# 自動化の手引き
# 顧客のセキュアハッシュ算出

# 自動化の手引き – 顧客のセキュアハッシュ算出

- ReFramework テンプレートを使用します。
    - 最初は Orchestrator のキューを使用せず、簡単な実装例で ReFramework を理解することを目的とします。
    - はじめに、ACME System 1 の Work Items に目を通してください。Work Items からデータを抽出するためには、Data Scraping ウィザードを使用し、テーブル全体を取り出します。
    - キューを使用する場合、キュー内に新たに追加された 1 つのトランザクションは、この Work Items 上から読み込んだデータの行を指します。
    - この方法は、入力データが Excel スプレッドシートや CSV ファイル、データベースから取り出したデータテーブルである場合にも応用可能です。
- 変更が予想される値は Config ファイルに保存します。Data フォルダ下にある Config.xlsx を開き、シート「Settings」上で設定を行います。
    - ACME System 1 の URL と SHA1 Online URL 用の設定を追加します。
    - ACME System 1 では資格情報を求められます。そのため今回は、Orchestrator のアセットを使用し、ACME System 1 の資格情報を保存します。資格情報名を保存する、Credential 型のアセット（アセット名「System1_Credential」）を追加し、ACME System 1 のユーザー名とパスワードを書き込みます。
    - シート「Setting」にも同様に追加します。
      Orchestrator 上のアセット名・Config ファイルの System1_Credential に対する Value

は、一致している必要があります。

- **Config** ファイルのシート「**Constants**」で、MaxRetryNumber の値を 2 に設定します。このパラメーターは、ReFramework を使用中にシステム例外が発生した場合、次の処理に進むまでの試行回数を指します。

- Framework 内にて次の変更を行います。

    - Data Scraping ウィザードにおいてテーブル全体を取り出し、それを 1 つの処理として一括処理するため、データーテーブル型から DataRow 型に変更します。**Main** ファイル内の変数「**TransactionItem**」を System.Data.DataRow に変更します。

    - また、ワークフロー「**GetTransactionData**」、「**Process**」、「**SetTransactionStatus**」のそれぞれの引数タイプを「**TransactionItem**」と同様の変数の型になるように変更します。

    - Orchestrator のキューは今回は使用しないため、「**SetTransactionStatus**」ワークフローから **SetTransactionStatus** アクティビティを 3 つ削除します。

    - この演習では、**ACME System 1** と **SHA1-Online.com** という 2 つのアプリケーションを使用するため、ルートディレクトリに “System1” と “SHA1Online” というフォルダーを 2 つ作成します。このフォルダーは、2 つのアプリケーションで作成したワークフローに使用します。

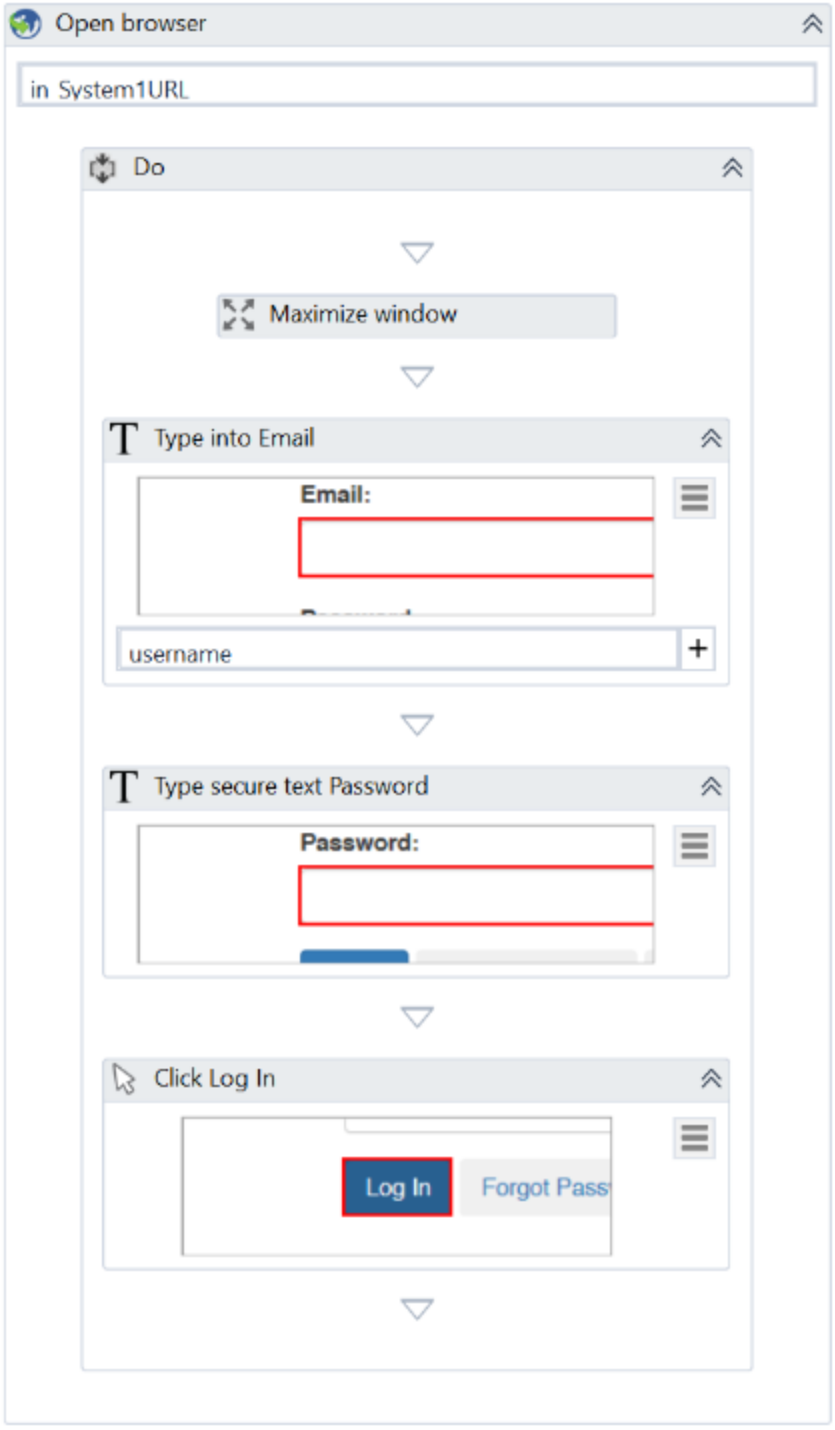
- System1 フォルダーに、System1 ログインプロセス用の空のシーケンスを作成します。他のワークフローにも使用できるように共通部品（再利用可能なコンポーネント）を作成します。

    - 新しいシーケンスは、ワークフローの目的を説明する短めの説明文を注釈（Annotation）として追記します。
      使用される引数や前提条件、続いて実行される操作などの説明文を追記します。

    - **ACME System 1 URL** と、**System1 Credential** に対し、入力引数を作成します。

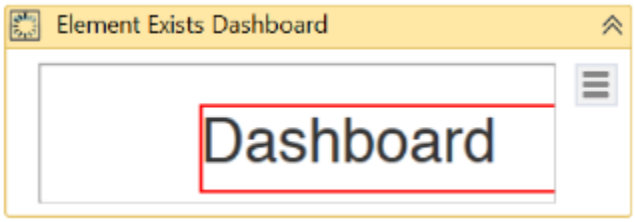
    - 「**Framework/GetAppCredential**」ワークフローを呼び出します。

    - シーケンスは次のスクリーンショットの通りになります。

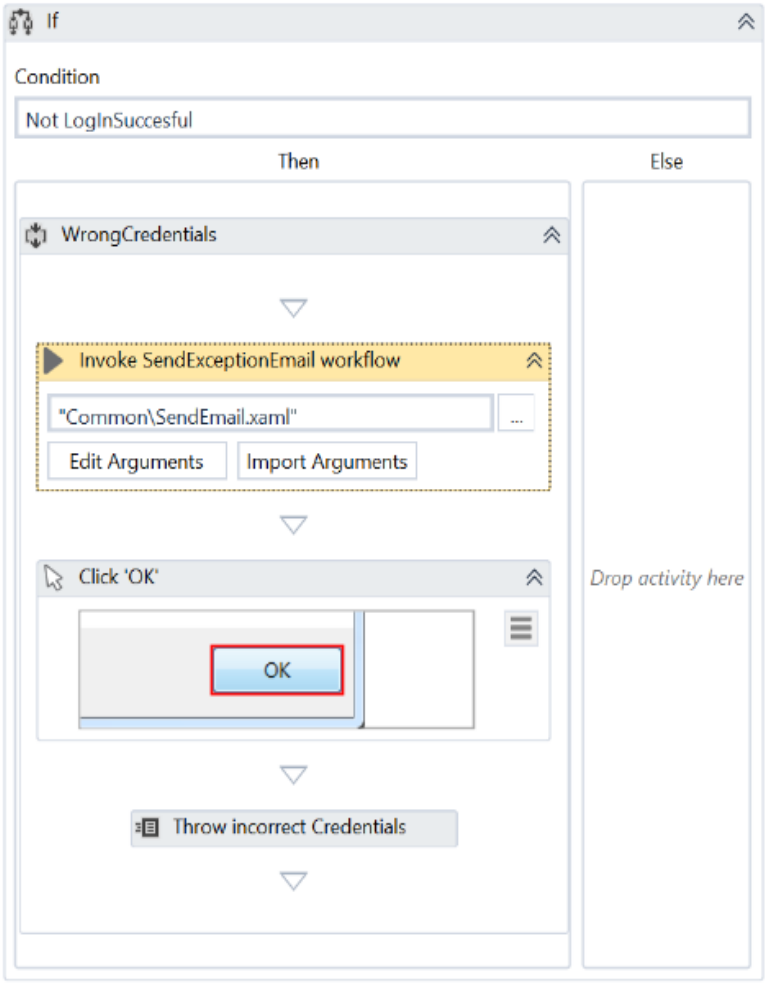

Open browser
in System1URL
Do
Maximize window
Type into Email
Email:
username
Type secure text Password
Password:
Click Log In
Log In
Forgot Pass

- 業務プロセス指示書にも記載があるように、プロセスの実行中に発生する可能性のある例外処理を考慮しましょう。今回はログインの可否をチェックして、間違った資格情報でログインが行われた場合は、誤りがあることを通知します。

- **Element Exist** アクティビティを使用し、この事例のみに表示される要素を検索してログインの可否をチェックします。

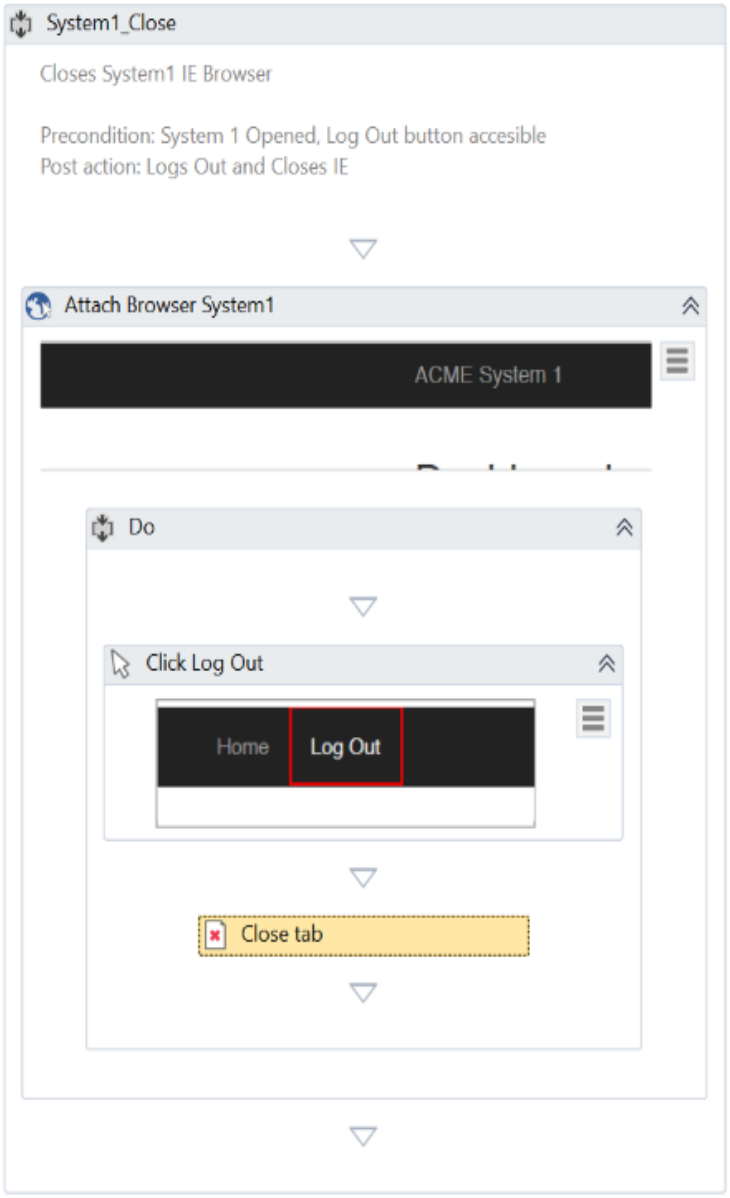



- **If** アクティビティを使用し、ログインの可否を確認します。失敗した場合は、次のアクションを実行します。
    - 例外処理としてメールを送信する。この場合、共通部品となるワークフローを **Then** セクションで呼び出す方法を推奨します。
      これは、（**SendEmail.xaml**）と名付けます。
    - **Click** アクティビティを使用しエラーメッセージを終了する。
    - ACME System 1 に誤った資格情報が渡された場合に対する例外を出力してプロセスを停止する。
- **If** アクティビティは次のスクリーンショットの通りになります。




- 引数の既定値を使い、作成されたファイルの単体テストを実施します。

- 次に、ACME System 1 からログアウトして終了するというワークフローを作成します。シーケンスは次のスクリーンショットの通りになります.

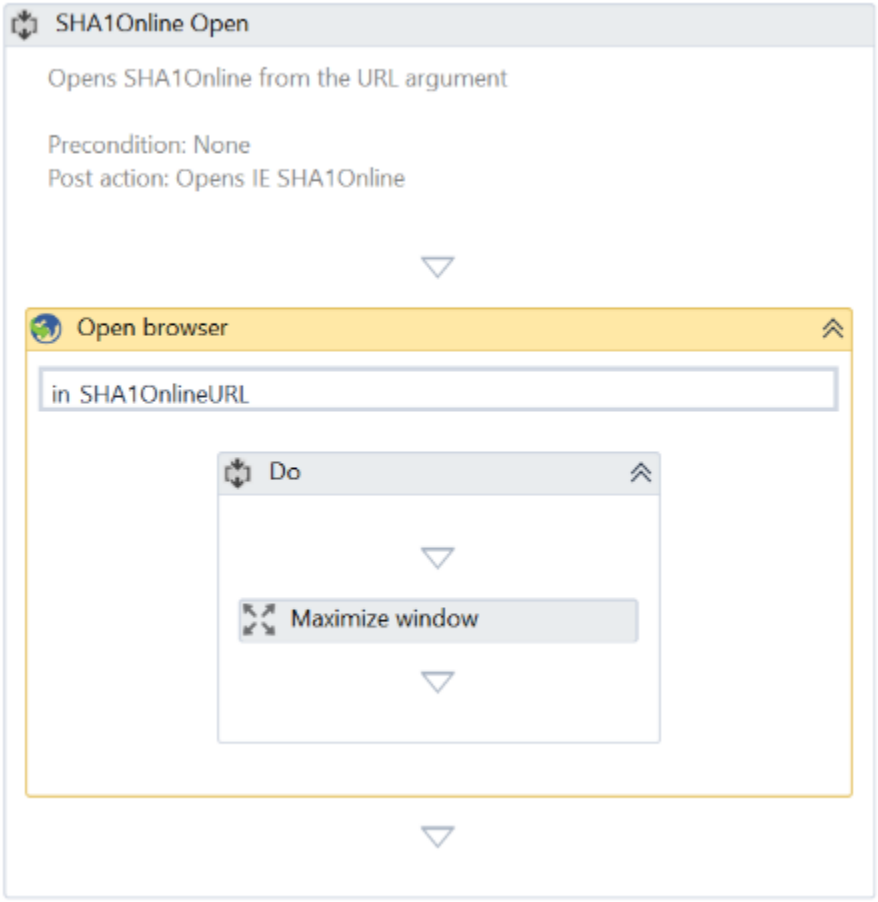



- 次に、SHA1 Online プロセスで使うもう 1 つのアプリケーション用に同じ手順を実行します。

- SHA1 Online フォルダーに、アプリケーション起動用の空のシーケンスワークフローを作成します。資格情報を求めるアプリケーションではないため、このシーケンスはシンプルです。

- 説明文（注釈）の追加を行います。
- プロジェクトのメンテナンスを簡易にするため、アプリケーションの URL は引数を使用し、ワークフローに渡します。URL の値は Config ファイルに保存します。ACME System 1 へも同様のプロセスでログインが可能です。

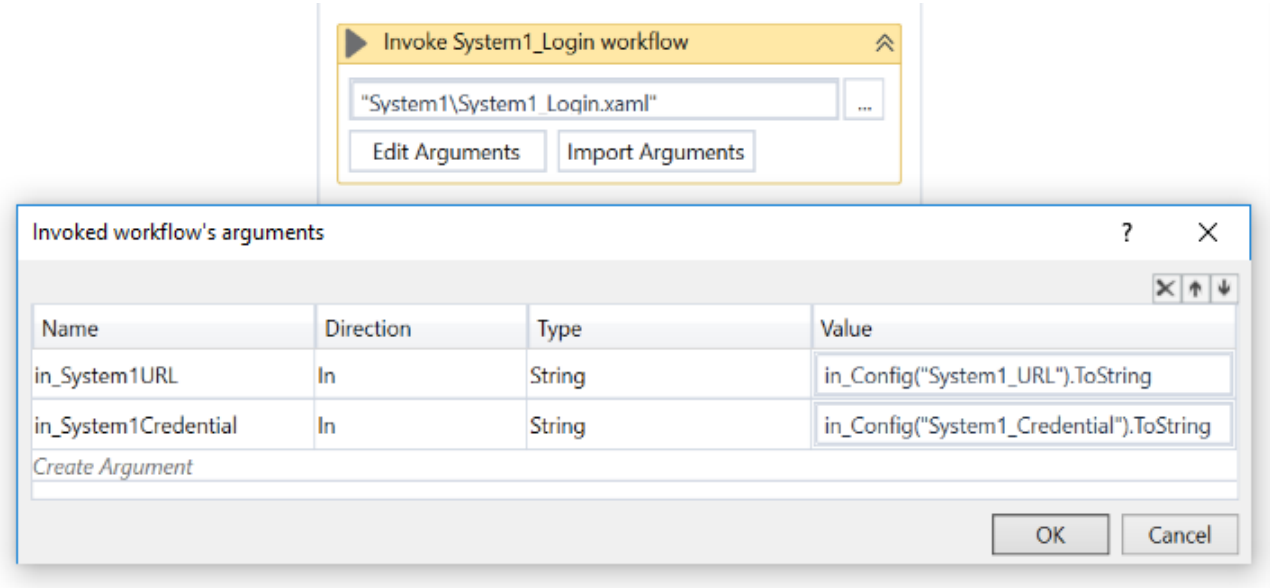
- シーケンスは次のスクリーンショットの通りになります。




- SHA1 Online アプリケーションの終了用のシーケンスを作成します。

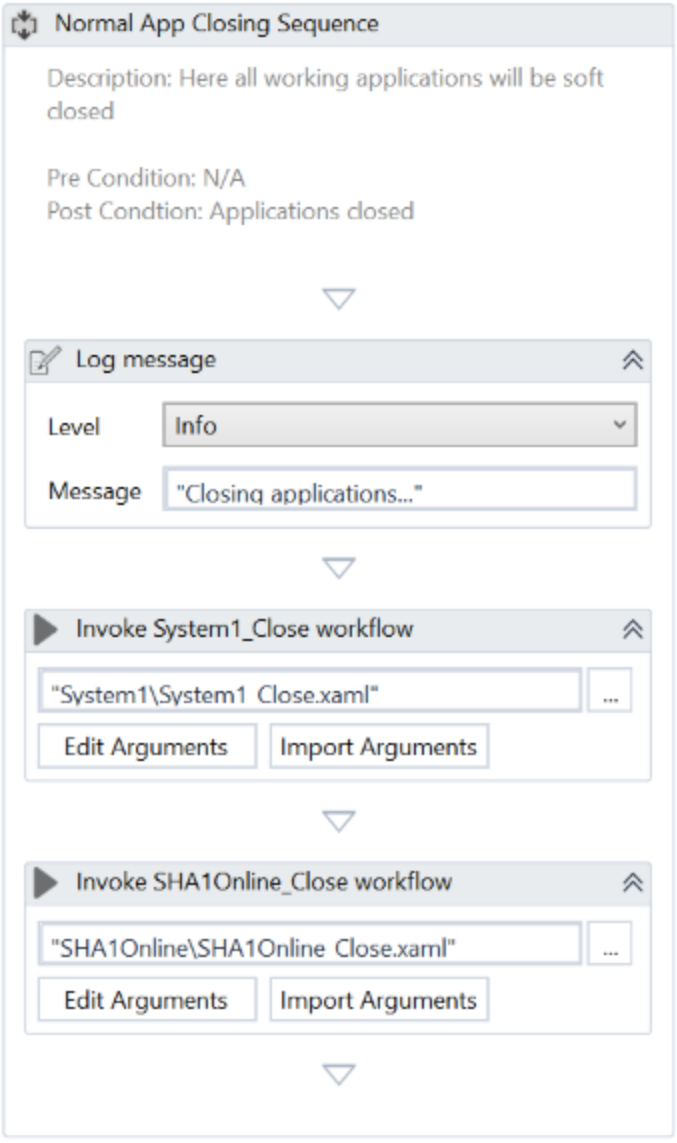
- 2 つのアプリケーションを起動、終了するワークフローの準備ができたところで、Framework の初期化と終了部分に変更を行います。

- 「**Framework/InitAllApplications**」ワークフローを開きます。
  - System1_Login ファイルをドラッグ＆ドロップすると、**Invoke Workflow File** アクティビティが自動で作成されます。
  - **Import Arguments** をクリックして、値を Config ファイルに取り込みます。
  - **Invoke Workflow File** アクティビティは次のスクリーンショットの通りになります。




  - SHA1Online_Login ワークフローをドラッグ＆ドロップします。
  - SHA1Online_Login ワークフローに対しても **Import Arguments** をクリックし、URL の引数を Config ファイルから取り込みます。

- 「**Framework/CloseAllApplications**」を開きます。
  - System1 と SHA1 Online アプリケーションの終了用に作成した 2 つのワークフローをドラッグ＆ドロップします。
  - 完成したシーケンスは次のスクリーンショットの通りになります。



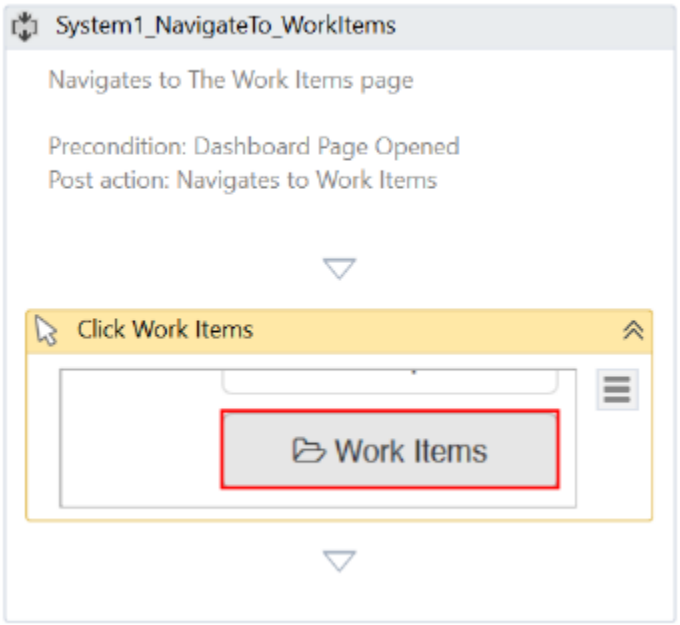

- 「**Framework/KillAllProcesses**」ワークフローを開きます。

- 2 つのアプリケーションをウェブブラウザで起動します。**Open Browser** アクティビティでは、BrowserType プロパティ上で、使用するブラウザの変更を行っていなければ Internet Explorer が使われます。

    - **Kill Process**
    - アクティビティを使用して、Process Name に “iexplore” を設定します。

- ACME System 1 アプリケーションの **Work Items** を表示する空のシーケンスワークフローを ACMESystem1 フォルダーに作成します。「Work Items」ページへの移動は別のプロジェクトでも使用することが考えられるため、独立したワークフローを作成し、共通部品として利用します。

    - 説明文（Annotation)を追加します。
    - 「Work Items」をクリックするために、**Click** アクティビティを使います。このアクティビティは Attach Browser のスコープ外にあるため、全体セレクタを使用します。
    - プロジェクトは次のスクリーンショットの通りになります。




- トランザクションの処理を開始する前に、入力データを Init State で読み取るために必要なアクティビティを追加します。入力データは、ウェブサイトに保存されているため、以下の追加手順が必要です。

- Type が WI5 となっている項目ごとにセキュアハッシュコードを算出するには、まずは WI5 タイプのみの項目リストが必要とされます。

- System1 フォルダーに空のシーケンスワークフローを作成後、ACME System 1 の Work Items 上の全てのデータを一括でデータテーブル型の変数に保存し、後に、Type が WI5 となっている処理項目のみに絞り込みます。

    - Data Scraping ウィザードを使って HTML テーブル全体を取り込みます。データが複数ページにまたがる場合「Yes」を選択し、次のページボタンを選択します。

    - **Maximum number of results** のオプションに 0 を設定することで、全範囲を出力データとして取り出すことが可能です。

    - 出力引数を作成し、取り出したデータテーブルの値を割り当てます。
    - 説明文（注釈）の追加を行います。

- **Main** ワークフローより、**「Init」**ステータスを展開表示します。

    - Entry 領域で、KillAllProcesses ワークフローが呼び出される箇所に対して以下の作業を行います。

    - **Invoke KillAllProcesses** アクティビティの後に、入力用のトランザクション・データテーブルを読み取る新しいシーケンスを追加します。

    - このシーケンスでは、これまでに作成したワークフローのうち次の 4 つの処理を行うワークフローを呼び出します。

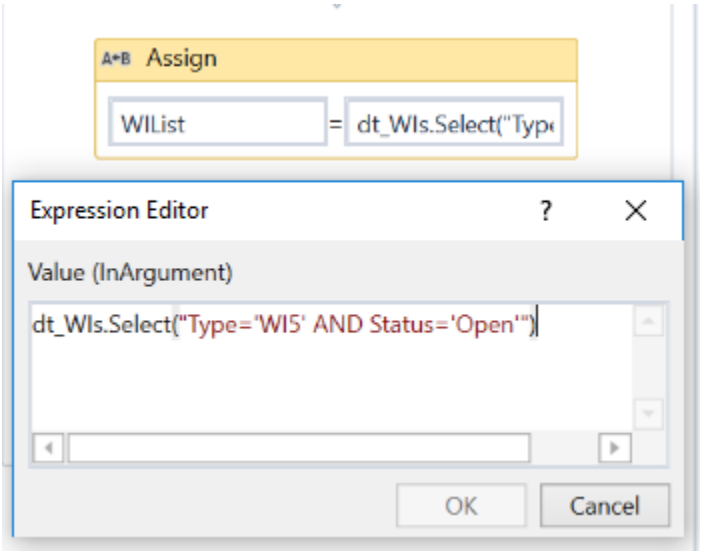
        - System1 Login
        - System1 Navigate to Work Item
        - System1 Extract Work Item Data Table
        - System1 Close

    - 必要に応じて引数の読み込みを行います。

- WorkItem 上のリストより、必要なリスト、すなわち Type が WI5 かつ Status が「Open」となっている項目を取り出します。

  - DataTable.Select メソッドを使って、上記の条件に合わない要素を除外します。結果を新しい変数に代入します。

  - **Assign** アクティビティは次のスクリーンショットの通りになります。




- 変数「TransactionNumber」（トランザクションインデックス）は 1 から始まります。一方、DataRow オブジェクトは、配列の中の 1 つの要素となります。配列は 0 から開始されるため、配列のインデックスは常に全てがトランザクションインデックスより 1 小さくなります。

● **「Framework/GetTransactionData」**ワークフローを開きます。

- 現在のプロセスに合わせて説明文を変更します。

- Work Items の処理項目は、配列形式のオブジェクトであるため、入力引数を設定します。

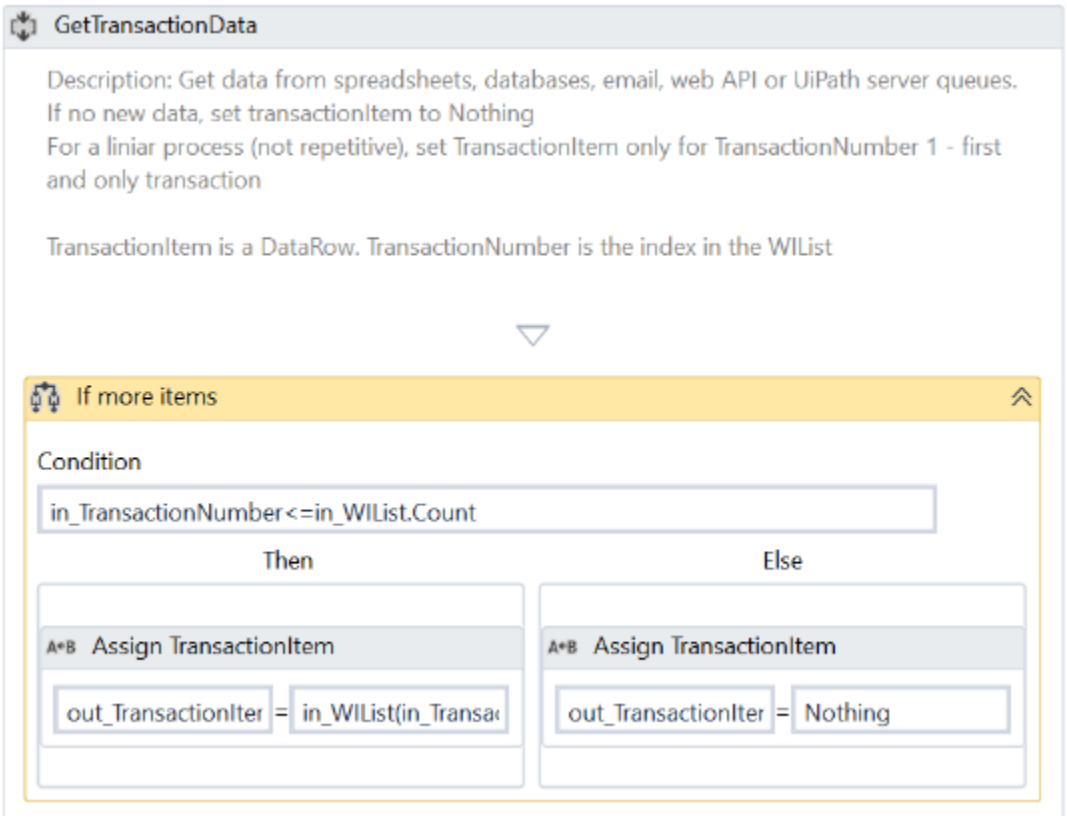
- 変数「TransactionItem」は配列形式の 1 つのオブジェクトです。新たに処理を行うには、トランザクションインデックスの値が、配列のインデックスの数字と同等以下である必要があります。

- **If** アクティビティを追加し、処理対象である新しいトランザクションがあるかどうかを確認します。

  - **Then** セクションで、**Assign** アクティビティを使い、出力引数である Transaction Item の値をインデックスに応じて設定します。

  - **Else** セクションで、Transaction Item の値に Nothing を設定します。

- **Get Transaction Data** のワークフローは次のスクリーンショットの通りになります、




- ログとして残すことを目的に、処理対象の Transaction Items が残っている場合は、Transaction ID を Work Item ID の値に設定します。

アクティビティは既に存在します。ここでは **Assign** アクティビティで TransactionID の値を「out_TransactionItem("WIID").ToString」に変更します。

- 以上で Framework の構成は終了です。**Main** ワークフローを実行し、入力データを取り出すことができるかどうか検証してみましょう。Output パネルを使うとデータの正確性を確認することができます。

次に、Work Items を処理するワークフローの作成に進みます。

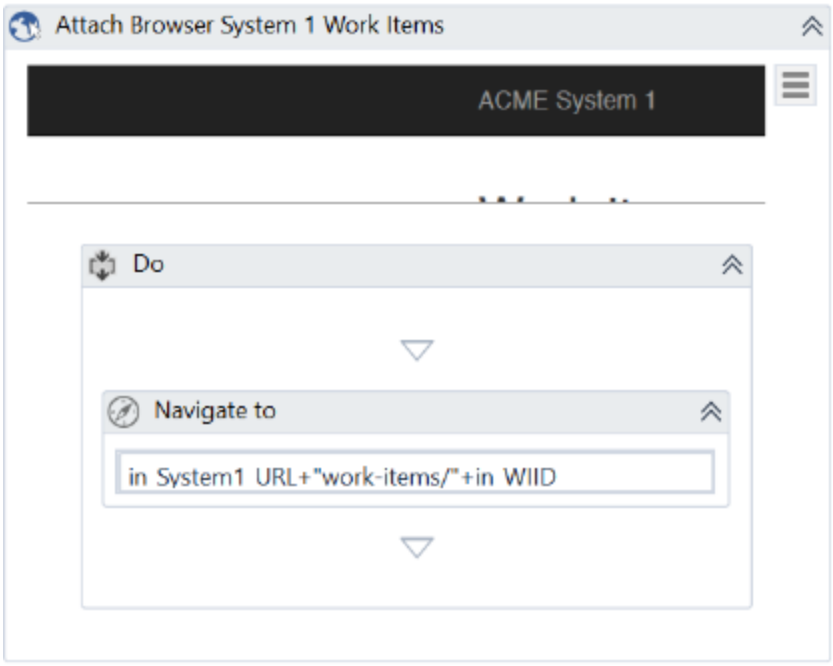
- **System1** フォルダーに、「Work Item Details」ページを表示する空のシーケンスを作成します。**System1_NavigateTo_WIDetails** という名前をつけます。

- Work Item ID を URL の後に追記することで、「Work Item Details」に移動にすることができます。/www.acme-test.com/work-items/ “Work Item ID” という構成です。

    - Work Item ID と System1 URL それぞれに、入力引数を 2 つ作成します。Work Item ID は Int32 型、System1 URL は String 型で作成します。

    - System1 Dashboard にアタッチしたら、**Navigate To** アクティビティを使って「Work Item Details」ページにアクセスします。

    - シーケンスは次のスクリーンショットの通りになります。



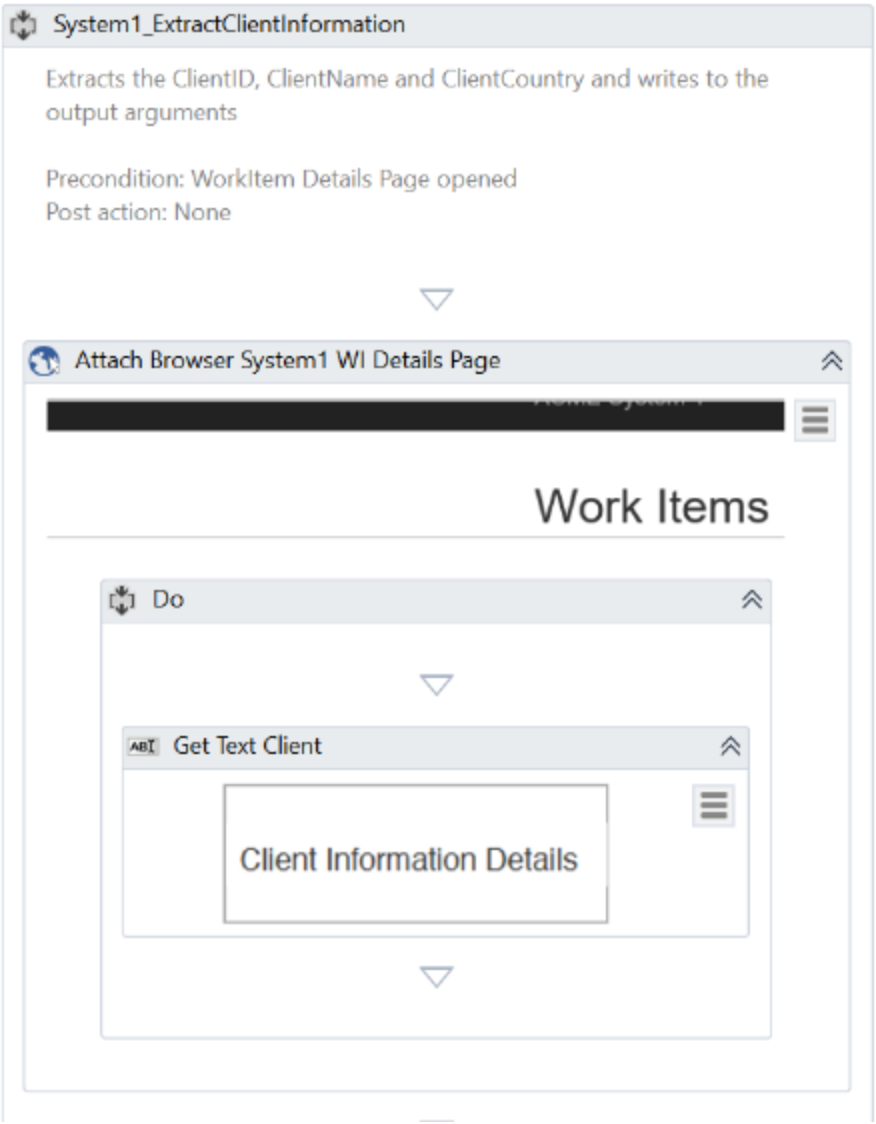

- System1 フォルダーに **System1_ExtractClientInformation** という名前で新たにワークフローを作成します。このワークフローを使用し、処理項目ごとの詳細情報を取り出します。

    - このワークフローに入力引数はありません。必要となる前提条件は、「Work Item Details」ページはすでに開かれているという点のみです。

    - Client ID・Client Name・Client Country という出力引数を作成します。

    - ウェブページからテキストを取り出す前に、対象ページが読み込み完了していることと、ページ上の要素が取得可能であることを確認する必要があります。
      そのためには、**Attach Browser** アクティビティを追加し、**Do** セクションでは **Get Text** アクティビティを使います。プロパティ上の **WaitForReady** に **Complete** がチェックされていることを確認します。

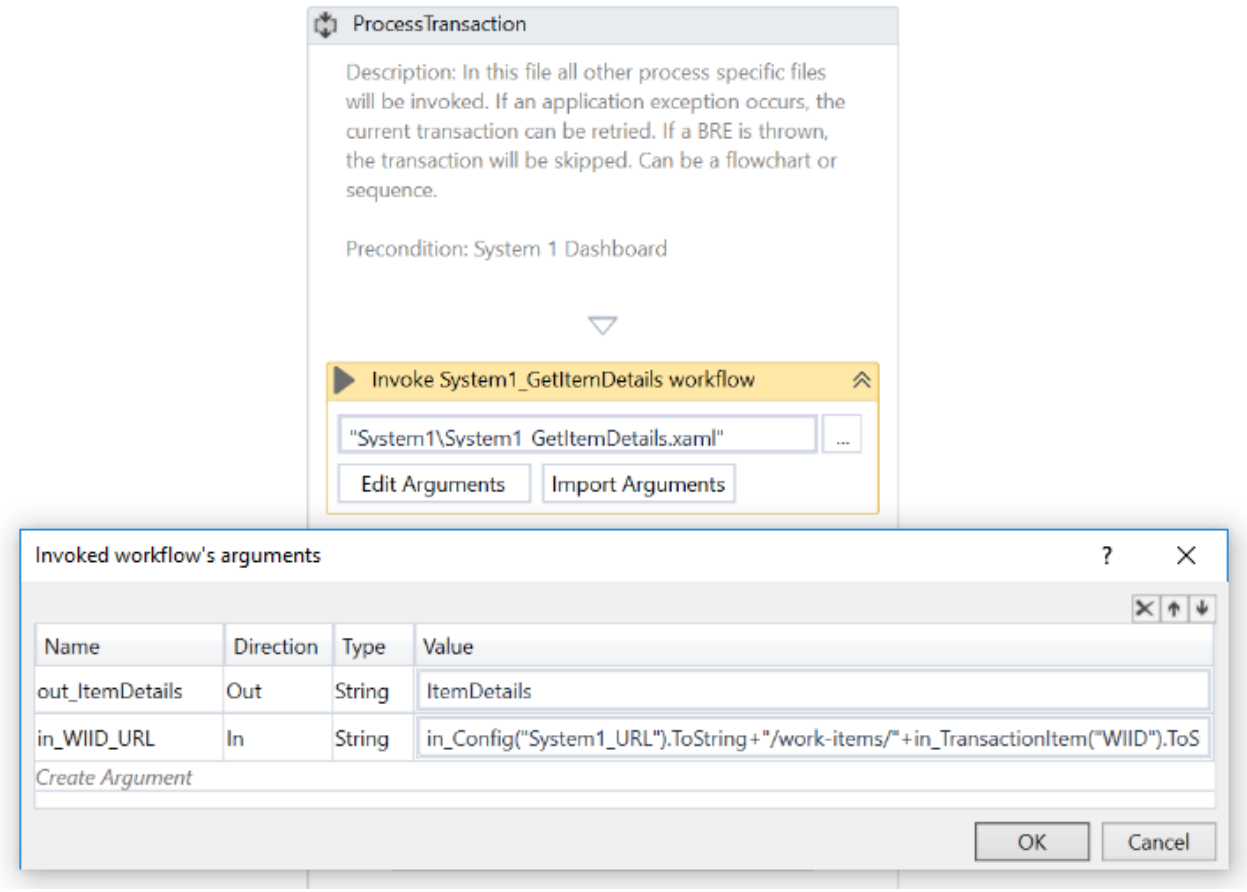
    - ワークフローは次のスクリーンショットの通りになります。




- 同じワークフロー内で、Client ID・Client Name・Client Country の値を取り出す必要があります。それぞれの値をテキストから抽出します。

- **Process** ワークフローを開きます。

    - プロセスに合わせて説明文を追加します。
    - **System1_NavigateTo_WIDetails** ワークフローを呼び出します。その次に、引数を取り込みます。
    - 結果は次のスクリーンショットの通りになります。




- **System1_ExtractClientInformation** ワークフローを呼び出します。3 つの出力引数をローカル変数に取り込みます。

- 次に、SHA1Online.com アプリケーションからセキュアハッシュコードを算出するワークフローを作成していきましょう。SHA1Online フォルダーに空のシーケンスワークフローを作成します。**SHA1Online_GetHashCode** という名前を付けます。

  - ハッシュコード算出するには、ハッシュ関数として（[ClientID]-[ClientName]-[ClientCountry]）String 型の入力引数が必要です。

  - また、算出したセキュアハッシュコードを保存するための String 型の出力引数も必要です。

  - セキュアハッシュコード算出に必要なアクティビティを追加します。

- 同じシーケンスを使用して次の処理を行うために、アプリケーションの最初のページに戻る必要があります。それには、**Go Back** アクティビティを追加します。

- ハッシュコードは **[ClientID]-[ClientName]-[ClientCountry]** という形式で算出しなければいけません。そのため、文字列を構築する必要があります。

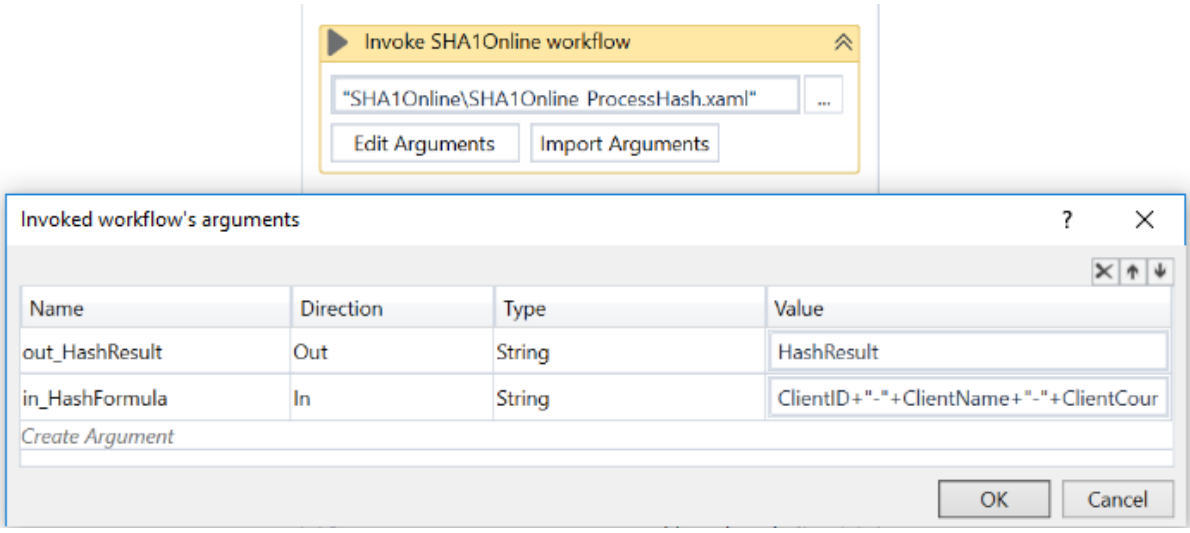
- **System1_ExtractClientInformation** ワークフローの出力値を **SHA1Online_GetHashCode** ワークフローの入力引数として使うことにしましょう。

- **Process** ワークフローに戻ります。

  - **Invoke Workflow** アクティビティを追加して、SHA1Online_GetHashCode ワークフローを呼び出します。

  - 引数を取り込みます。入力引数としてハッシュ関数を使用します。結果を格納する新しい変数を作成します。

  - ワークフローは次のスクリーンショットの通りになります。




- 算出したセキュアハッシュコードを、Work Items を更新する新規のワークアイテムを System1 フォルダーに作成します。このワークフローも、その他のプロジェクトにも流用できる独立したワークフローとして

作成することをお勧めします。Work Items の更新に必要なのは、コメントの追加・新しいステータスの設定・変更完了の送信です。

- 説明文を追加して新しいシーケンスを開始します。前提条件は、Work Item を更新できるように、「Work Item Details」ページがすでに開かれていることです。

- String 型の入力引数を 2 つ追加します。1 つはコメント用、もう 1 つは新しいステータス用です。

- **Click** アクティビティを追加して、**Update Work Item** ボタンをターゲットとして設定します。また、**Properties** パネルの Simulate Click を有効にします。

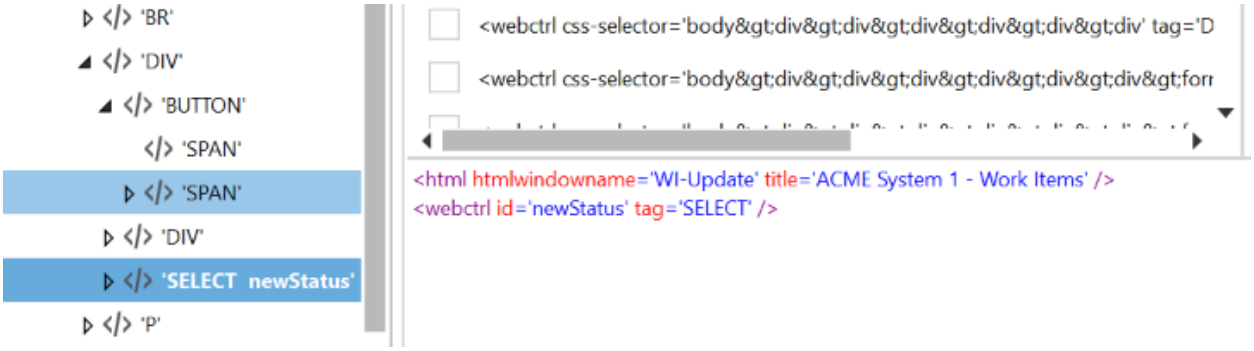
- **Type Into** アクティビティを使って、**Update Work Item** ページ上のコメントフィールドに入力します。Properties パネルの Simulate Click を有効にします。

- WorkItems の Status を更新します。**Select Item** アクティビティを使用し、**Indicate on Screen** より、**New Status** フィールドのドロップダウンボックスを指定します。
  しかし、**Select Item** に対応していないというエラーメッセージが表示されるはずです。これはは HTML Select 要素の上に他の UI 要素があることが原因です。

  - **UiExplorer** の Select Target Element をクリックして検出される UI 要素は、クリックした場所にもよりますが、Button または Span UI 要素になります。

  - **New Status** フィールドをクリックします。Visual Tree パネルで Button または Span の下にある要素を選択し、Select Item アクティビティに出力されたセレクタを使用します。

  - UiExplorer の画面表示は次のスクリーンショットの通りになります。




- 最後に in_Status 引数の Item プロパティを設定します。

  - Update Work Item ボタン用に **Click** アクティビティを追加します。また、**Simulate Click** オプションが有効になっていることを確認します。

  - 確認メッセージが表示されたら、別の **Click** アクティビティを使って OK ボタンを選択する必要があります。この場合も、**Simulate Click** オプションを有効にしてください。

  - Update Work Item ウィンドウは、右上コーナーにあるクローズボタンのクリックで閉じることができます。

- Process ワークフローに戻り、新しく作成したワークフローを呼び出します。入力引数の値として正しい変数が使われていることを確認してください。

- 最後に、次の処理に移行するためにアプリケーションの初期状態に戻る必要があります。Dashboard ページに移行するためのワークフローを作成し呼び出します。

- 以上で自動化の実装は完了です。次は、プロセス全体の動作検証が必要です。個々のワークフローについては、引数の既定値を使ってそれぞれ動作検証を行いましょう。

  - Main ワークフローを何度か実行して、その都度正しく実行されることを確認します。

  - **User options** メニューの **Reset test data** オプションを使うと、新しいテスト用データ一式が作成されます。

- 再試行機能が正しく動くか検証しましょう。Config.xlsx の MaxRetryNumber パラメーターを 1 に変更し、ロボットが現在の Work Item をオープンした直後に別のリンクメニュー／Home リンク等をクリックして、ロボットの動作を妨げることで検証します。

この作業により、Update Work Item ボタンが無効となり、UI 要素が見つからく　　なることからシステム例外が発生します。

- システム例外が発生した場合は、再初期化が行われ、失敗した作業項目から実行が再開されます。ロボットの動作を妨げた場合、指定した再試行回数に達し、プロセスは停止します。

# まとめ

処理対象となる項目一覧の取得は、主に Data Scraping ウィザードを使って行いましたが、それには以下の課題を解決する必要がありました。

「大量のトランザクションがあるため、ページの読み込みに時間がかかり、「Init」ステートの Extract Data アクティビティが失敗した場合はどうなるか。エラー処理の効率性をあげるためには仕様をどのように変更すればよいか。」

今回は、Work Item 上の対象処理項目を一つずつ処理しています。
それぞれの処理項目は独立しており依存関係にないため、複数のロボットを使用して処理を行うことができました。