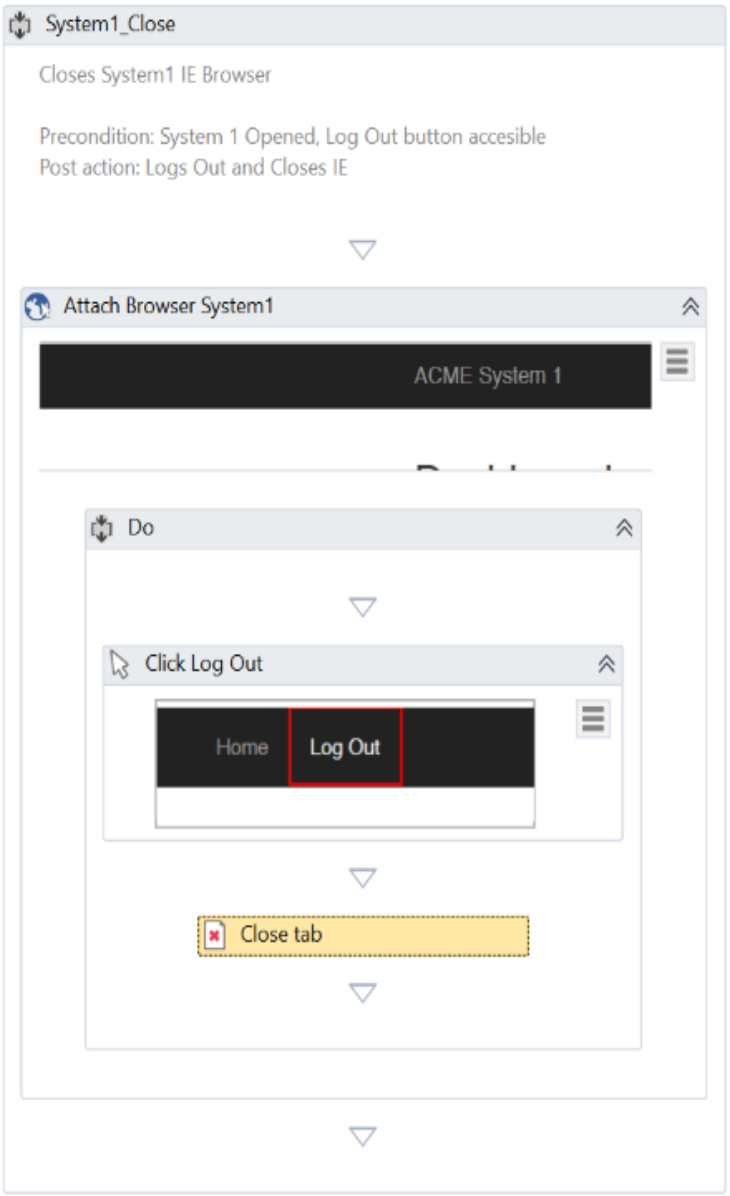
- Используйте действие **If**, чтобы проверить, удалось ли выполнить попытку входа. Если войти не удалось, выполните действия, перечисленные ниже.
  - Отправьте сообщение электронной почты с исключением. Рекомендуется создать отдельный повторно используемый рабочий процесс и вызвать его в разделе **Then**.
  - Закройте сообщение об ошибке с помощью действия **Click**.
  - Чтобы остановить процесс, выдайте ошибку для неверных учетных данных (Incorrect Credentials), предоставленных в System1.
- Так должно выглядеть действие **If**.



Пошаговое руководство – Вычисление контрольной суммы безопасности клиента 4

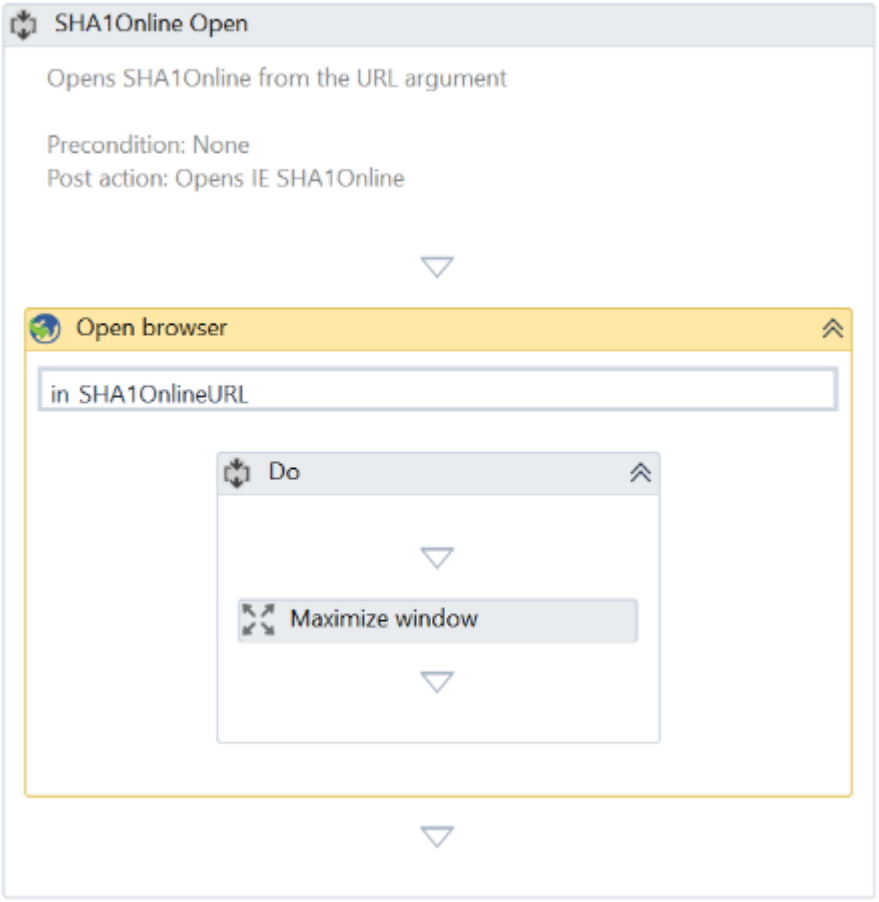
- Блочно протестируйте созданный файл, используя значения аргументов по умолчанию.

- Далее создадим рабочий процесс для выхода из системы и закрытия System1. Последовательность проста - вот как она должна выглядеть:

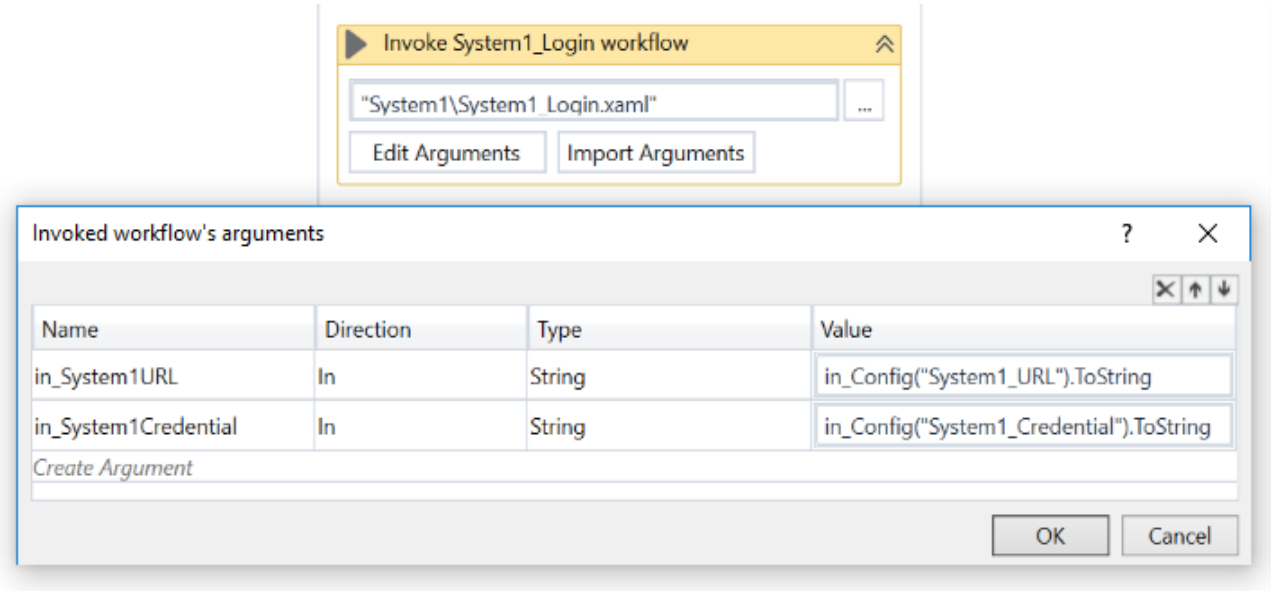


- Далее мы сделаем то же самое для другого приложения, используемого в процессе SHA1 Online.
- Создайте пустую последовательность рабочего процесса в папке SHA1Online, чтобы открыть приложение. Эта последовательность проще, так как приложение не требует аутентификации.
    - Естественно, начнем с аннотации.

- URL-адрес приложения должен быть передан в рабочий процесс с помощью аргумента, чтобы упростить обслуживание проекта. Значение URL-адреса хранится в файле конфигурации процесса. Здорово, если вы сделали это и в System1_Login!
- Последовательность должна выглядеть так:

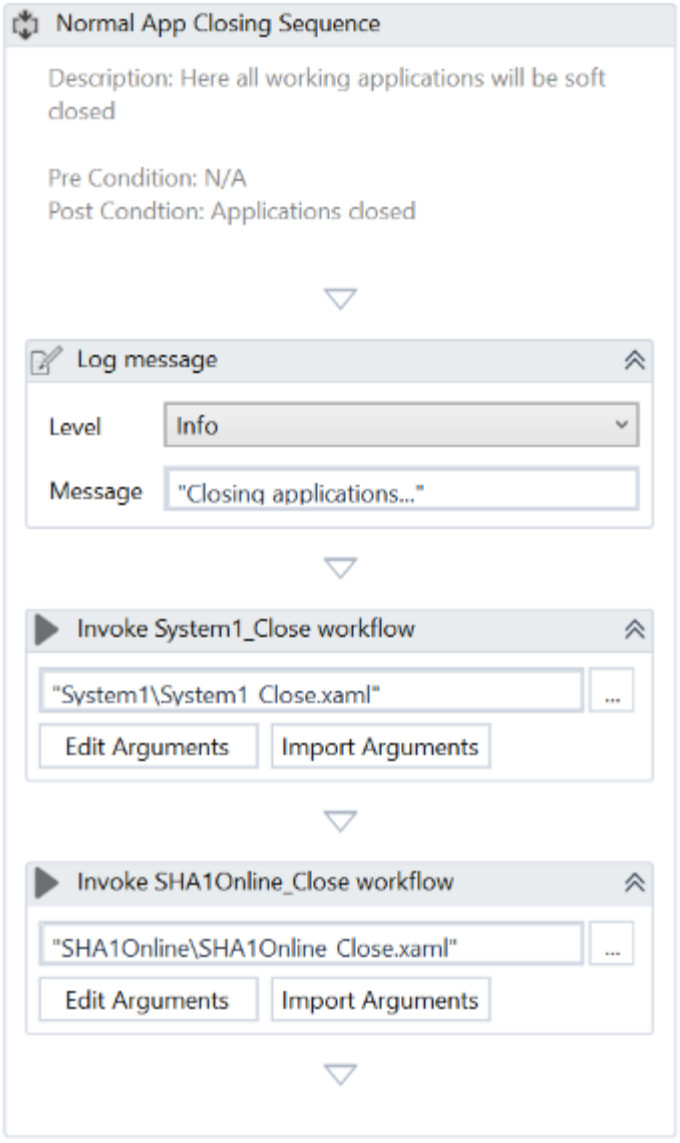


- Создайте пустую последовательность, чтобы закрыть приложение SHA1Online. Сделать это очень легко.
- Теперь, когда у нас есть рабочие процессы, которые открывают и закрывают оба приложения, мы можем вносить изменения в части инициализации и закрытия шаблона.
- Откройте рабочий поток **Framework\InitAllApplications**.
  - Перетащите файл System1_Login. Действие **Invoke Workflow File** создается автоматически.
  - Нажмите кнопку **Import Arguments** и привяжите значения к тем, которые находятся в конфигурационном файле.
  - Вот как выглядит действие **Invoke Workflow File**:



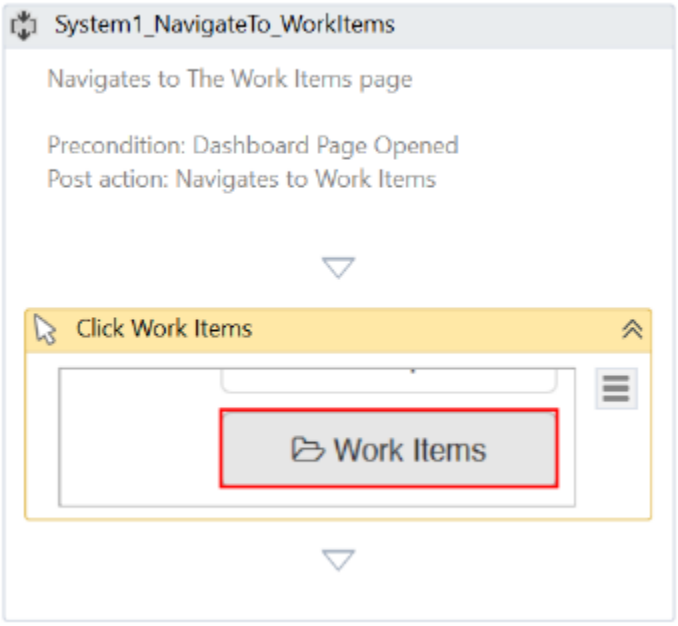
- Перетащите рабочий процесс SHA1Online_Login.
- Нажмите кнопку **Import Arguments** и привяжите аргумент URL к значению в конфигурационном файле.

- Откройте **Framework\CloseAllApplications**.
  - Перетащите два рабочих процесса, созданных для закрытия приложений System1 и SHA1Online.
  - Так в результате выглядит последовательность:



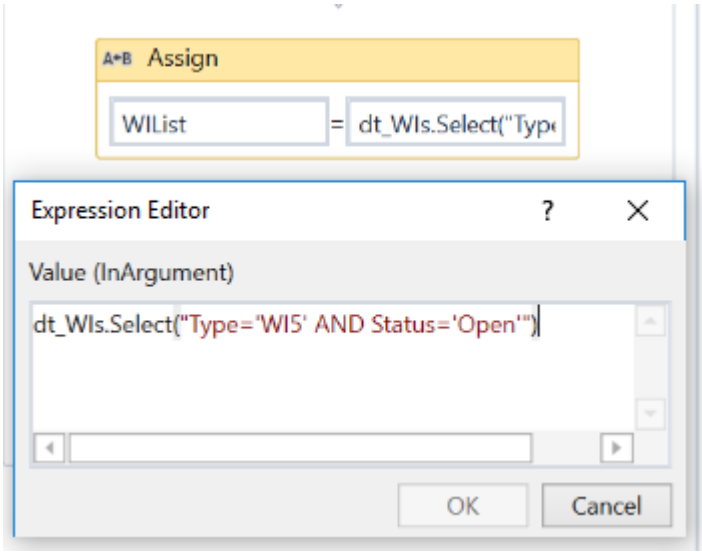
- Откройте **Framework\KillAllProcesses.**
  - Оба наших приложения размещены в веб-браузере. Если значение по умолчанию свойства **BrowserType**в действиях **Open Browser**не было изменено, используется обозреватель Internet Explorer.
  - Используйте действие **Kill Process** и задайте для имени процесса (Process Name) значение "iexplore".
- Создайте пустую последовательность рабочего процесса в папке System1 для перехода к **Work Items** в приложении System1. Мы хотим, чтобы это действие было включено в отдельный рабочий процесс, поскольку мы также собираемся переходить к рабочим элементам в других проектах.
  - Добавьте необходимую аннотацию.

- Используйте действие **Click** на кнопке Work Items ("Рабочие элементы"). Убедитесь, что используется полный селектор, так как это действие не находится внутри действия Attach Browser.
- Вот как должен выглядеть проект:



- Перед началом обработки транзакций необходимо добавить действия, необходимые для считывания входных данных в состоянии Init State. Как правило, это очень просто - вы можете использовать либо Read Range либо Read CSV. Однако в нашем процессе входные данные хранятся на веб-сайте, поэтому требуется больше шагов.
- Помните назначение этого процесса, а именно извлечение хэш-кода для каждого элемента типа WI5. Для этого нам нужен сначала список всех WI5 элементов.
- Создайте пустую последовательность рабочего процесса в папке System1, чтобы извлечь переменную таблицы данных (Data Table), которая содержит все рабочие элементы в приложении System1. Мы извлечем все доступные рабочие элементы и потом отфильтруем тип WI5.
  - Используйте Мастер считывания данных (Data Scraping wizard) для извлечения всей HTML-таблицы . На вопрос о том, охватывают ли данные несколько страниц, ответьте Да, и наведите указатель на кнопку перехода на следующую страницу.
  - Установите для параметра **Maximum number of results** значение 0, чтобы все идентифицированные элементы могли быть извлечены как выходные данные.
  - Создайте выходной аргумент и назначьте ему значение извлеченной таблицы данных.
  - Если вы уже добавляли аннотацию, то этот рабочий процесс готов! Если вы еще этого не сделали, добавьте аннотацию сейчас.
- Откройте рабочий процесс **Main** и разверните состояние **Init**, дважды щелкнув на нем.
  - В области Entry найдите место, где вызывается рабочий процесс KillAllProcesses.

- Добавьте новую последовательность после действия **Invoke KillAllProcesses** для считывания таблицы данных входных транзакций.
- Внутри этой последовательности вызовите четыре из ранее созданных рабочих процессов, как показано ниже:
  - System1 Login ("Вход в систему")
  - System1 Navigate to Work Item ("Перейти к рабочему элементу")
  - System1 Extract Work Item Data Table ("Извлечение таблицы данных рабочих элементов")
  - System1 Close ("Закрыть")
- При необходимости импортируйте и привяжите аргументы.
- Из списка рабочих элементов извлеките только элементы, необходимые в текущем процессе - те Элементы типа WI5, статус которых "Open".
  - Используйте метод DataTable.Select, чтобы отфильтровать элементы, которые не соответствуют указанным выше критериям. Назначьте результат новой переменной.
  - Так должно выглядеть действие **Assign**

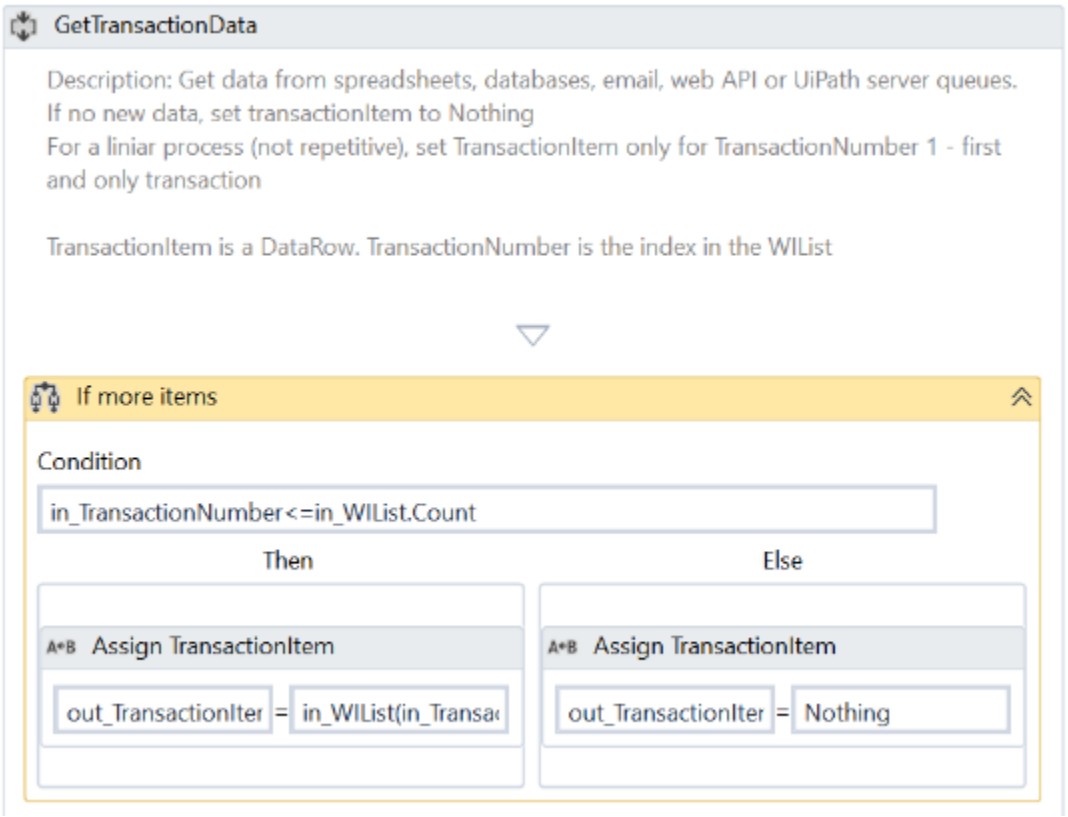


- Мы будем обрабатывать каждый элемент массива по очереди. Таким образом, все объекты DataRow становятся элементами транзакции (Transaction Items). Индекс транзакции начинается с 1, но массивы начинаются с нуля, поэтому индекс элемента в массиве на единицу меньше, чем номер транзакции.

- Откройте рабочий поток **Framework\GetTransactionData**.
  - Обновите аннотацию для текущего процесса.
  - Добавьте входной аргумент для массива Рабочих элементов (Work Items).
  - Помните, что наш элемент транзакции является одним объектом в массиве рабочих элементов. Мы можем иметь новую транзакцию только в том случае, если номер транзакции меньше или равен индексу элемента массива.
  - Добавьте действие **If**, чтобы проверить, есть ли какая-либо новая транзакция для обработки.

- В разделе **Then** используйте действие **Assign**, чтобы задать значение аргумента выходного элемента транзакции в соответствии с его индексом.
- В разделе **Else** установите для элемента транзакции значение Nothing.

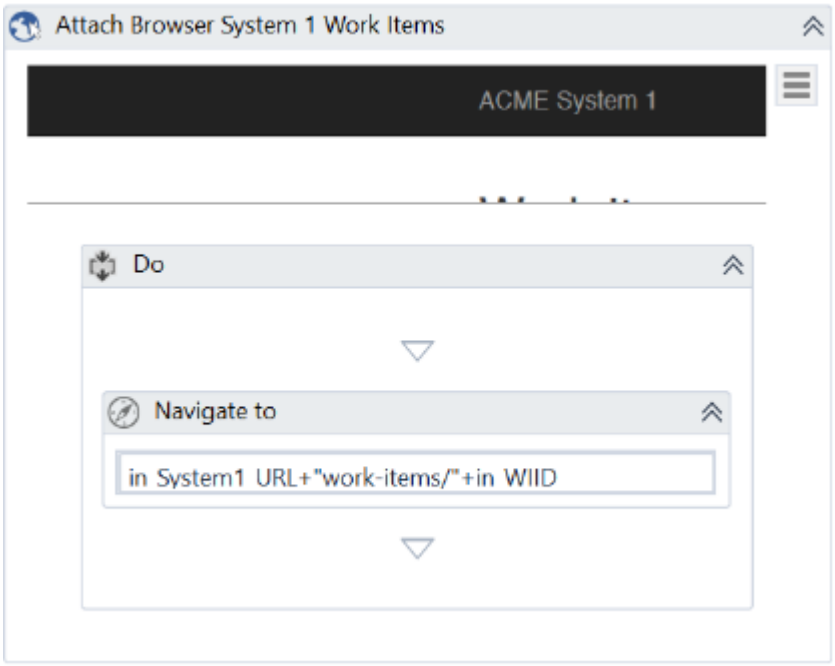
- Вот так выглядит последовательность Get Transaction Data:



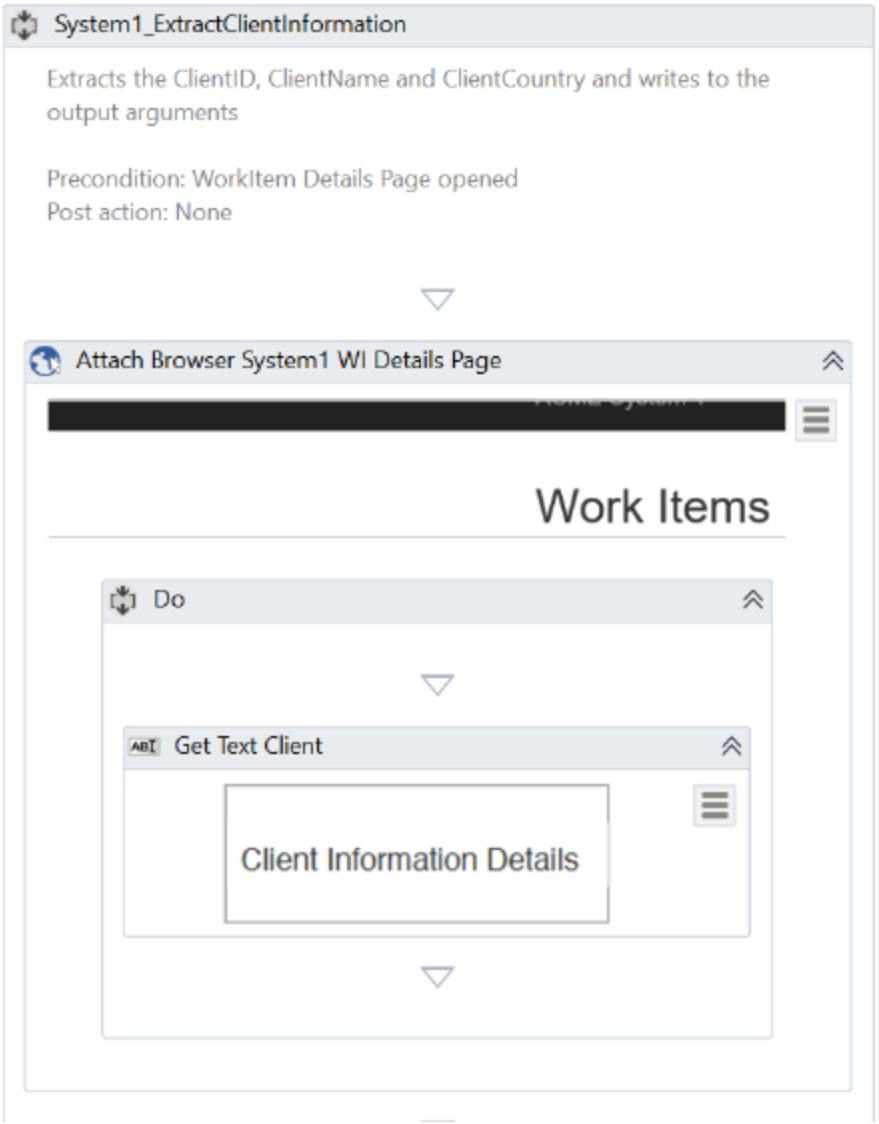
- Кроме того, при наличии элементов транзакции, оставшихся для обработки, необходимо установить идентификатор транзакции в значение идентификатора рабочего элемента. Действия уже на месте, поэтому нам нужно лишь изменить значение TransactionID в действии **Assign** на “out_TransactionItem("WIID").ToString”.

- На этом этапе конфигурация шаблона завершена. Мы можем протестировать рабочий процесс **Main**, запустив его, а затем проверить извлечение входных данных. Используйте окно "Журнал вывода" (Output log), чтобы проверить правильность данных.
- Теперь давайте приступим к разработке рабочих процессов, которые обрабатывают рабочие элементы.
- Создайте пустую последовательность в папке **System1**, чтобы перейти к странице сведений о рабочем элементе (Work Item Details page). Назовите ее **System1_NavigateTo_WIDetails**. Мы можем использовать идентификатор рабочего элемента (Work Item ID) для перехода непосредственно к странице сведений об элементе. Откройте рабочий элемент и обратите внимание на формат URL-адреса, который состоит из URL-адреса System1, строки "/work-item/" и идентификатора рабочего элемента.
  - Идентификатор рабочего элемента и URL-адрес System1 являются необходимыми входными данными. Создайте два In-аргумента - один из типа Int32, а другой типа String.

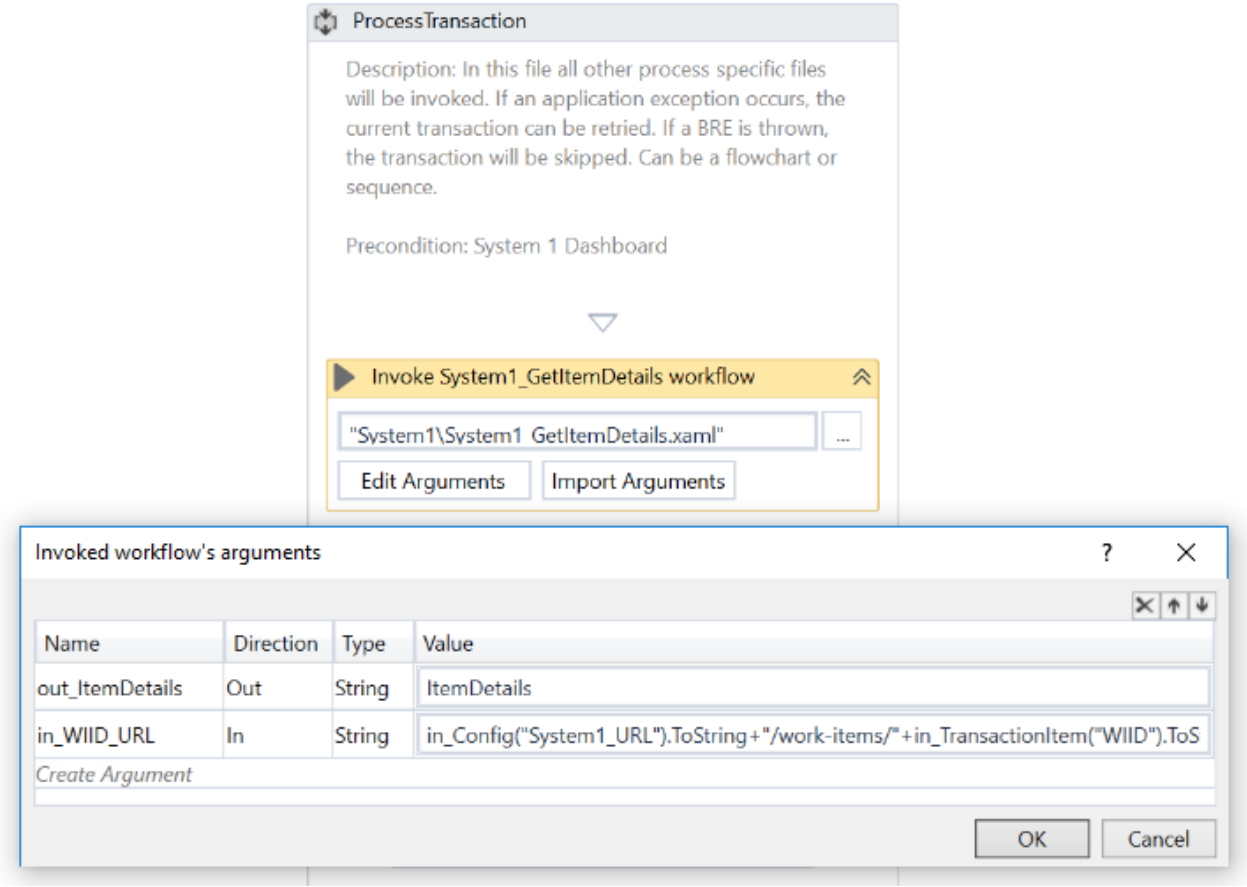
- Подключитесь к Панели управления System1 (System1 Dashboard), а затем с помощью действия **Navigate To** перейдите на страницу сведений о рабочем элементе.
- Последовательность выглядит так:



- Создайте пустой рабочий процесс в папке System1 с именем **System1_ExtractClientInformation**. Мы используем его для извлечения деталей элемента.
  - В этом рабочем процессе отсутствуют входные аргументы. Существует только предусловие - страница сведений о рабочем элементе должна быть уже открыта.
  - Нам нужны 3 выходных аргумента: Идентификатор клиента (Client ID), имя клиента (Client Name) и страна клиента (Client Country).
  - Перед получением текста с веб-страницы, мы должны убедиться, что страница загружена, и все элементы доступны. Добавьте действие **Attach Browser**. В разделе **Do** мы будем использовать действие **Get Text** со свойством **WaitForReady**, установленным как **Complete**.
  - Рабочий процесс выглядит следующим образом:



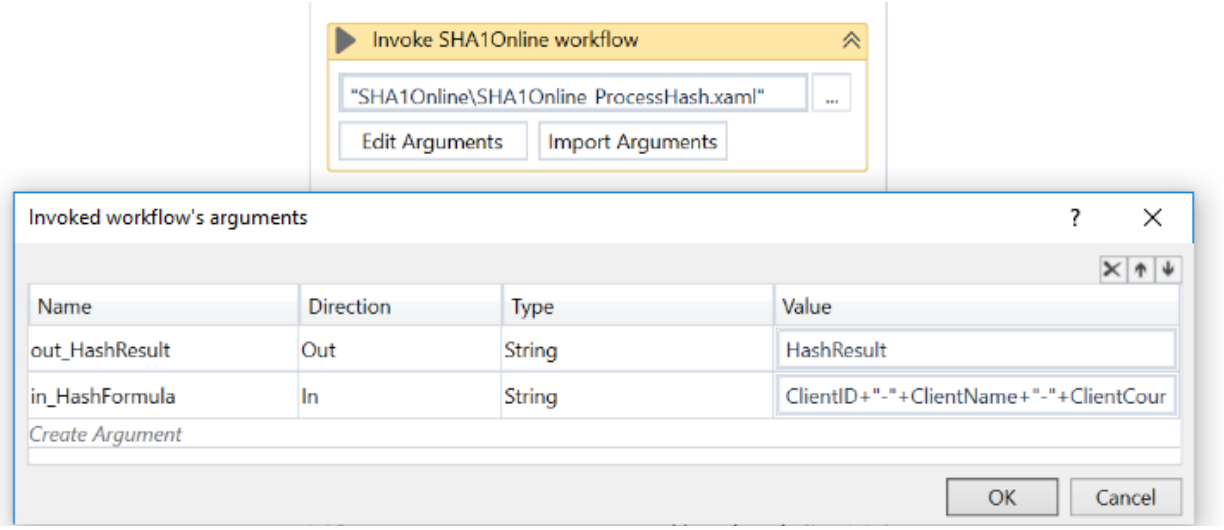
- В том же рабочем процессе нам необходимо извлечь значение идентификатора клиента, имени клиента и страны клиента. Вы уже должны быть знакомы с преобразованием строк, так что вам не составит труда извлечь каждое отдельное значение из текста.
- Откройте рабочий процесс **Process**.
    - Отредактируйте аннотацию.
    - Запустите рабочий процесс **System1_NavigateTo_WIDetails**. Импортируйте и привяжите аргументы.
    - Вот как это выглядит:



- Запустите рабочий процесс **System1_ExtractClientInformation**. Свяжите три выходных аргумента с локальными переменными.

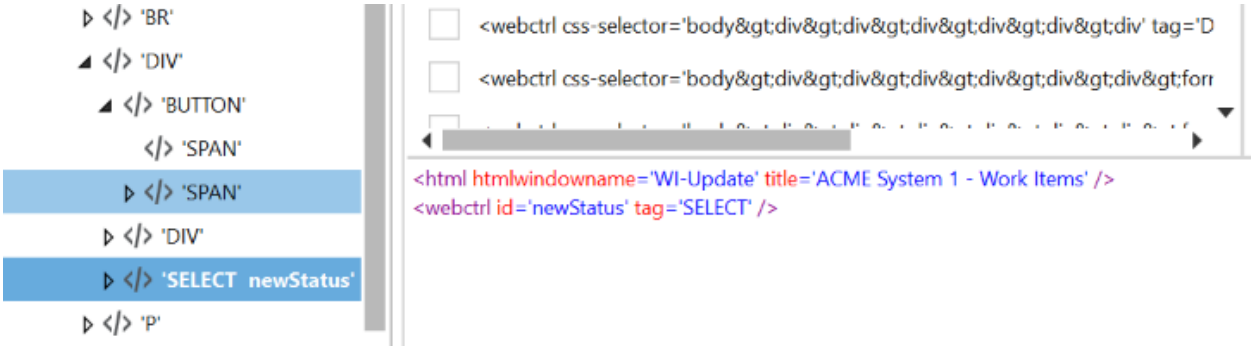
- Теперь создадим рабочий процесс для извлечения контрольной суммы из приложения SHA1Online.com. Создайте пустой рабочий процесс в папке SHA1Online. Назовите его SHA1Online_GetHashCode.
  - Нам нужен входной аргумент типа String для хранения формулы.
  - Для хранения вычисляемой контрольной суммы нам также нужен выходной аргумент типа String.
  - Добавьте необходимые действия для вычисления хэша. Это вы сделаете сами.
  - Наконец, нам нужно вернуться к начальной странице приложения, чтобы мы могли использовать ту же последовательность для обработки следующего элемента. Чтобы сделать это, добавьте действие Go Back ("Вернуться назад").
- Помните, что мы должны вычислить хэш-код для последовательности [ClientID]-[ClientName]-[ClientCountry], поэтому нам нужно составить эту строку. Используем выходное значение рабочего процесса **System1_ExtractClientInformation** как входной аргумент в рабочем процессе **SHA1Online_GetHashCode**.
- Вернитесь к рабочему процессу Process.
  - Добавьте действие **Invoke Workflow** и выберите SHA1Online_GetHashCode.

- Импортируйте аргументы. Используйте хэш-формулу в качестве входного аргумента. Создайте новую переменную для хранения результата хеширования.
- Рабочий процесс выглядит следующим образом:



- Создайте пустой рабочий процесс в папке System1, чтобы обновить рабочие элементы с помощью вычисляемой контрольной суммы. Лучше всего создать универсальный рабочий процесс, который можно будет повторно использовать в будущих проектах. Чтобы обновить рабочие элементы, нам просто нужно добавить комментарий, установить новый статус, а затем отправить изменения.
  - Начните новую последовательность, добавив аннотацию. Предусловие заключается в том, что страница сведений о рабочем элементе открыта, поэтому мы можем обновить рабочий элемент.
  - Добавьте два входных строковых аргумента, один для комментария, а другой для нового статуса.
  - Добавьте действие **Click** и установите кнопку **Update Work Item** в качестве его мишени. В панели **«Свойства»** (Properties) установите флажок в поле Simulate Click («Имитировать щелчок»).
  - Для заполнения поля «Комментарий» на странице **Update Work Item** («Обновление рабочего элемента») используйте действие **Type Into**. Кроме того, установите флажок **Simulate Type** («Имитировать ввод с клавиатуры») на панели **Свойства**.
  - Теперь необходимо обновить состояние этого рабочего элемента. Для этого перетащите действие **Select Item**. Нажмите кнопку **Indicate on Screen** ("Показать на экране"), а затем выберите раскрывающийся список в поле **New Status** ("Новый статус"). Возможно, вы получаете сообщение об ошибке, указывающее, что этот элемент управления не поддерживает выбор элемента. Это происходит потому, что у нас есть другие элементы пользовательского интерфейса, размещенные над элементом Select.

- Откройте **UiExplorer** и щелкните Select Target Element ("Выбрать целевой элемент управления"). Возвращаемый элемент может быть кнопкой (Button) или элементом Span, в зависимости от того, где вы щелкнули.
- Щелкните на поле **New Status**. Выберите элемент под Button или Span на панели Visual Tree и используйте созданный селектор для действия Select Item.
- Вот как это выглядит в UiExplorer:



- Наконец, задайте свойство Item аргументу in_Status.

  - Добавьте действие **Click** для кнопки Update Work Item. Убедитесь, что параметр **Simulate Click** включен.
  - Появится сообщение с подтверждением, поэтому нам нужно использовать еще одно действие **Click** для выбора кнопки OK. Здесь также должна быть включена опция **Simulate Click**.
  - Теперь можно закрыть окно Update Work Item, щелкнув кнопку "Закрыть" в правом верхнем углу.

- Вернитесь к рабочему процессу Process и вызовите только что созданный рабочий процесс. Убедитесь в том, что вы используйте правильные переменные в качестве значений входных аргументов.
- Наконец, нам нужно вернуть приложение в исходное состояние, чтобы мы могли обработать следующий элемент. Для этого необходимо вызвать рабочий процесс, созданный для перехода на страницу Панели управления (Dashboard).
- Мы закончили с реализацией процесса. Далее необходимо протестировать весь процесс. Вы уже должны были протестировать каждый отдельный рабочий процесс сразу после разработки, используя для аргументов значения по умолчанию.
  - Запустите рабочий процесс Main несколько раз и убедитесь в том, что каждый раз он выполняется корректно. Если нет, устраните неполадки и запустите снова.
  - Используйте параметр **"Сброс тестовых данных"** (Reset test data) в меню **Параметры пользователя** (User options) для создания нового набора данных для целей тестирования.
  - Проверьте функциональность повторной попытки. Установите параметр MaxRetryNumber в config.xlsx на 1 и помешайте роботу, нажав на другой ссылке меню, то есть на ссылку Home сразу после того, как робот откроет текущий рабочий

элемент. Из-за этого кнопка "Обновить рабочий элемент" (Update Work Item) будет недоступна и возникнет системная ошибка, поскольку элемент пользовательского интерфейса не найден Процесс повторно инициализируется, и выполнение возобновляется с рабочего элемента, на котором возник сбой. При повторном вмешательстве процесс прекращается по мере достижения максимального числа повторных попыток.

Примечания по реализации процесса.

Мы начали с извлечения списка всех элементов, которые должны быть обработаны с помощью Мастера считывания данных (Data Scraping wizard). Задайте себе такой вопрос: "Что произойдет, если у вас есть большой набор транзакций и действие Extract Data в состоянии Init state не срабатывает из-за истечения время ожидания браузера? Как мы можем улучшить код, чтобы усовершенствовать обработку ошибок?"

- Мы обрабатывали по одному элементу за раз. Все элементы являются независимыми друг от друга, так что мы могли бы также обрабатывать их параллельно, используя несколько роботов. В следующем упражнении мы увидим, как функциональность очередей Orchestrator может быть использована для реализации процесса, в котором рабочие элементы распределяются между роботами и обрабатываются только один раз, параллельно.