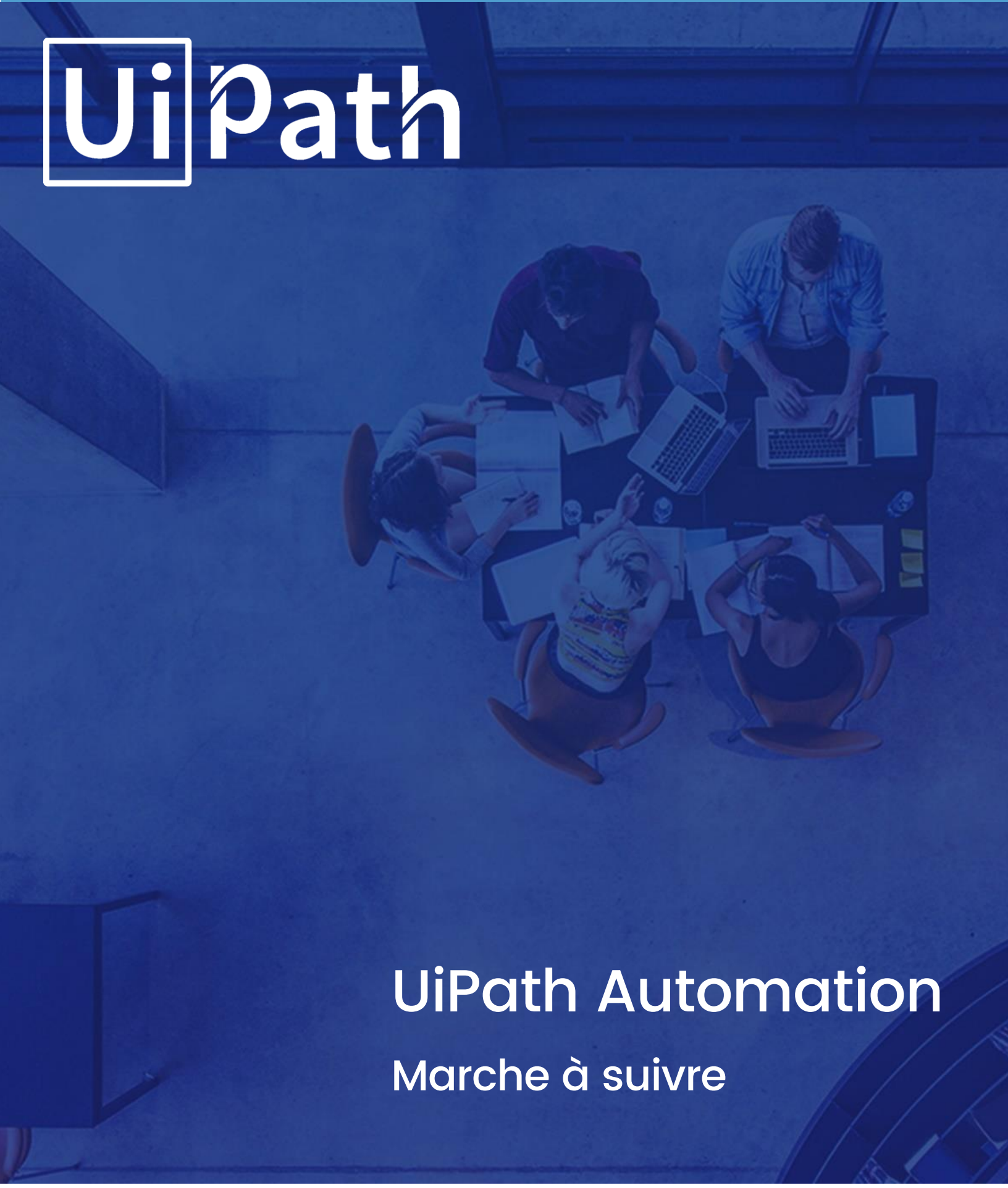


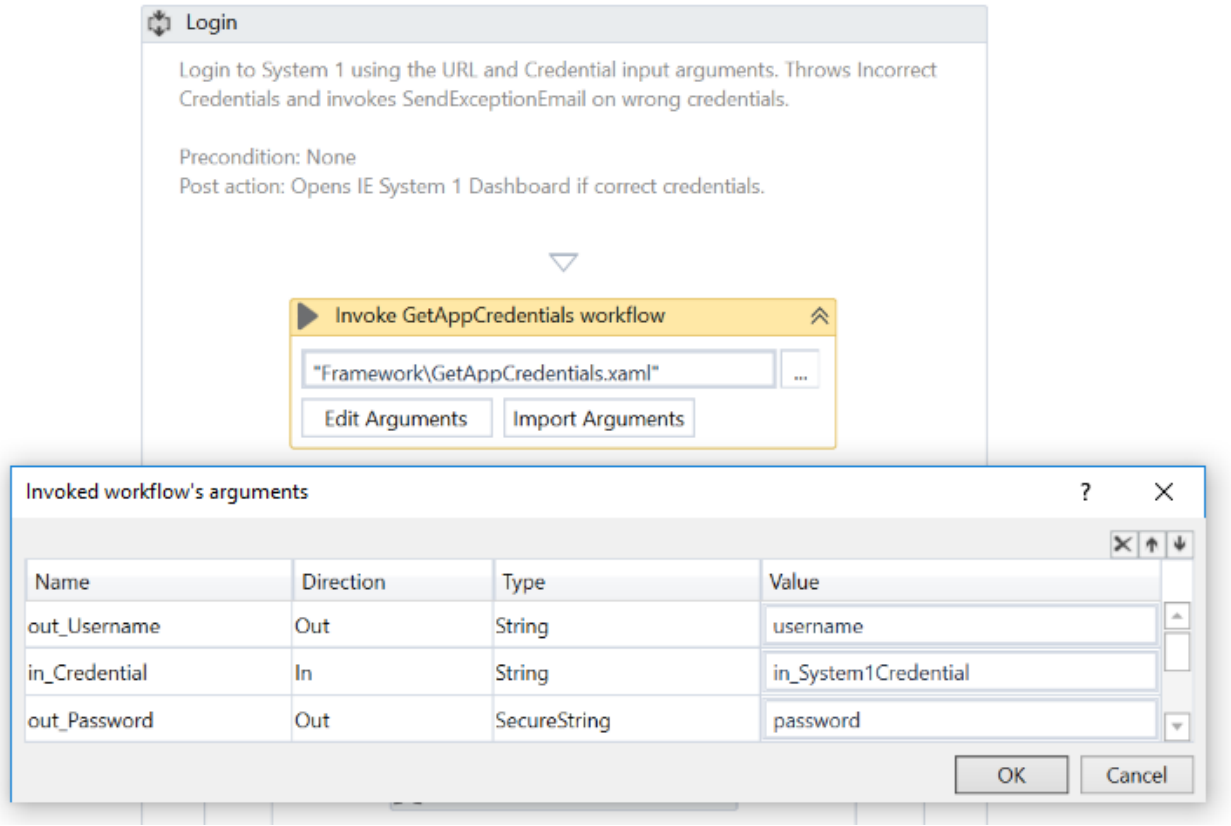
UiPath Automation

Marche à suivre

Marche à suivre : calcul du hash de sécurité du client

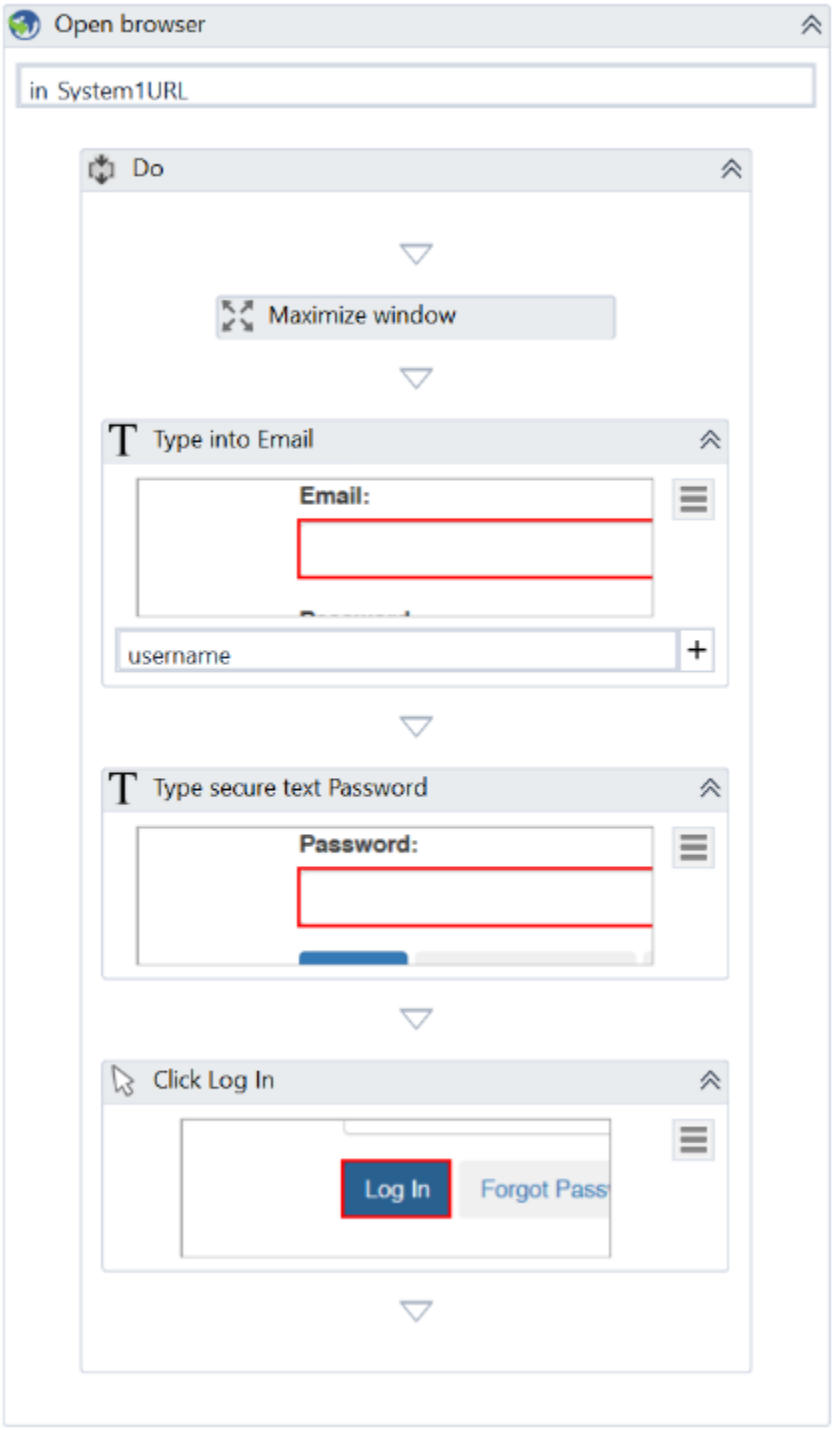
- Commençons avec le modèle REFramework.
    - Nous débutons par une simple mise en œuvre visant à faire la démonstration de REFramework sans utiliser Orchestrator Queues.
    - Examinez les données d'entrée (liste des éléments de la file d'attente) : nous pouvons extraire tout le tableau avec l'assistant Data Scraping.
    - L'élément opération se présente sous la forme d'une ligne de données dans la liste. La même démarche s'applique si l'entrée est un tableau de données extrait d'un tableur Excel, d'un fichier .csv ou d'une base de données.
- Il est recommandé de conserver dans le fichier de configuration les valeurs susceptibles de varier. Ouvrez le classeur Data\Config.xlsx et passez à la feuille Settings.
    - Ajoutez les paramètres de **System1 URL** et de **SHA1 Online URL**.
    - L'application System1 exige une authentification ; nous nous servirons des Orchestrator Assets pour stocker les identifiants de System1. Ajoutez un autre paramètre, System1_Credential, pour stocker le nom Credential.
    - Ajoutez un Asset de type Credential et écrivez le nom d'utilisateur et le mot de passe de System1. Vérifiez que le nom Asset coïncide avec la valeur du paramètre System1_Credential.
- Passez à la feuille **Constants** du classeur **Config** et attribuez la valeur « 2 » à MaxRetryNumber. Ce paramètre contrôle le nombre de fois que le cadre essaie de traiter un élément de travail lorsqu'il échoue avec une exception d'application, avant de passer à la suivante.
- Apportez les modifications suivantes dans le cadre.
    - La variable **TransactionItem** du fichier **Main** doit être de type System.Data.DataRow, puisque nous sommes en train d'extraire le tableau tout entier pour en traiter chaque ligne l'une après l'autre. Il faut également modifier le type d'argument des processus **GetTransactionData**, **Process** et **SetTransactionStatus** en le remplaçant par TransactionItem.
    - Retirez les trois activités **SetTransactionStatus** du processus SetTransactionStatus, dans la mesure où nous n'utilisons pas la fonctionnalité de gestion des opérations fournie par Orchestrator.
- Dans cet exercice, nous recourons à deux applications, **ACME System1** et **SHA1-Online.com**. Créez deux dossiers, « System1 » et « SHA1Online » dans le répertoire racine des solutions, et utilisons-les avec les processus créés pour les deux applications.

- Créons une séquence vide dans le dossier System1, pour le processus de connexion à System1. Nous souhaitons créer un composant réutilisable que l'on puisse employer avec de nombreux identifiants différents.
    - Il est recommandé de débuter votre nouvelle séquence par une courte annotation expliquant la finalité du processus. C'est ce que nous allons faire à l'avenir dans tous les fichiers. L'annotation commence par la description comprenant les arguments employés, un prérequis, et une action Post.
    - Créez deux arguments In : l'un pour **System1 URL**, l'autre pour **System1 Credential**.
    - Invoquez le fichier du processus Framework\GetAppCredential. Servez-vous de l'argument System1 Credential pour entrée. Créez deux variables pour stocker les deux valeurs de sortie (nom d'utilisateur et mot de passe).
    - Voici à quoi doit ressembler la séquence :



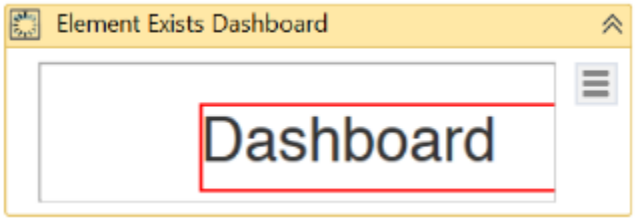
    - Appliquez une activité **Open Browser** pour naviguer jusqu'à l'URL de System1. Par défaut, Internet Explorer est utilisé, sauf si la propriété **BrowserType** est éditée.
    - En option, dans la séquence **Do** d'**Open Browser**, ajoutez une activité **Maximize Window** pour obtenir une meilleure visibilité au moment d'exécuter les trois activités suivantes.

- Utilisez une activité **Type Into** pour fournir le nom d'utilisateur. Vérifiez que la case **SimulateType** est cochée dans le volet **Properties**.
- Utilisez une activité **Type Secure Text** pour entrer le mot de passe. Vérifiez que la case **SimulateType** est cochée dans le volet **Properties**.
- Utilisez une activité **Click** pour sélectionner le bouton Log In. Vérifiez que la case **SimulateClick** est cochée dans le volet **Properties**.
- Voici à quoi doit ressembler l'activité Open Browser :

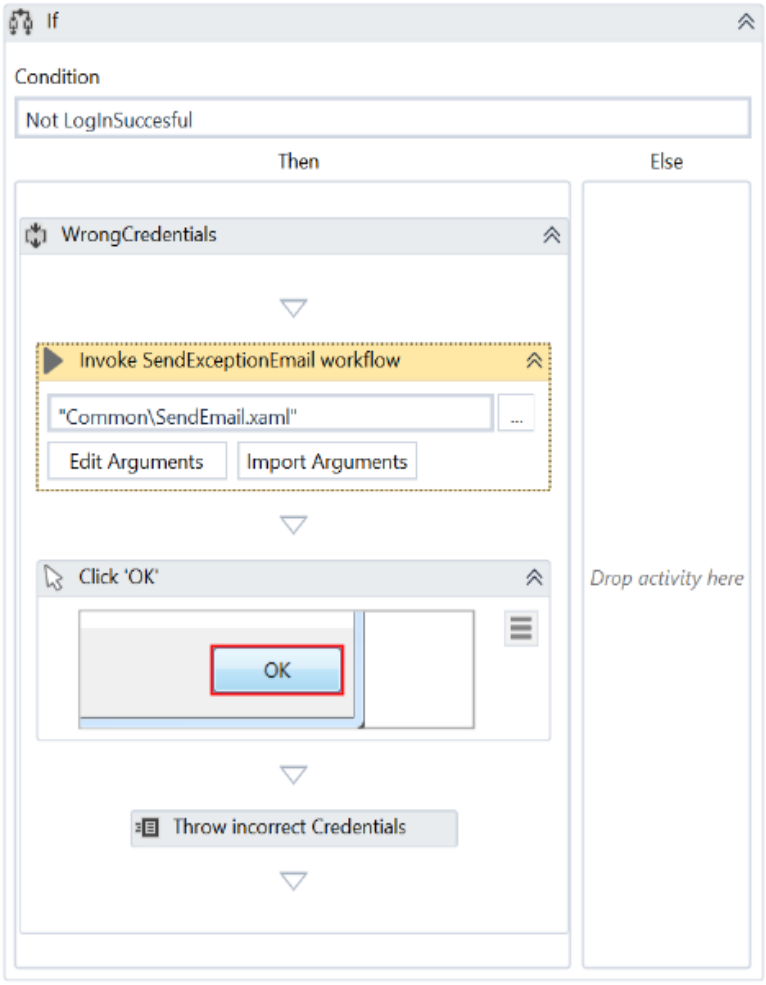


- Il est toujours préférable d'envisager les exceptions pouvant survenir pendant l'exécution de notre processus. Dans ce cas, nous devons vérifier si la connexion a réussi ou non. Cette exigence est également présentée dans le fichier PDD. Essayez de vous connecter avec des identifiants erronés et observez la différence.

- Servez-vous d'une activité **Element Exist** pour vérifier si la connexion a réussi en recherchant un élément n'apparaissant que dans ce cas.

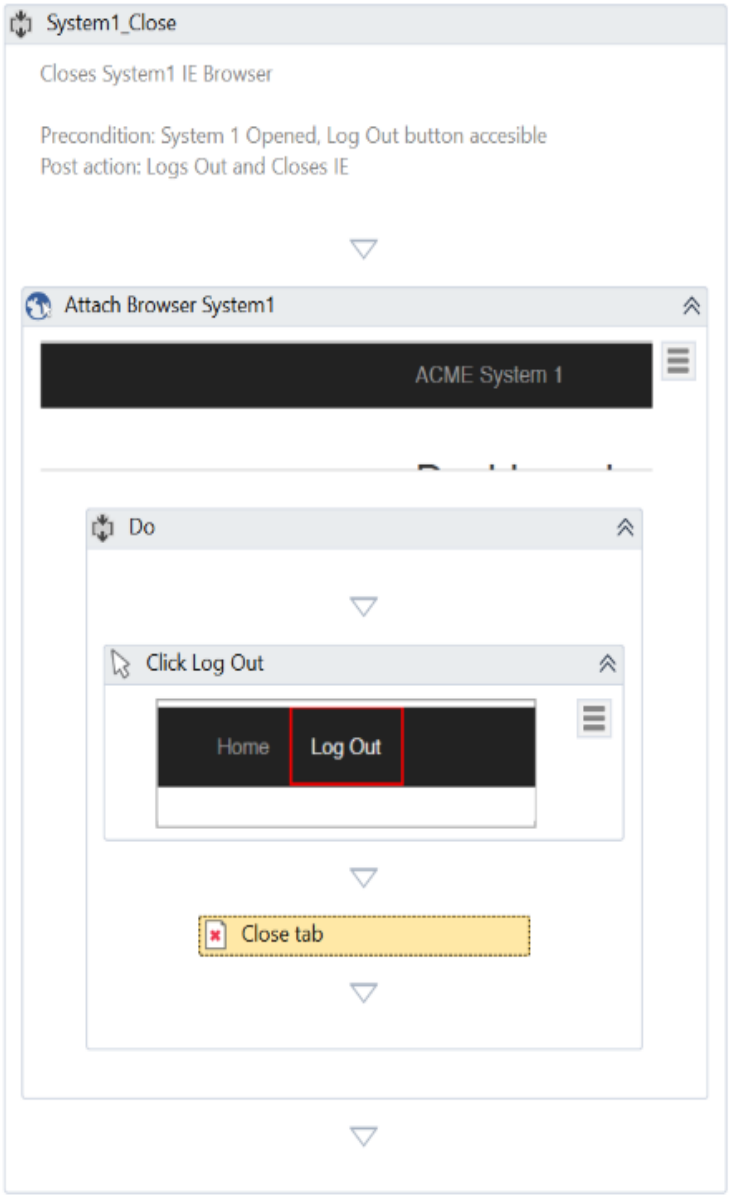


- Utilisez une activité **If** pour vérifier si la tentative de connexion a échoué. Si tel est le cas, effectuez les actions suivantes.
  - Envoyez un email avec l'exception. Il est recommandé de créer un processus réutilisable distinct et de l'invoquer dans la section **Then**.
  - Fermer le message d'erreur au moyen d'une activité **Click**.
  - Levez une exception pour les Incorrect Credentials fournis à System1 afin d'arrêter le processus.
- Voici à quoi doit ressembler l'activité **If** :



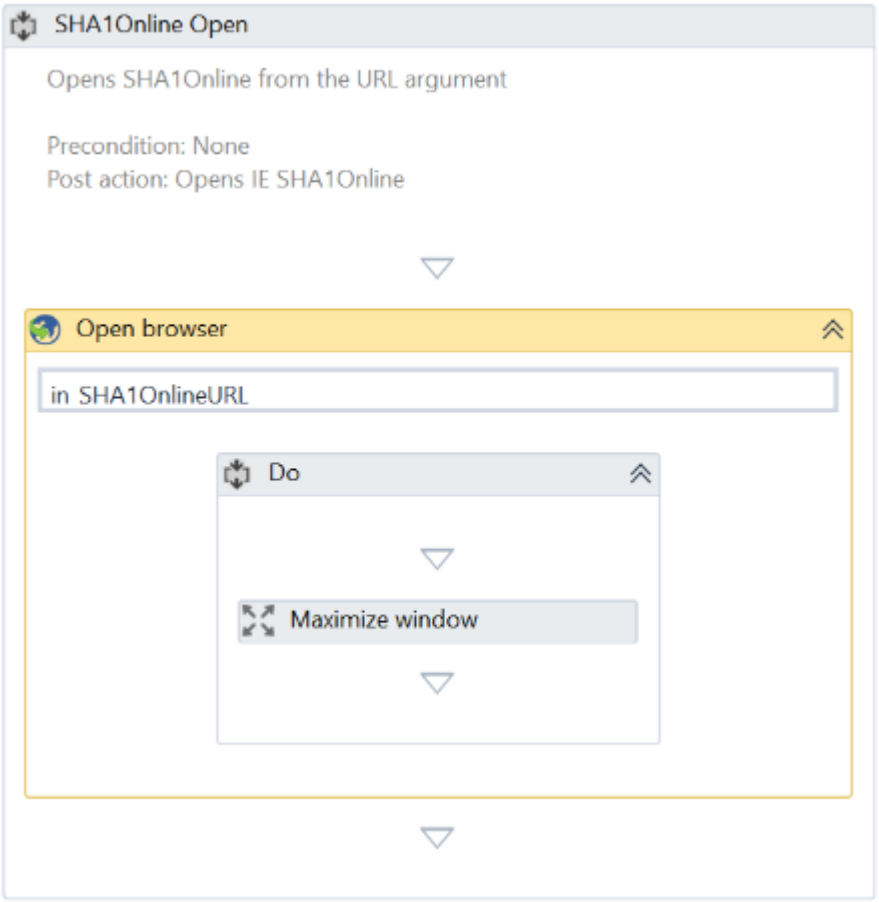
- Testez individuellement le fichier créé en utilisant les valeurs par défaut des arguments.

- Créons ensuite le processus permettant de se déconnecter et de refermer System1. Créez une séquence vide dans le dossier System1.
  - La nouvelle séquence doit débuter par une annotation.
  - Servez-vous de l'activité **Attach Browser** pour vous intégrer à la fenêtre de navigation contenant l'application System1. Nous pouvons extraire ici le premier prérequis du processus : l'application System1 est ouverte.
  - À l'intérieur de l'activité **Attach Browser**, faites un glisser-déposer d'une activité **Click**. Choisissez pour cible le bouton Log Out. Il s'agit du second prérequis du processus : le bouton Log Out est accessible.
  - Utilisez une activité Close Tab pour refermer la fenêtre du navigateur System1.
  - Voici à quoi doit ressembler la séquence :



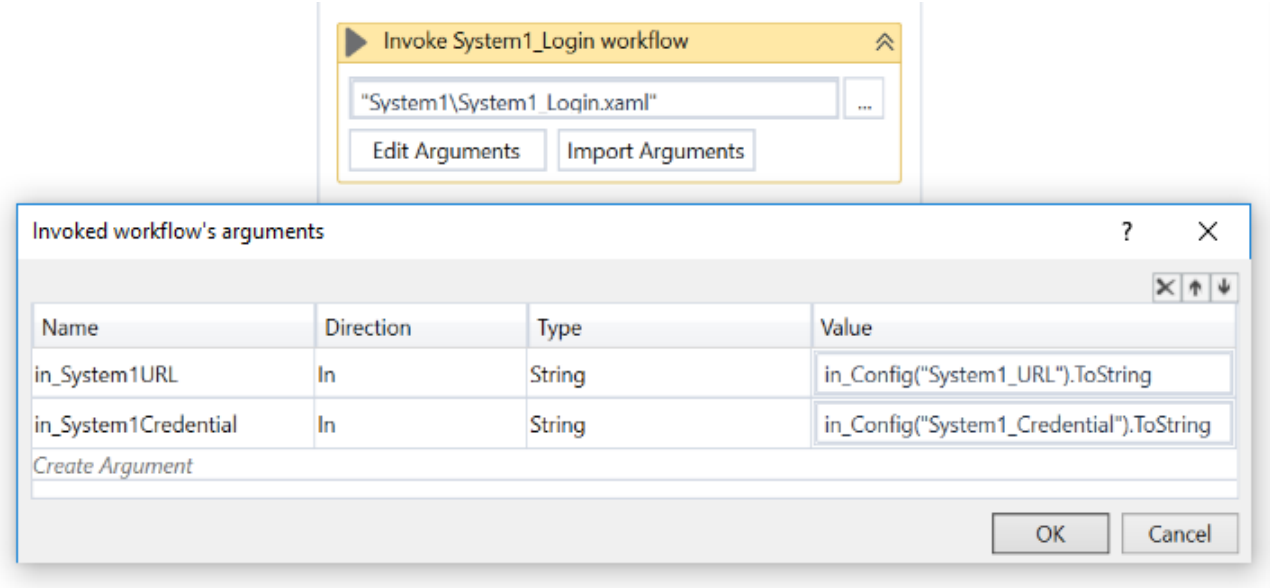
- Nous faisons ensuite de même pour l'autre application intervenant dans le processus SHA1 Online.
- Créez un processus séquentiel vierge dans le dossier SHA1Online pour ouvrir l'application. Cette séquence est plus simple, car l'application n'exige pas d'authentification.
    - Nous commençons bien sûr par une annotation.
    - L'URL de l'application doit être transférée au processus utilisant un argument, pour faciliter la maintenabilité du projet. La valeur URL est stockée dans le fichier de configuration du processus.

- Choisissez une activité **Open Browser** avec l'argument créé pour URL d'entrée. Par défaut, Internet Explorer est utilisé, sauf si la propriété **BrowserType** est éditée.
- Vous pouvez éventuellement employer une activité **Maximize Window** dans la fenêtre de navigation.
    - Voici à quoi doit ressembler la séquence :

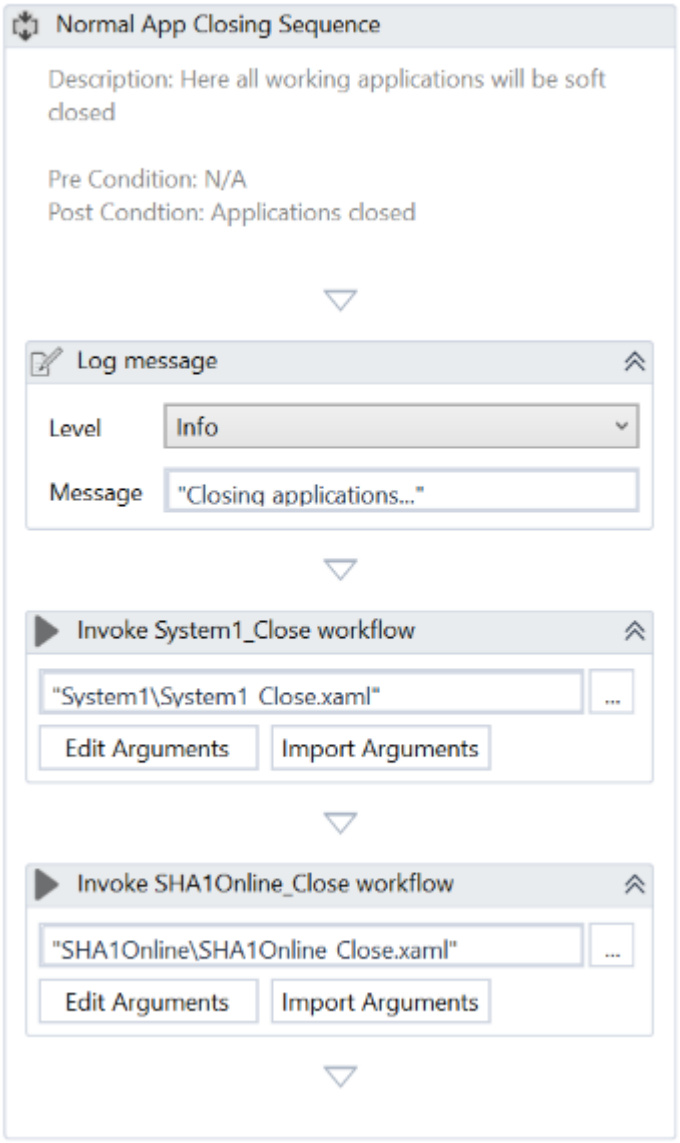


- Créez une séquence vide pour refermer l'application SHA1Online. Celle-ci doit être simple.
- Maintenant que nous disposons des processus permettant d'ouvrir et de fermer les deux applications, nous pouvons apporter les changements voulus dans les parties initialisation et fermeture du cadre.
- Ouvrez le processus **Framework\InitAllApplications**
    - Faites un glisser-déposer du fichier System1_Login. Une activité **Invoke Workflow File** est créée automatiquement.
    - Cliquez sur **Import Arguments** et associez les valeurs à ceux du fichier « Config ».

- Voici à quoi doit ressembler l'activité **Invoke Workflow File** :

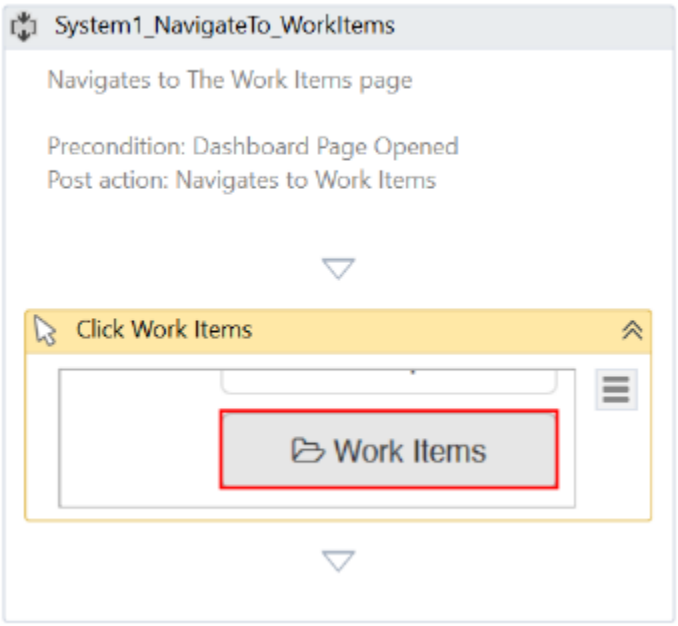


  - Faites un glisser-déposer du processus SHA1Online_Login.
  - Cliquez sur **Import Arguments** et associez l'argument URL à la valeur du fichier « Config ».
- Ouvrez **Framework\CloseAllApplications**.
  - Faites un glisser-déposer des deux processus créés pour refermer les applications System1 et SHA1Online.
  - Voici à quoi doit ressembler la séquence résultante :



- Ouvrez **Framework\KillAllProcesses**.
    - Nos deux applications se trouvent à l'intérieur d'un navigateur web. Si la valeur par défaut de la propriété **BrowserType** des activités **Open Browser** n'a pas été modifiée, Internet Explorer est utilisé.
    - Utilisez une activité **Kill Process** et choisissez « iexplore » pour Process Name.
- Créez un processus séquentiel vierge dans le dossier System1 pour accéder à **Work Items** dans l'application System1. Nous voulons que cette action soit incluse dans un processus distinct, puisque nous allons également accéder aux Work Items d'autres projets.
    - Ajoutez l'annotation voulue.

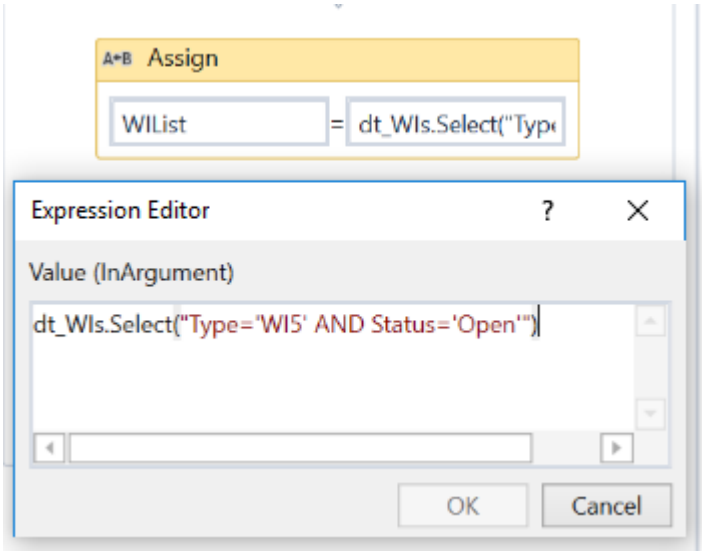
- Appliquez une activité **Click** au bouton Work Items. Pensez à utiliser un sélecteur complet, puisque cette activité n'est pas à l'intérieur d'une portée Attach Browser.
- Voici à quoi doit ressembler le projet :



- Avant de lancer les opérations du processus, il faut ajouter les activités nécessaires à la lecture des données d'entrée dans l'Init State. L'opération est normalement très simple : vous pouvez utiliser soit Read Range, soit Read CSV. Dans notre processus, les données d'entrée étant toutefois stockées sur un site web, il faut en passer par des étapes supplémentaires.
- Rappelez-vous la finalité du processus : récupérer le code hash de chaque élément de type WI5. Pour ce faire, il nous faut d'abord une liste de l'ensemble des éléments WI5.
- Créez un processus séquentiel vierge dans le dossier System1 pour extraire une variable DataTable contenant l'ensemble des Work Items de l'application System1. Nous allons extraire l'ensemble des éléments de travail existants avant de filtrer ceux de type WI5.
  - Servez-vous de l'assistant Data Scraping pour extraire le tableau HTML tout entier. Lorsqu'il vous est demandé si les données peuvent occuper plusieurs pages, répondez Yes, et pointez le bouton de la page suivante.
  - Attribuez à l'option **Maximum number of results** la valeur « 0 », pour que l'ensemble des éléments identifiés puissent être extraits comme sortie.
  - Créez un argument de sortie et attribuez-lui la valeur du tableau de données extrait.

- Si vous avez ajouté une annotation, vous voilà à la fin des manipulations avec ce processus ! Dans le cas contraire, ajoutez l'annotation maintenant.

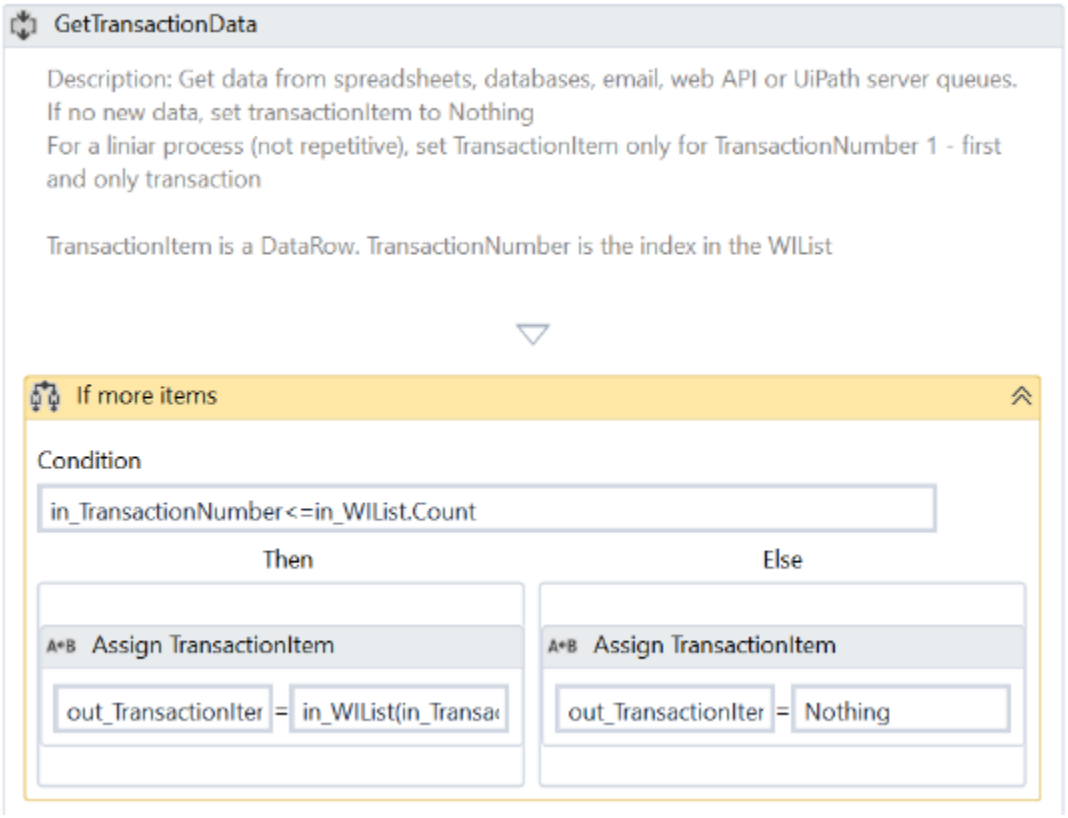
- Ouvrez le processus **Main** et agrandissez l'état **Init** en double-cliquant dessus.
  - Dans la région Entry, repérez où est invoqué le processus KillAllProcesses.
  - Ajoutez une nouvelle séquence après l'activité **Invoke KillAllProcesses** pour lire le tableau de données des opérations d'entrée.
  - À l'intérieur de cette séquence, invoquez quatre des processus précédemment créés, à savoir :
    - System1 Login
    - System1 Navigate to Work Item
    - System1 Extract Work Item Data Table
    - System1 Close
  - Importez et associez-leur des arguments si nécessaire.
  - Dans la liste des éléments de travail, extrayez seulement les éléments nécessités dans le processus courant : ceux de type WI5, avec le statut « Open ».
    - Appliquez la méthode DataTable.Select pour filtrer les éléments ne répondant pas aux critères précédents. Attribuez le résultat à une nouvelle variable.
    - Voici à quoi doit ressembler l'activité **Assign** :



  - Nous allons traiter les différents éléments du tableau un à un. De cette façon, tous les objets DataRow deviennent des Transaction Items. L'indice de l'opération commence à 1, mais la numérotation des tableaux étant à base zéro, l'indice d'un élément du tableau correspond au numéro de l'opération, moins 1.

- Ouvrez le processus **Framework\GetTransactionData**

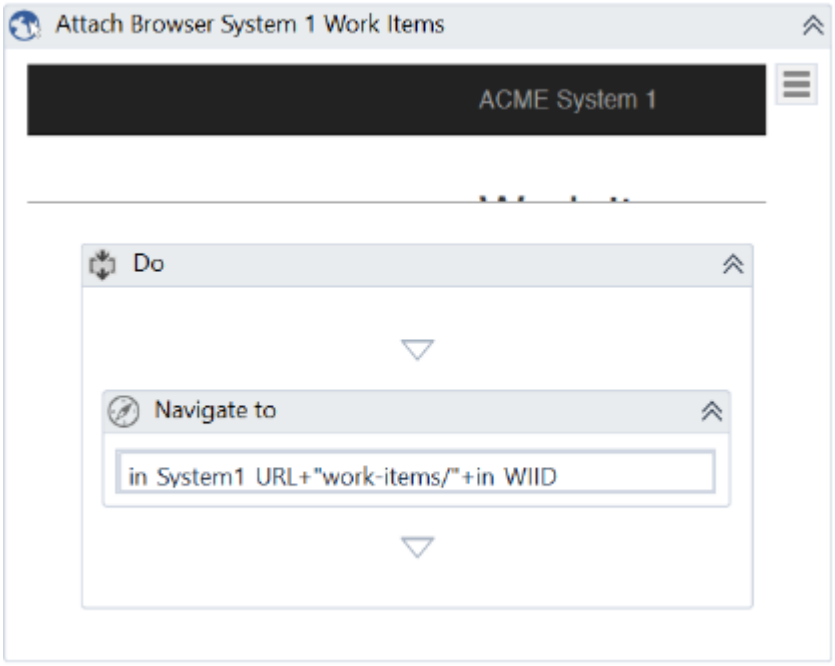
- Actualisez l'annotation du processus courant.
- Ajoutez un argument d'entrée pour le tableau des Work Items.
- Souvenez-vous que notre Transaction Item constitue un objet dans le tableau Work Item. Nous ne pouvons avoir de nouvelle opération que si le numéro de l'opération est inférieur ou égale à l'indice de l'élément du tableau.
- Ajoutez une activité **If** pour vérifier s'il existe une nouvelle opération à traiter.
  - Dans la section **Then**, utilisez une activité **Assign** pour attribuez à la valeur l'argument Transaction Item de sortie en fonction de son indice.
  - Dans la section **Else**, choisissez « Nothing » pour valeur Transaction Item.
- Voici à quoi doit ressembler la séquence Get Transaction Data :



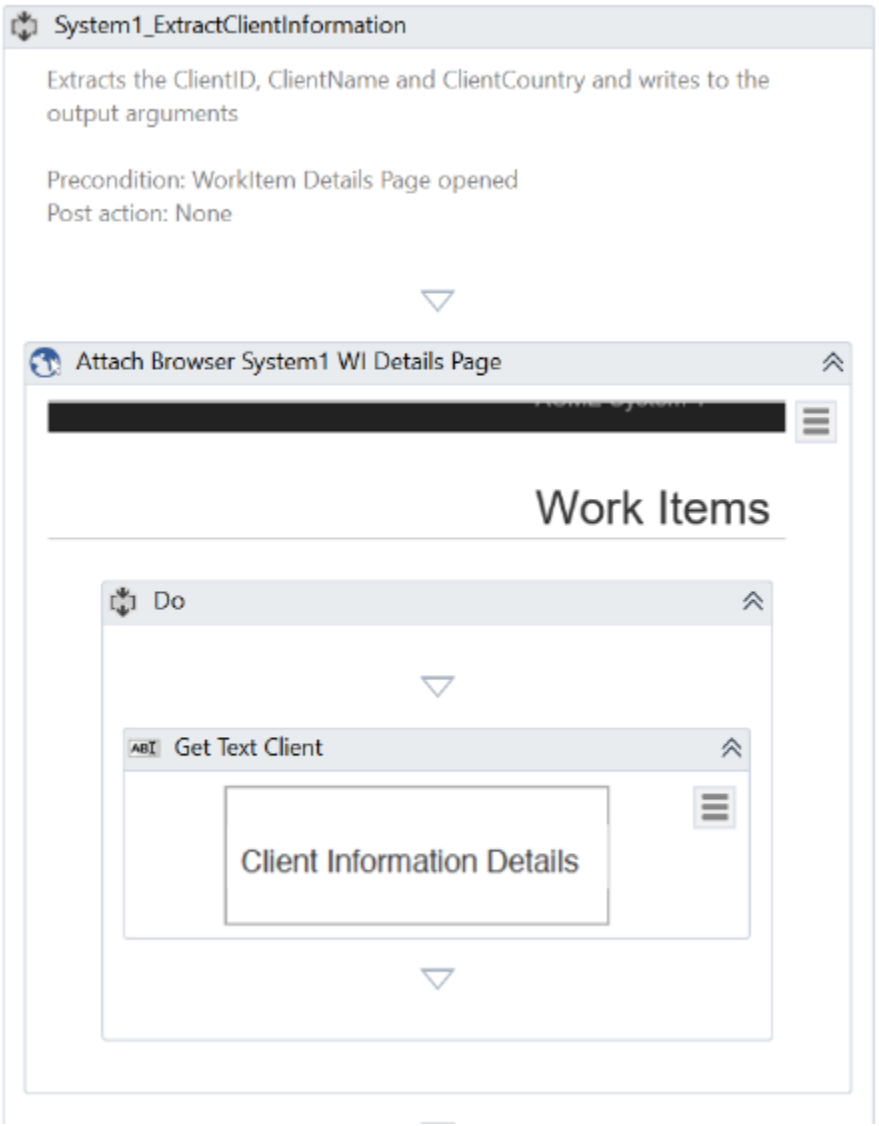
- Par ailleurs, lorsqu'il reste des Transaction Items à traiter, il faut attribuer au Transaction ID la valeur du Work Item ID. Les activités étant déjà en place, nous avons seulement besoin de faire passer la valeur du Transaction ID de l'activité **Assign** à « out_TransactionItem("WIID").ToString ».

- À ce moment, la configuration du cadre est achevée. Nous pouvons tester le processus **Main** en le lançant, puis vérifier l'extraction des données d'entrée. Servez-vous de la fenêtre de connexion Output pour savoir si les données sont exactes.
- Maintenant, entamons le développement des processus traitant les Work Items.

- Créez une séquence vide dans le dossier **System1** pour accéder à la page Work Item Details. Nommez-la **System1_NavigateTo_WIDetails**. Nous pouvons utiliser le Work Item ID pour naviguer directement jusqu'à la page Item Details. Ouvrez un élément de travail et observez le format de l'URL : il se compose de l'URL System1, de la chaîne « /work-item/ », et du Work Item ID.
    - Le Work Item ID et l'URL System constituent l'entrée voulue. Créez deux arguments In : l'un de type Int32, l'autre de type String.
    - Procédez à l'intégration au Dashboard de System1, puis appliquez une activité **Navigate To** pour vous rendre à la page Work Item Details.
    - Voici à quoi doit ressembler la séquence :



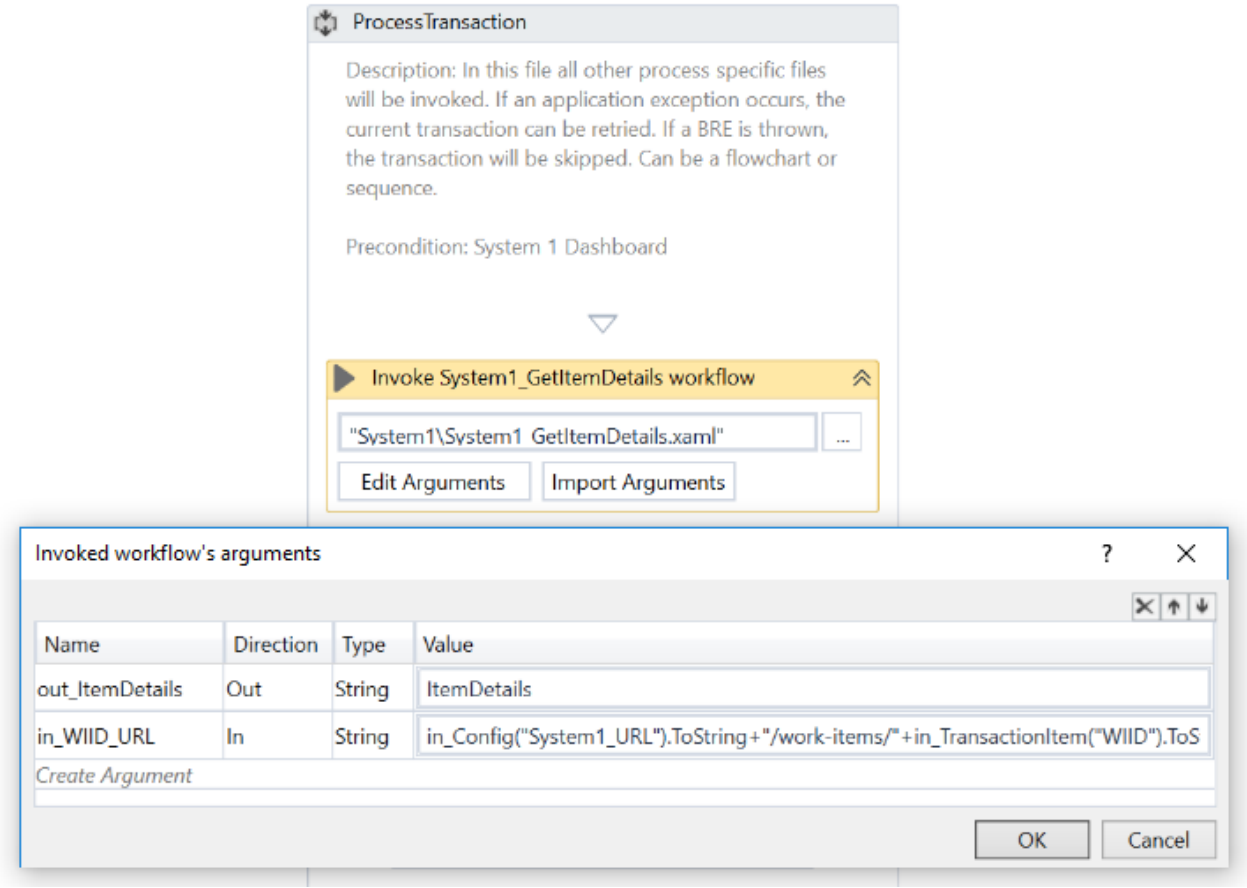
- Créez un processus séquentiel vierge dans le dossier System1 appelé **System1_ExtractClientInformation**. Nous allons nous en servir pour récupérer les détails d'un élément.
    - Il n'existe aucun argument d'entrée dans ce processus. Il existe un seul prérequis : la page Work Item Details doit déjà être ouverte.
    - Il nous faut trois arguments de sortie : Client ID, Client Name et Client Country.
    - Avant de tirer le texte de la page web, nous devons faire en sorte que la page soit chargée, et que tous les éléments soient disponibles. Ajoutez une activité **Attach Browser**. Dans la section **Do**, nous allons employer une activité **Get Text**, sa propriété **WaitForReady** ayant pour valeur **Complete**.
    - Voici à quoi doit ressembler le processus :



- Dans le même processus, il nous faut extraire la valeur de Client ID, Client Name et Client Country.
    - Ajoutez une activité **Assign**. Servez-vous de l'argument de sortie pour Client ID dans le champ **To**.
    - Dans le champ **Value**, nous appliquons les méthodes Substring et Split pour extraire uniquement les informations voulues. L'expression résultante est ClientInformation.Substring(ClientInformation.IndexOf("Client ID: ")+"Client ID: ".Length).Split(Environment.NewLine.ToCharArray)(0)
    - Ajoutez deux autres activités **Assign** pour Client Name et Client Country. L'entrée dans le champ **Value** doit être similaire à celle employée pour l'argument Client ID.

- Après quoi, il est recommandé d'écrire les valeurs extraites dans le volet Output, au moyen de l'activité **Write Line** ; cela nous aidera à déboguer notre processus ultérieurement.

- Ouvrez le processus **Process**
  - Éditez l'annotation.
  - Invoquez le processus **System1_NavigateTo_WIDetails**. Importez et associez les arguments.
  - Voici ce que nous devons obtenir :



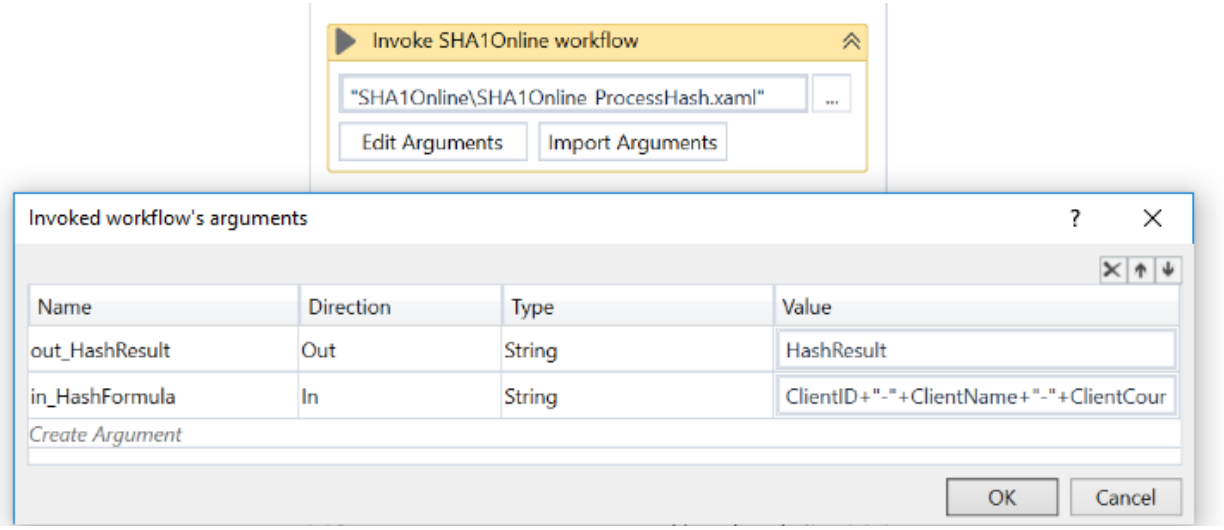
  - Invoquez le processus **System1_ExtractClientInformation**. Associez les trois arguments de sortie aux variables locales.

- Créons à présent un processus permettant de récupérer la valeur hash de l'application SHA1Online.com. Créez un processus séquentiel vierge dans le dossier SHA1Online. Nommez-le SHA1Online_GetHashCode.
  - Il nous faut un argument d'entrée de type String. Sa valeur est celle dont nous devons obtenir le code hash.
  - Nous avons également besoin d'un argument de sortie de type String pour le code hash calculé.

- Ajoutez une activité **Type Into** et configurez-la pour taper l'argument d'entrée dans le champ hash de l'application System1. Dans le volet **Properties**, cochez éventuellement la case du champ **Simulate Type** si vous voulez que cette activité fonctionne correctement en arrière-plan.
- Utilisez une activité **Click** pour sélectionner le bouton **hash**. En option, allez dans le volet **Properties** pour cocher la case du champ **Simulate Type** si vous voulez que cette activité fonctionne correctement en arrière-plan.

- Nous pouvons à présent obtenir le résultat à l'aide d'une activité Get Full Text ; utilisez l'argument de sortie pour stocker le texte de sortie.
- Enfin, il nous faut revenir à la première page de l'application, pour pouvoir appliquer la même séquence au traitement de l'élément suivant. Pour cela, ajoutez une activité **Go Back** afin de naviguer jusqu'à la page voulue.

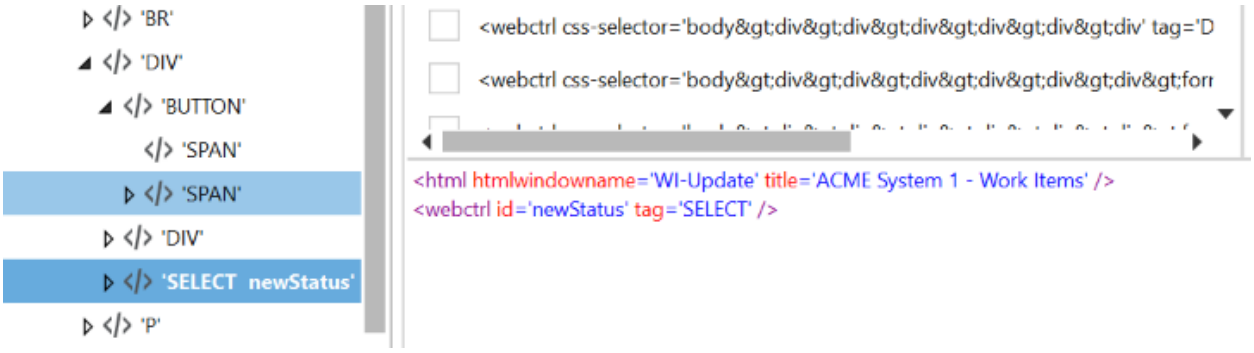
- Rappelez-vous que nous devons calculer le code hash de [ClientID]-[ClientName]-[ClientCountry], ce qui suppose de composer cette chaîne. Employons la valeur de sortie du processus **System1_ExtractClientInformation** pour argument d'entrée dans le processus **SHA1Online_GetHashCode**.
- Revenons au processus **Process**.
  - Ajoutez une activité **Invoke Workflow** et sélectionnez SHA1Online_GetHashCode.
  - Importez les arguments. Servez-vous de la formule du hash pour argument d'entrée. Créez une nouvelle variable pour stocker le résultat du hash.
  - Voici à quoi doit ressembler le processus :



- Créez un processus séquentiel vierge dans le dossier System1 pour actualiser les Work Items avec le code hash calculé. Mieux vaut créer un processus générique pouvant être réutilisé dans de futurs projets. Pour actualiser les éléments de travail,

il suffit d'ajouter un commentaire, de sélectionner le nouveau statut, puis de soumettre les modifications.

- Débutez la nouvelle séquence en ajoutant une annotation. La page Work Item Details devant être ouverte pour prérequis, nous pouvons actualiser le Work Item.
- Ajoutez deux arguments String d'entrée, l'un pour le commentaire, l'autre pour le nouveau statut.
- Ajoutez une activité **Click** et choisissez le bouton **Update Work Item** pour cible. Dans le volet **Properties**, cochez la case dans le champ Simulate Click.
- Appliquez une activité **Type Into** pour renseigner le champ de commentaire sur la page **Update Work Item**. Cochez également la case **Simulate Type** dans le volet **Properties**.
- Vous devez à présent mettre à jour l'état de cet élément de travail. Pour ce faire, faites un glisser-déposer d'une activité **Select Item**. Cliquez sur **Indicate on Screen**, puis sélectionnez la liste déroulante du champ **New Status**. Vous recevez probablement un message d'erreur indiquant que cette commande ne prend pas en charge l'élément sélectionné. Cela s'explique par le fait que d'autres éléments de l'IU se trouvent au-dessus de l'entrée Select.
  - Ouvrez **UiExplorer** et cliquez sur Select Target Element. L'élément IU retourné se présente sous forme de bouton ou de *span*, selon l'endroit où vous avez cliqué.
  - Cliquez sur le champ **New Status**. Sélectionnez l'élément se trouvant au-dessous du bouton ou de la *span*, dans le volet Visual Tree, et servez-vous du sélecteur généré pour l'activité Select Item.
  - Voici ce que nous devons obtenir dans UiExplorer :



  - Enfin, attribuez l'argument in_Status à la propriété Item.
- Utilisez une activité **Click** sur le bouton Update Work Item. Vérifiez que l'option **Simulate Click** est cochée.

  - Un message de confirmation s'affiche : il vous faut employer une autre activité **Click** pour sélectionner le bouton OK. L'option **Simulate Click** doit elle aussi être cochée.
  - Vous pouvez à présent fermer la fenêtre Update Work Item en cliquant sur le bouton Close situé en haut à droite de l'écran.

- Revenez au processus Process et invoquez le processus nouvellement créé. Veillez à employer les bonnes variables pour valeurs des arguments d'entrée.
- Enfin, quittez l'application dans son état initial pour pouvoir traiter l'élément suivant. Pour cela, invoquez le processus créé de manière à naviguer jusqu'à la page Dashboard.
- Nous en avons terminé avec la mise en œuvre du processus. Il nous faut ensuite tester le processus entier. Vous devez déjà avoir testé chaque processus individuel, juste après le développement, en utilisant les valeurs par défaut des arguments.
  - Lancez le processus Main plusieurs fois et vérifiez qu'il s'exécute correctement à chaque fois. Dans le cas contraire, résolvez les problèmes et relancez-le.
  - Utilisez l'option **Reset Test Data** du menu **User Options**, pour générer un nouvel ensemble de données de test.
  - Testez la fonctionnalité Réessayer. Attribuez la valeur « 1 » au paramètre MaxRetryNumber de Config.xlsx to 1 et interférez avec le robot en cliquant sur un lien différent, par exemple sur le lien Home, juste après que le robot ouvre le Work Item courant. De cette façon, le bouton Update Work Item n'est pas disponible, et une exception système est levée puisque l'élément IU n'est pas trouvé. Le processus se réinitialise et l'exécution reprend à partir de l'élément de travail à l'origine de l'échec. Si vous interférez à nouveau, le processus va s'arrêter, dans la mesure où le nombre maximal de nouvelles tentatives est atteint.

Notes concernant la mise en œuvre du processus

Nous avons commencé par récupérer la liste de l'ensemble des articles à traiter en faisant appel à l'assistant Data Scraping. Posez-vous les questions suivantes : Que se passe-t-il si vous êtes en présence d'une série d'opérations nombreuses et que l'activité Extract Data dans l'état Init échoue à cause d'une temporisation du navigateur ? Comment améliorer la conception pour perfectionner le traitement des erreurs ?

- Nous avons traité un article à la fois. Tous les éléments étant indépendants l'un de l'autre, nous pouvons également les traiter en parallèle, en recourant à plusieurs robots. Dans le prochain exercice, nous verrons comment la fonctionnalité Orchestrator Queues sert à mettre en œuvre un processus dans lequel les éléments de travail sont répartis entre les robots et traités seulement une fois, en parallèle.