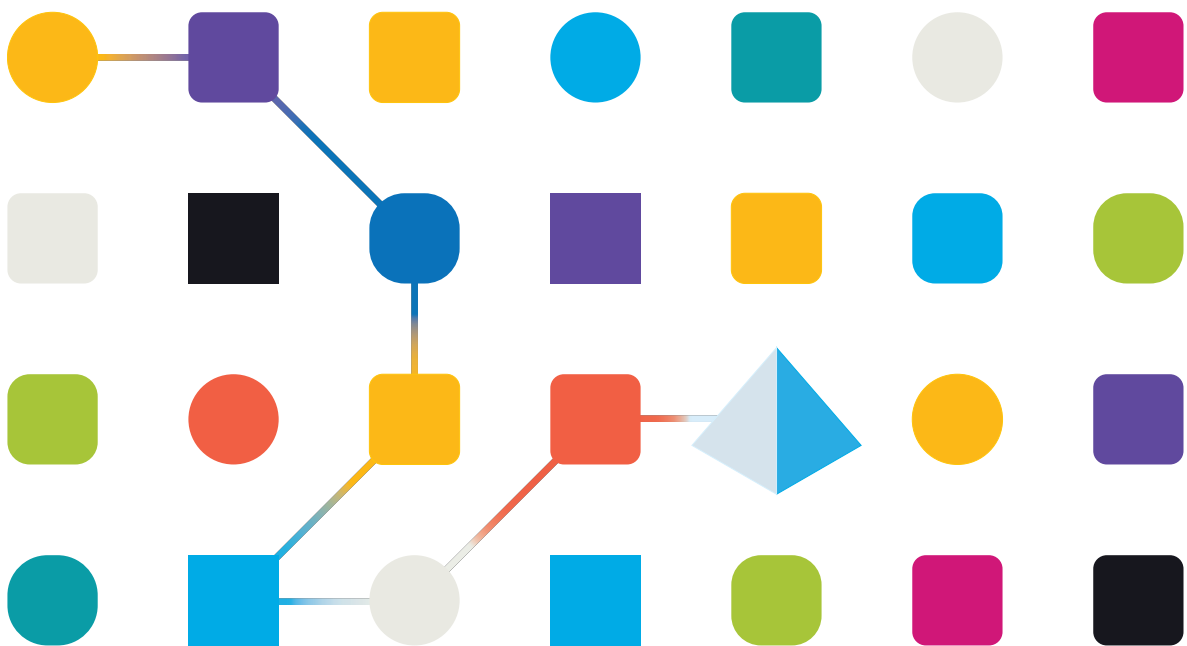


blueprism[®]

Wireframer 3.0

User Guide

Document Revision: 1.0



Trademarks and copyrights

The information contained in this guide is the proprietary and confidential information of Blue Prism Cloud Limited and/or its affiliates and should not be disclosed to a third-party without the written consent of an authorized Blue Prism representative. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying without the written permission of Blue Prism Cloud Limited or its affiliates.

© 2021 Blue Prism Cloud Limited. "Blue Prism", the "Blue Prism" logo and Prism device are either trademarks or registered trademarks of Blue Prism Limited and its affiliates. All Rights Reserved.

All other trademarks are hereby acknowledged and are used to the benefit of their respective owners.

Blue Prism Cloud Limited and its affiliates are not responsible for the content of external websites referenced by this help system.

Blue Prism Cloud Limited, 2 Cinnamon Park, Crab Lane, Warrington, WA2 0XP, United Kingdom.

Registered in England: Reg. No. 8534024. Tel: +44 870 879 3000. Web: www.blueprism.com

Contents

Trademarks and copyrights	2
Wireframer	4
Releases	4
Benefits	4
Basics	5
Create a new wireframe	6
Saving	11
Editing a draft wireframe	11
Deploying a wireframe	11
Complete a wireframe	13
Spying	14
Attach page	15
Init page	15
Activate page	16
Wireframe action page	18

Wireframer

One of the biggest recommendations that can be given to a delivery/development team is to never underestimate the importance of the 'design phase'. It is important to ensure that you have a design phase, especially if you want to approach automation development and delivery holistically.

Blue Prism Cloud follow a 'design-centric' development lifecycle with the separation of the development initiative being split into design and development. The design aspect is undertaken by an architect or senior lead in the project, focusing on delivering a development wireframe to the core development team. The purpose of the design stage is the understand what objects, components, processes, queues, environment variables and files need to be created, how they will be managed and maintained and understand what they purpose is, not just in the short term for delivering this process automation, but also in the longer term of how you want to build up the ecosystem of automation artifacts. These objects, components, and processes will then be wireframed together to give a skeleton to the business process automation, without mapping or integrating into any business applications. They may include some basic logic, but it is common that there is no advanced logic prepared. It is recommended that the process wireframe is inherited from a process template. This ensures that the core process design is the same across every process automation, thus readable and maintainable by all developers.

Blue Prism Cloud provide various toolsets and prepared workflows for achieving fluid, rapid design implementations, subsequently providing a quicker and more agile development phrase. The main tool to assist the Automation Developer in this task is Wireframer within Hub.

Releases

This user guide covers Wireframer, a plugin within Hub which is part of the Blue Prism Cloud platform. This specific guide is for the following releases of the Blue Prism Cloud product:

- Release 3.0 of Wireframer is the standalone version released in February 2020; and
- Release 4.0 of Wireframer is the on-premise variant released in August 2020.

Wireframer is changing location from release 4.0. It will now be found under the Automation Lifecycle Management (ALM) section of Hub, rather than the Design Studio section. This guide outlines functionality and usage of the Wireframer plugin. By utilizing Wireframer, automation developers can create a skeleton of the automation which will adhere to Blue Prism Cloud's best practice. It is assumed as part of this guide that the user is familiar with Blue Prism Cloud digital workforce and has experience with components such as Hub and Blue Prism.

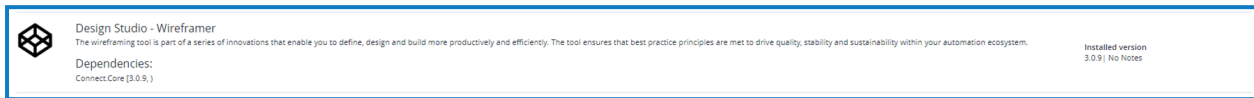
Benefits

The Wireframer allows you to efficiently define business objects that can be used as part of an automation process. The benefits of designing using this methodology is that it allows the Automation Designer to rapidly deploy business objects and actions that will form the structure of the business process being automated. The simplicity in the plugin allows the definition of these business objects and actions, along with best practice techniques, to ensure that enterprise grade automations are always built. Wireframer is designed to work with Blue Prism Cloud's approach to types of object structure, which are:

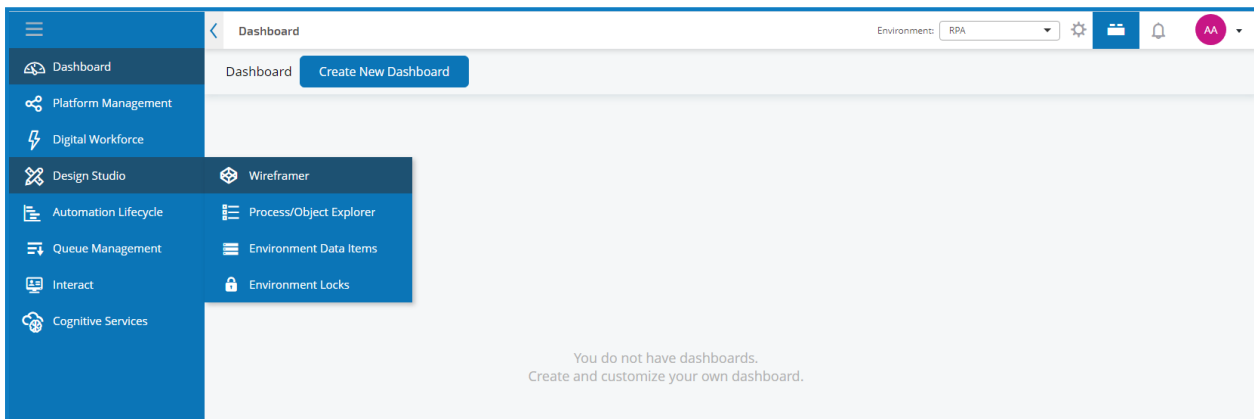
- OPP/OPS – Objects per Page/Object per Screen;
- MOPP/MOPS – Multiple Objects per Page/Multiple Objects per Screen;
- OPA – Object per Application.

Basics

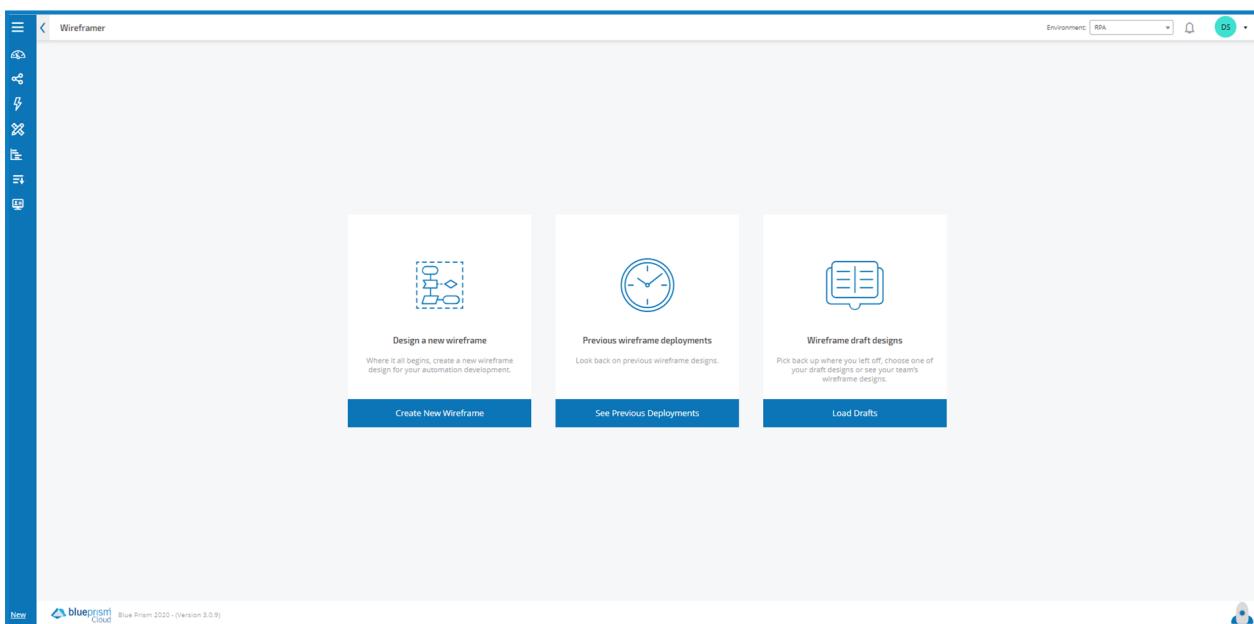
As mentioned, Wireframer is a plugin that is installed into Hub. Like all plugins Wireframer will need to be installed by your Hub administrator before it can be made available to your user community. The Wireframer plugin is found under the Design Studio section of plugins in the Plugin Repository.



After installation Wireframer is launched from the left-hand navigation menu bar.



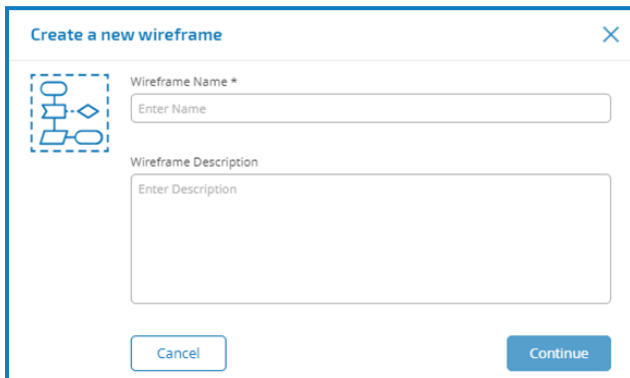
After launching Wireframer the following appears.



This menu allows the user to create a new wireframe, look back on previous deployed wireframes, or to continue where you left of if you were only partially through a wireframe design.

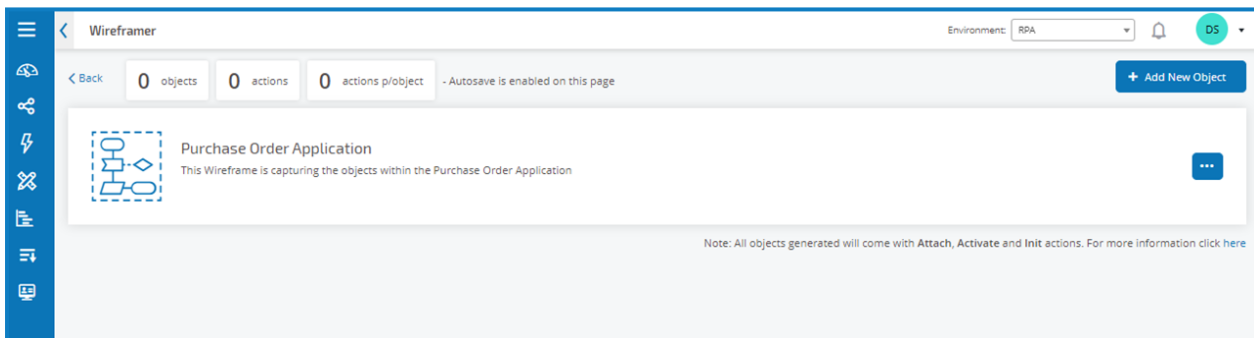
Create a new wireframe

By selecting the Create New Wireframe menu option on the Wireframer landing page, the user is prompted to enter a name and a description for the new wireframe.



The dialog box titled "Create a new wireframe" has a close button (X) in the top right corner. On the left is a wireframe icon. The main area contains two input fields: "Wireframe Name *" with a placeholder "Enter Name" and "Wireframe Description" with a placeholder "Enter Description". At the bottom are "Cancel" and "Continue" buttons.

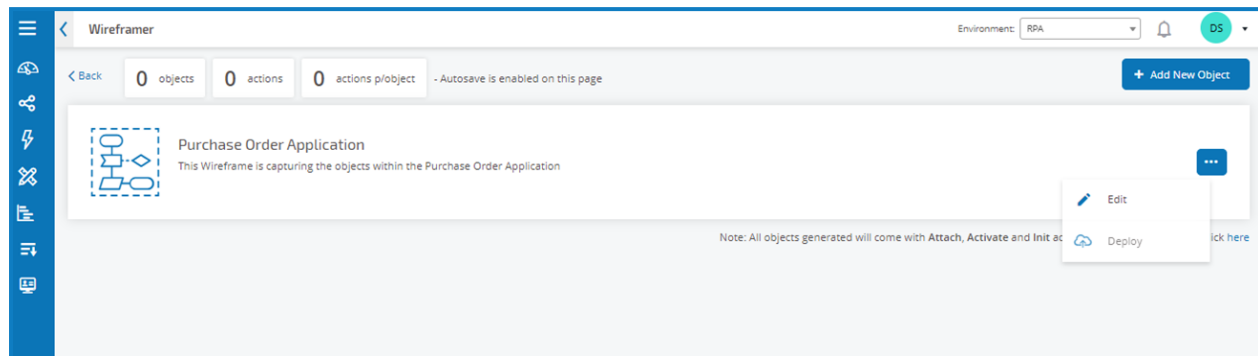
Enter a wireframe name and an optional description. The wireframe name is used to enable easy identification of the wireframe using the search and filtering capabilities of Hub. This enables the organization to identify wireframes when saved as draft or deployed. Though the description is optional it is best practice that this is supplied to ensure that if others are working on the same project they understand what the wireframe you have created relates to. The description enables a user to view the details of the wireframe without needing to open the wireframe and view each individual object, so a good description is really essential. Once the form has been completed, press Continue to launch Wireframer.



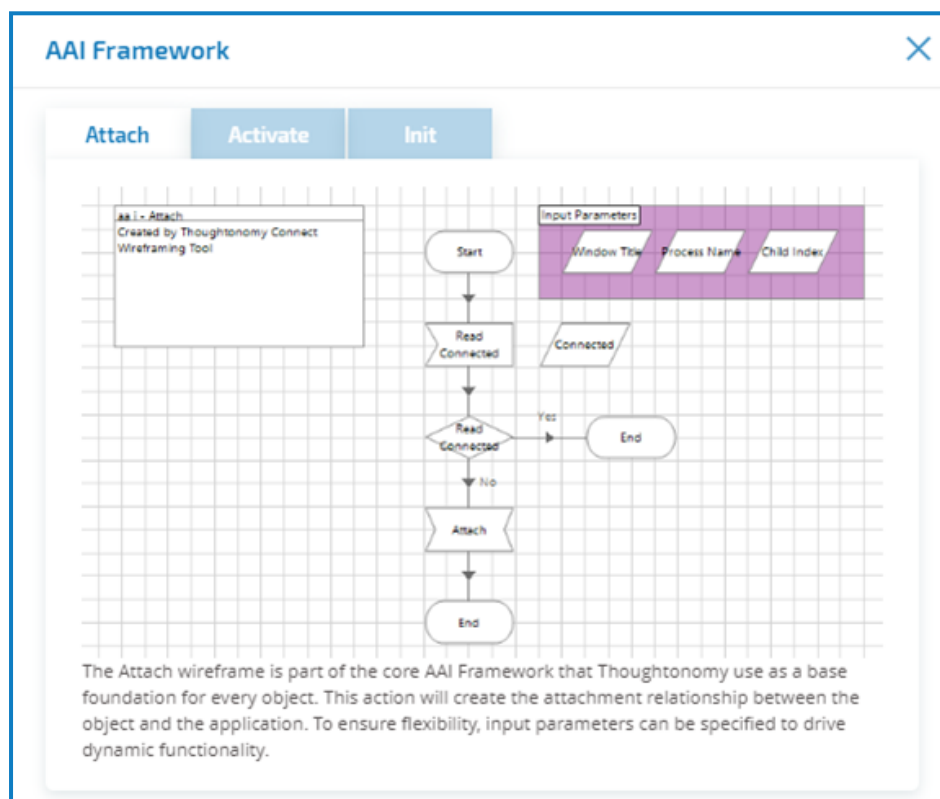
The wireframe created, enables you to define Objects and Actions, the same principle and manual techniques that are used in every automation build. At the top level, Objects can be added.

After you have added Objects you can then add Actions to those Objects, defining the capabilities these objects can offer to an all-consuming automation component or process. A tally is kept of the Objects and Actions that the wireframe consists of across the top of the Wireframer in addition to the average actions included within each object.

The options button to the right allows the user to Edit the wireframe name and description provided earlier; or once completed Deploy the wireframe.



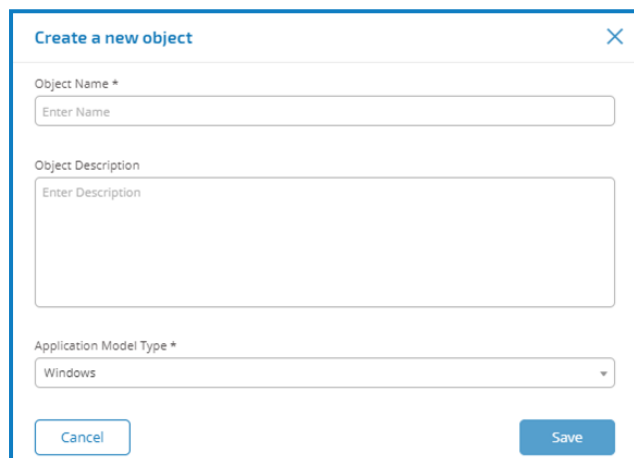
Automatically for any new Object added, three Actions are added to the Object. By clicking on the “here” link you can view the Actions.



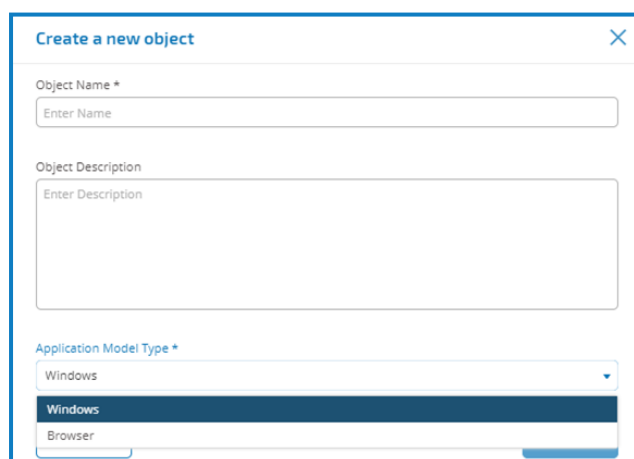
To ensure that best practice is adhered all objects that are defined within the Wireframer, adhere to the AAI Framework concept that Blue Prism Cloud recommends. In summary, this creates three Actions by default:

- **Attach** – This Action creates the attachment relationship between the object and the application;
- **Activate** – This Action will bring the application window to the foreground ensuring that global mouse clicks or keyboard events are directed at the focused application;
- **Init** – This Action ties together the Attach and Activate Actions so they can be called at the top level of any subsequent Action.

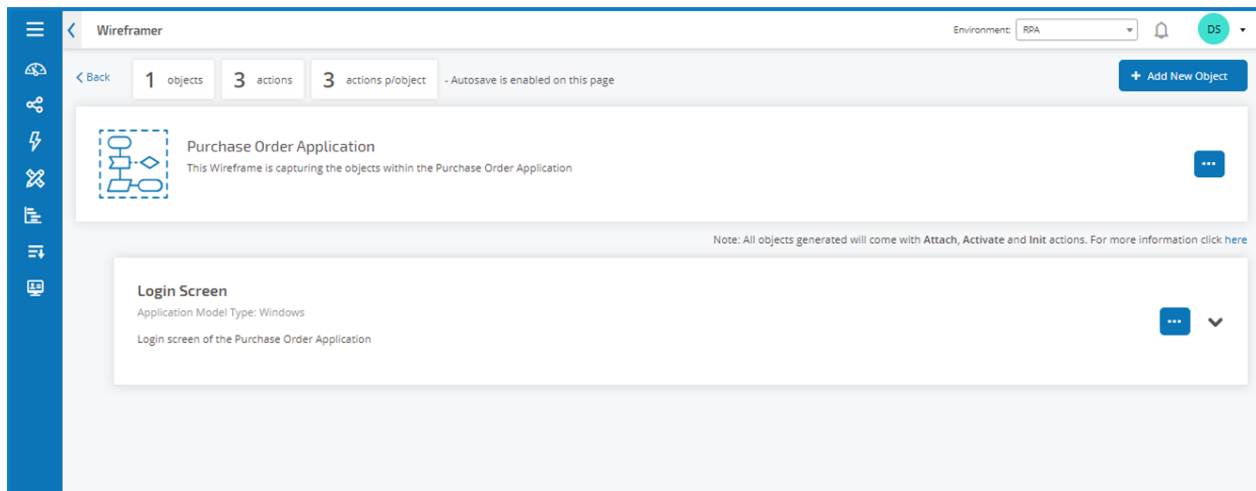
To build your wireframe process, click on the Add New Object button. This will display the Create a new object form. Again, you need to enter a name and a description of the Object.



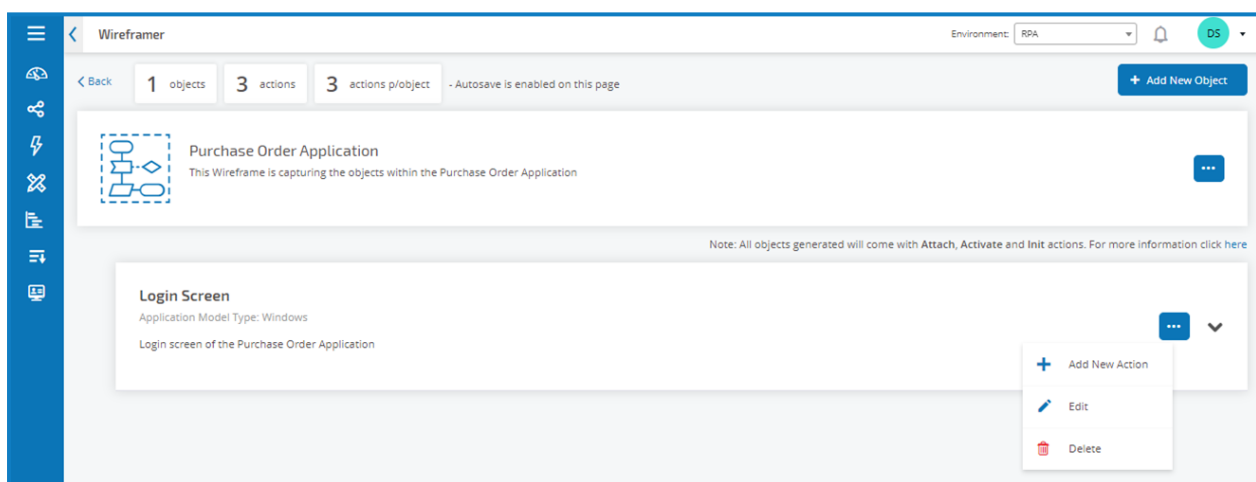
If you consider the design of an automation processes you would normally start with 'launching' the application. When using Wireframer we would build this step separately to the wireframe. The use of Wireframer is to map the screens within the application that we are interacting with using the different structural types of design. In addition to setting the name and description, you need to select from the dropdown whether the Application Model Type is a Windows or Browser based application that you want to interact with. This is used to construct a basic Application Model within the object which can be used.



After selecting your desired Application Model Type, press Save to create the Object. The Wireframer is updated with the information provided and the tally is updated to reflect the new addition.



By clicking on the options button within the Object panel, you can Edit the name and description of the Object; Add New Action to add additional Actions; and Delete if you wish to remove the Object and all associated Actions.



We will now select the Add New Action menu option to create an additional Action alongside the existing default Actions. Like the early forms, enter a name for the Action and a suitable description, then using the dropdown under the Wireframe Type select the type the action will perform.

Create a new action

Action Name *

Enter Name

Action Description

Enter Description

Wireframe Type *

Empty

Preview

☐ Publish this action

Cancel

Save

There are six Action types that can be chosen:

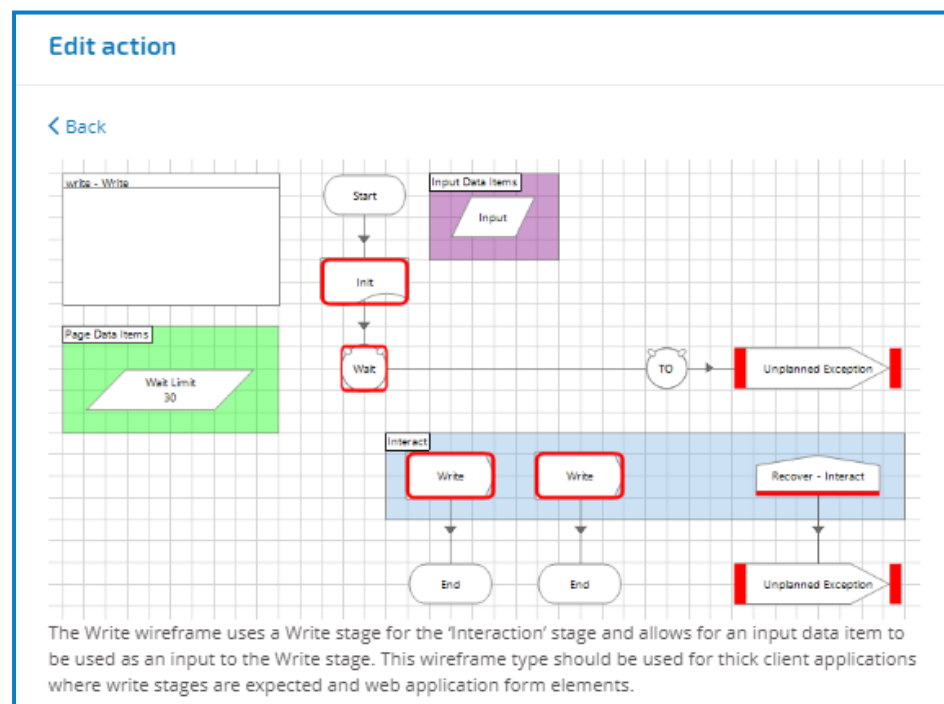
- **Empty** – an empty Action that has no inbuilt functionality added to it, it is simply a Start stage linked to an End stage;
- **Init-only** – an Init Action which is a Start and End stage with a Page Reference stage in between that references the Init action. This will ensure that the application is attached and the window is focused on the screen;
- **Navigate** – deployment of a Navigate stage so you can associate it with a button push or click;
- **Read** – a Read stage that enables the reading of information from the application;
- **Select** – is a Navigate stage with an Input, that enables you to access drop-downs or check boxes where you can pass a true or false flag to 'tick' the required box for example;
- **Write** – a Write stage enables the population of a field in an application form.

Finally, after the form has been configured for the Action you require, choose to 'Publish this action'. If published was not selected when deploying, the created Actions in the Object will not be available to the Object when you come to use it in your Processes, so make sure that the Actions are selected to Publish before deploying.

The AAI Actions are not published. You would only need to call them within the Object themselves, so a local interaction and therefore there is no need for them to be published.

- The Init Action calls the Attach and Activate Actions (Internally within the Object);
- Your Actions calls the Init Action (Internally within the Object);
- Your Actions are published meaning they are exposed for other Processes or Objects to use those actions.

You can view the Action you have added to the Object by selecting the Preview button. This will display the selected Action type as an example the Write action.



To close the Preview, press the back button on the display. Once you have configured the Action, press Save to store the changes. The same as the previous option menus the option button allows you to Edit and Delete the Action as required. You can then repeat the steps for adding other Actions as required to

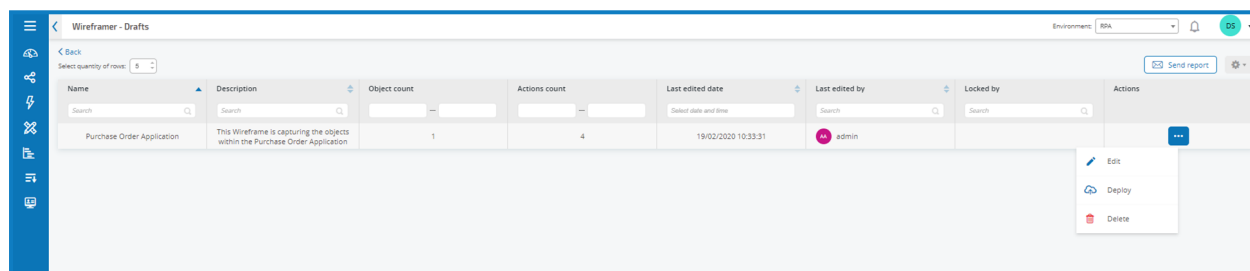
your Object, or add additional Objects and Actions as required to map out a particular application or section of an application as required.

Saving

Any changes you make to the wireframe as you build out the Objects and Actions are automatically saved. If you close the window the wireframe is saved as a draft and you can resume from where you left off.

Editing a draft wireframe

To edit a wireframe that has been saved as a draft, from the main window select the Load Drafts from the main menu as shown below. Within the Drafts display, using the menu button, you can continue to Edit the wireframe, Deploy if the wireframe is complete and ready to be built, or Delete if the wireframe is no longer required.



Deploying a wireframe

By selecting the Deploy you are presented with the following form.

Deploy

Once deployed the wireframe cannot be modified.

Environment
RPA

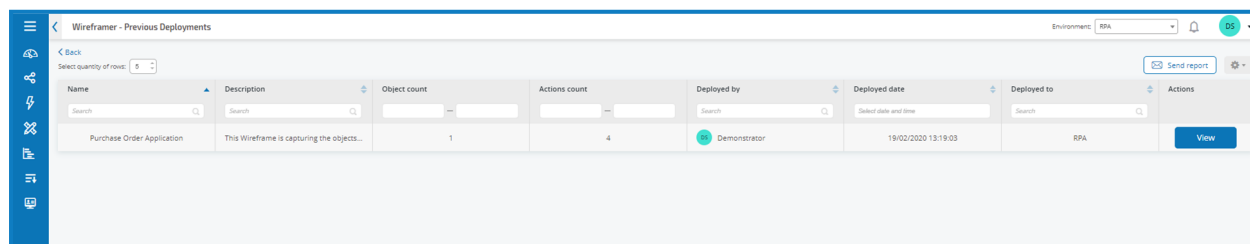
This will create 1 object(s)

Cancel Deploy

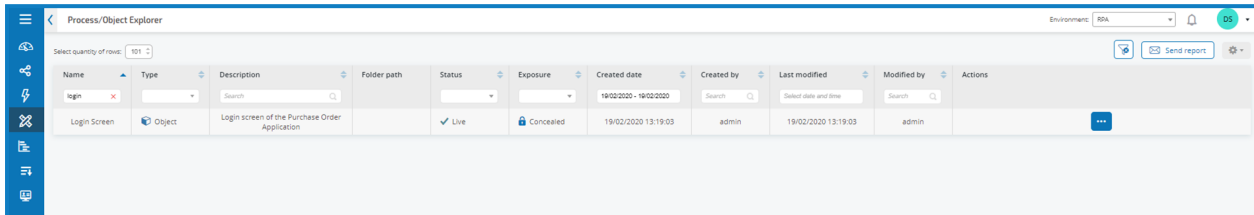
This presents you with a summary of what will be deployed and allows you to select which environment you wish to deploy the Object(s) into.

You can also choose to Deploy from the Create Wireframe display if you have completed all the Object(s) and Action(s) in one session without needing to use the draft functionality.

By pressing Deploy the wireframe is created and Object(s) deployed to that environment. Once an Object is deployed through the Wireframer it cannot be edited or deleted within Hub. The deployment is added to Blue Prism and is only available there for further additional work or if required deletion. Once deployed the Object can be viewed in the wireframe deployment screen, accessible by selecting **See Previous Deployments** on the main wireframe menu.



By selecting View you can see the details of the wireframe that you built earlier. To view the associated XML code created during the creation of each Object within this wireframe, you will need to ensure you have access to the Process/Object Explorer plugin within Hub and if necessary switch to the environment that the Objects were deployed into. Search for the Object name that we created, not the wireframe name, so in our example Login Screen.



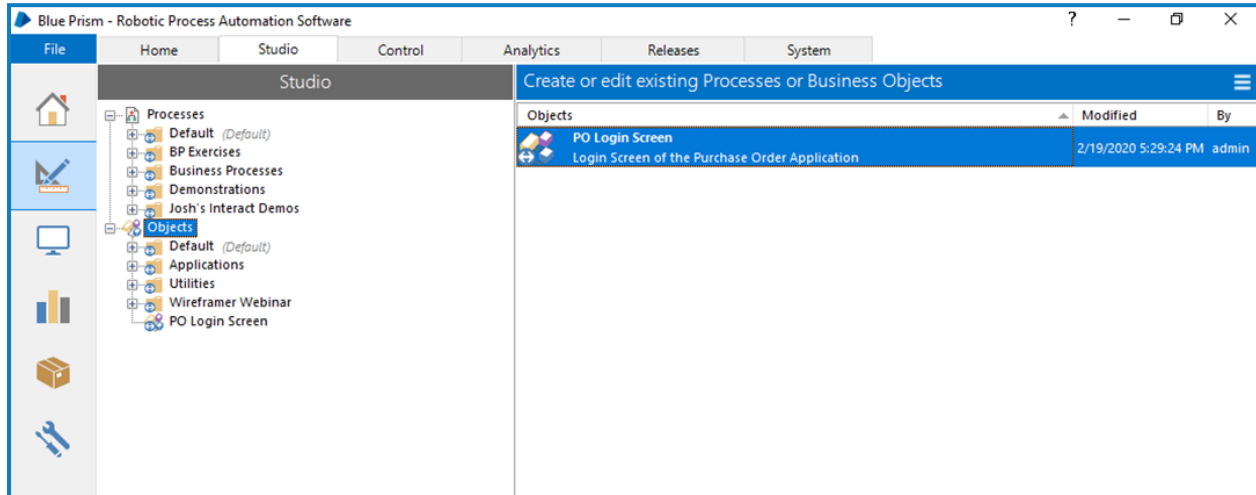
Name	Type	Description	Folder path	Status	Exposure	Created date	Created by	Last modified	Modified by	Actions
Login Screen	Object	Login screen of the Purchase Order Application		✓ Live	Concealed	19/02/2020 13:19:03	admin	19/02/2020 13:19:03	admin	...

By selecting View XML from the associated option button, the XML code will appear in a new browser tab.

```
<process name="Login Screen" version="1.0" bpversion="5.0.23.0" narrative="Login screen of the Purchase Order Application" type="object" runmode="Exclusive" preferredid="3dc91207-0a71-4e3a-aa91-4a893886eca6">
  <appdef>
    <element name="Application">
      <id>136d8b85-8217-48a5-830f-c386fb04cd5</id>
      <group name="Main Screen">
        <id>e78644c3-69f5-491f-a445-18a2dcead047</id>
      </group>
      <type>Application</type>
      <basetype>Application</basetype>
      <datatype>unknown</datatype>
      <diagnose>false</diagnose>
    </element>
    <apptypeinfo>
      <id>Win32Attach</id>
      <parameters>
        <parameter>
          <name>WindowTitle</name>
          <value />
        </parameter>
        <parameter>
          <name>ProcessName</name>
          <value />
        </parameter>
        <parameter>
          <name>Path</name>
          <value />
        </parameter>
        <parameter>
          <name>CommandLineParams</name>
          <value />
        </parameter>
        <parameter>
          <name>ProcessMode</name>
          <value>Internal</value>
        </parameter>
        <parameter>
          <name>NonInvasive</name>
          <value>true</value>
        </parameter>
      </parameters>
    </apptypeinfo>
  </appdef>
  <view>
    <camerax>0</camerax>
    <cameray>0</cameray>
    <zoom version="2">1.25</zoom>
  </view>
  <preconditions />
  <endpoint narrative="" />
  <subsheet subsheetid="1743028d-4082-4a85-930f-577d65765f0e" type="CleanUp" published="True">
    <name>Clean Up</name>
    <view>
      <camerax>0</camerax>
      <cameray>0</cameray>
      <zoom version="2">1.25</zoom>
    </view>
  </subsheet>
</process>
```

Complete a wireframe

The wireframe is a skeleton of an object and does not contain all the necessary information to perform the automation. If we switch to Blue Prism, we can see that the deployed wireframe Object, in our example here PO Login screen has been added.

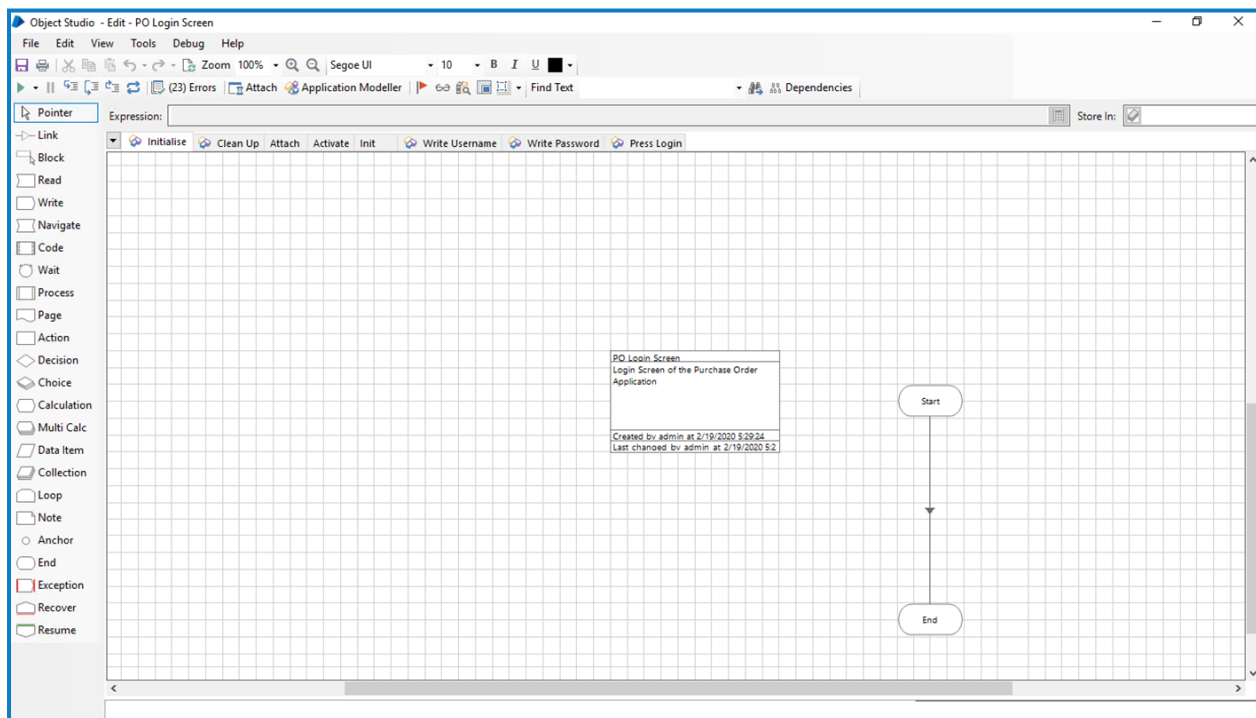


You should first drag and drop the wireframe Object into the respective Group folder before commencing any work.

Only a Blue Prism System Administrator role can access Objects at the top level of the hierarchy.

Objects need to be within Group folder before they can be worked upon by other Blue Prism roles. Once within a Group folder, by double-clicking on the Object, the Object Editor is launched.

The image shows the Object as designed within the Wireframer plugin.



The standard default pages for an Object, Initialise and Clean Up are provided as normal, as well as the three default pages that were configured in Wireframer, which are the Attach, Activate and Init. Then we have the pages for the Actions we added to the model.

Spying

Before we can complete the pages and assign the properties, we need to 'spy' the application using application modeller. This way we have all the elements captured that are needed to feed into the Action stages as properties. We are not going to cover 'spying' an application in this user guide however there are some key points when 'spying' that need to be performed. On the majority of the Actions that are created from Wireframer there are two Action stages This is again a best practice technique for automation design. This best practice is called primary and backup elements. When you are designing and 'spying' elements in the application modeller, we recommend the use of Primary and Backup elements wherever possible. A primary element is usually an element in the application modeller which:

- Has been spied in the most accurate mode (e.g. Win32 for thick client applications);
- Includes match attributes that specifically dictate where the application modeller can find this element in the application.

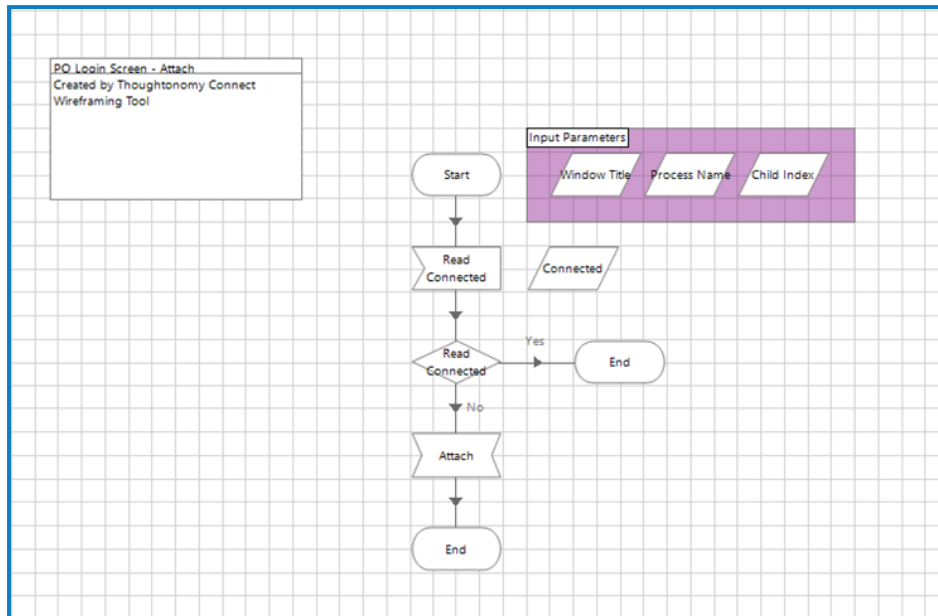
A backup element is usually an element in the application modeller which:

- Has been spied in the most accurate mode using an alternative method (e.g. AA mode for thick client applications);
- Includes match attributes that don't specifically dictate where the application modeller can find the element in the application. The match attribute criteria require that the application modeller, search for the element within the DOM based on the match attribute selection, or search within the application element hierarchy API.

The reasons behind the primary and backup elements is to improve speed and resiliency of the Object. If we consider a case where we need to press an 'OK' button on a web application. We would spy the web application using 'HTML' mode and use the 'Path' variable to match the attribute. However, if the web application is updated or due to additional HTML elements being added it has resulted in the spied 'Path' value becoming invalid the web application button wouldn't be found. Therefore, a second spied version of the element would provide backup resiliency if this occurred. Once we have multiple versions of an element spied, we can leverage a wait stage that waits for either the primary or the backup element to exist. Whichever of the elements is found first, then that element is used to send a click to (or appropriate action). Once all the elements have been captured we can go through each of the pages using the 'spied' elements to complete the Object.

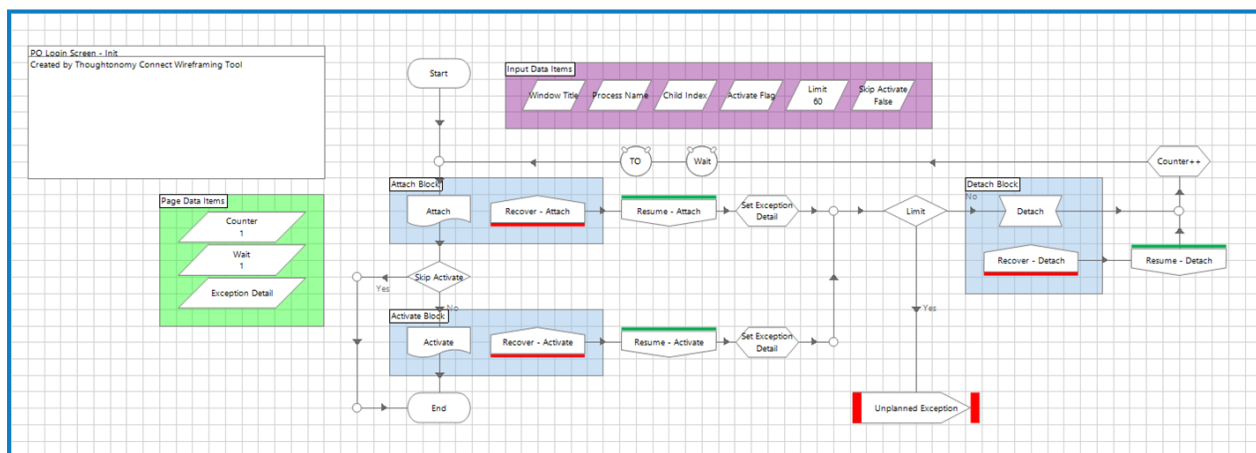
Attach page

The Attach page is complete, and no further changes are required.



Init page

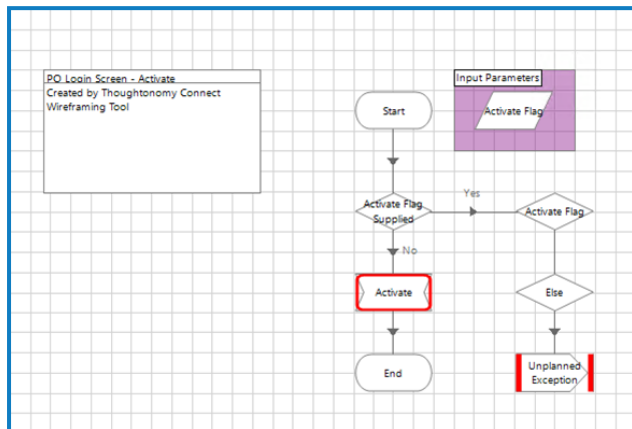
The Init page is also complete, and no further changes are required.



In the Init page in line with best practice techniques you can see the exception logic for recovery and resumption. This page, also as mentioned earlier, references the Attach and Activate Actions through a Page Reference stage, to ensure that the application is attached, and the appropriate window is brought to the foreground and focused.

Activate page

The Activate page does need an application element to be provided before it can be used, this is indicated by the red square, breakpoint, around the stage.



By double-clicking on the Activate Navigate stage you will see that details relating to the application element needs to be provided to the stage.

Navigate Properties

Name:

Description:

Application Explorer

Filter the tree...

- Application
 - Main Screen
 - Username
 - UsernameBU
 - Password
 - PasswordBU
 - Login Button
 - Login ButtonBU

Actions

Element	Params	Action	Inputs Set
Main Screen	...	Activate Application	N/A

Pause After Each Step (timespan/secs)

Move Up Move Down Add Remove

Inputs

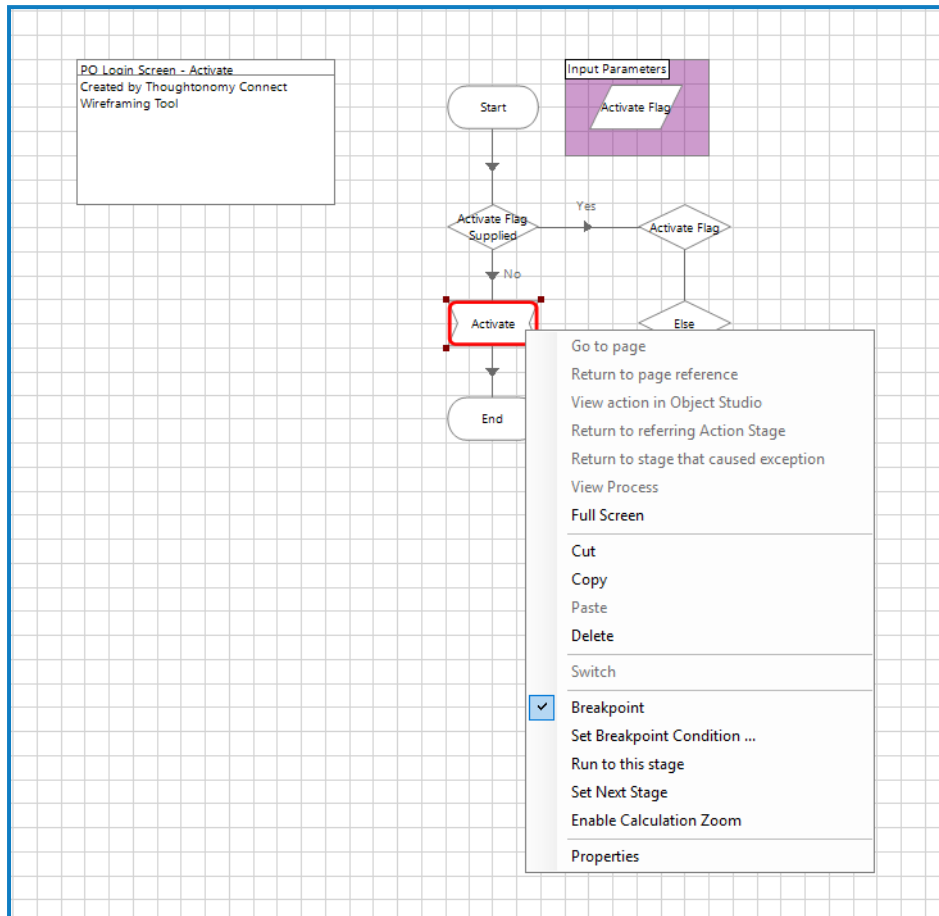
Name	Datatype	Value
------	----------	-------

Stage logging:

Warning threshold: Number of minutes (0 to disable)

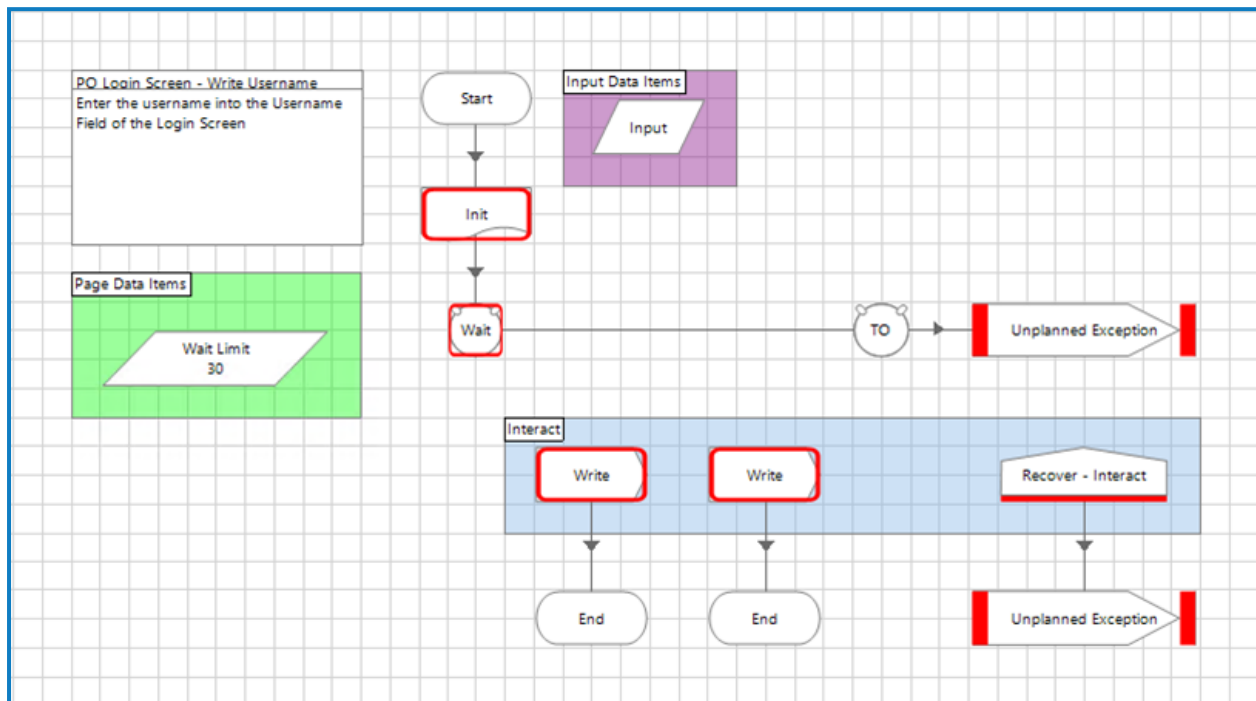
OK Cancel

Once you have configured the properties to Activate the application window. The next step is to clear the Breakpoint using the menu of a right-mouse click.

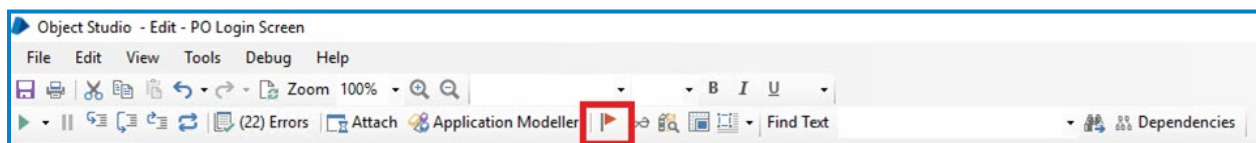


Wireframe action page

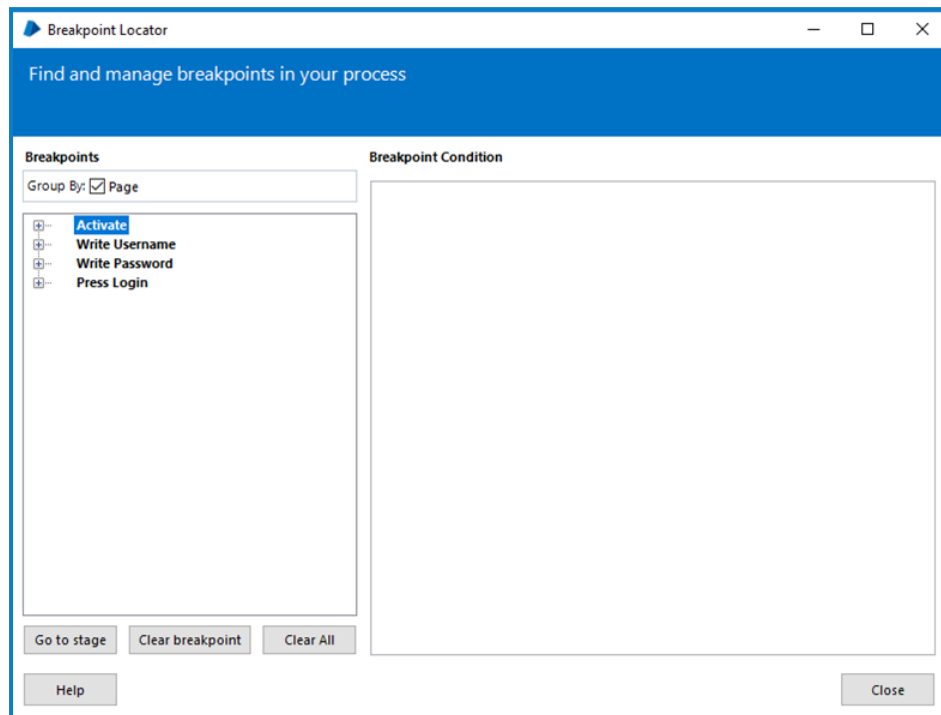
Using the Wireframer Write action as an example we can review the stages that need to be configured. The Write Username page in our example has numerous stages identified that require properties to be set or application model elements to be aligned.



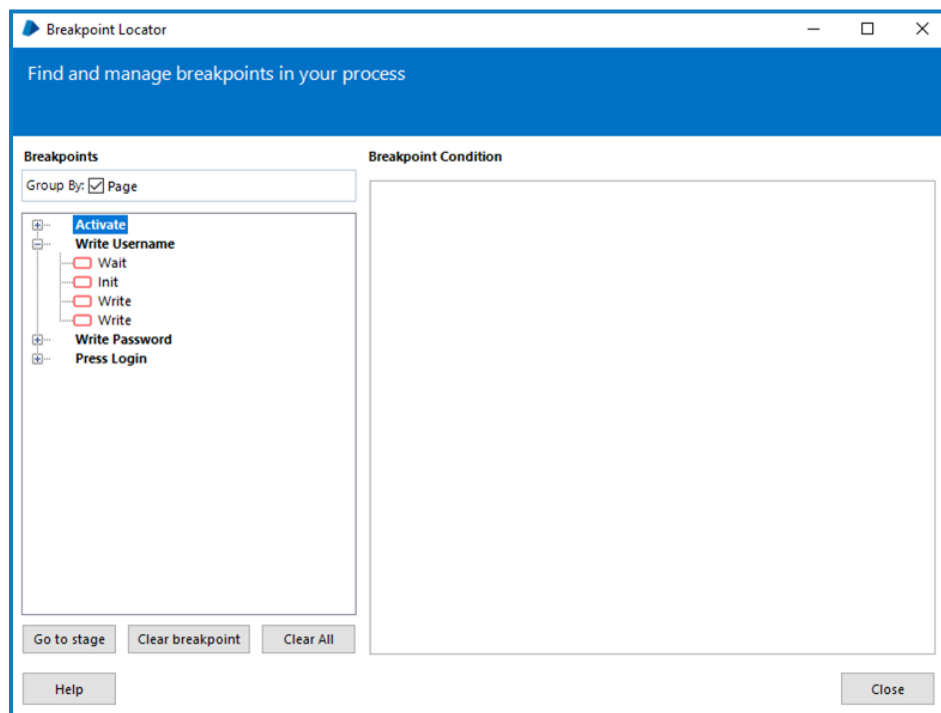
To work through the different stages that require attention, another method can be used; the View Breakpoints option. Invoked by selecting the flag icon in the main menu bar.



By selecting this command, the Breakpoint Locator form appears which lists all the breakpoints set within the Object.



The form lists all the breakpoints across the Object pages, these can be expanded and then used to navigate to the appropriate stage on the appropriate page, by selecting Go to stage.



This form also enables you to track your progress when adding properties and application model elements to all the stages that have breakpoints aligned to them. Once you have actioned all the changes required, the View Breakpoints command should produce a clear form with no breakpoints remaining. The Init Stage should be configured to the specific application being used. Once applied the Breakpoint is then removed.

The screenshot shows the 'Page Reference Properties' dialog box. It has a title bar with a question mark, minus, and close button. The 'Name' field is 'Init' and the 'Description' is 'Created by Thoughtonomy Connect Wireframing Tool'. Below this is a 'Page' dropdown set to 'Init'. There are three tabs: 'Inputs', 'Outputs', and 'Conditions'. The 'Inputs' tab is active, showing a table with columns 'Name', 'Data Type', and 'Value'. The table contains five rows: 'Window Title' (Text, 'Orders - Login'), 'Process Name' (Text, 'Orders'), 'Child Index' (Text, empty), 'Activate Flag' (Number, empty), and 'Skip Activate' (Text, empty). A 'Limit' row is at the bottom with a 'Number' data type and an empty value. To the right of the table is a 'Group' section with checkboxes for 'Page' (unchecked) and 'Data Type' (checked), and a 'View All Items' checkbox (checked). Below the table is a 'Stage logging' dropdown set to 'Enabled' and a 'Warning threshold' dropdown set to 'System Default'. There is also a 'Number of minutes' field set to '5' and a '(0 to disable)' label. At the bottom right are 'OK' and 'Cancel' buttons.

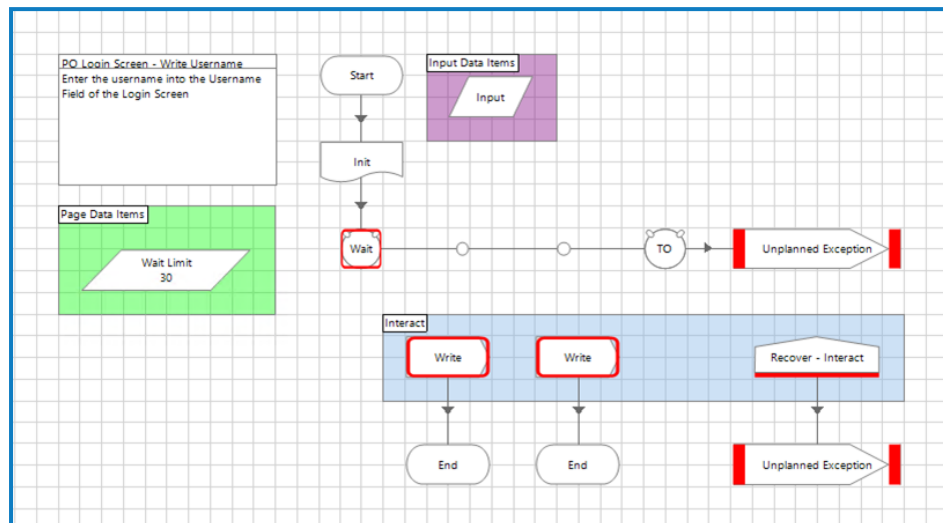
Name	Data Type	Value
Window Title	Text	"Orders - Login"
Process Name	Text	"Orders"
Child Index	Text	
Activate Flag	Number	
Skip Activate	Text	
Limit	Number	

When configuring the properties on the Wait Stage it is important to add two element rows, so that you build the resiliency within the Object. In the image below we show the two rows configured with the two 'spied' elements.

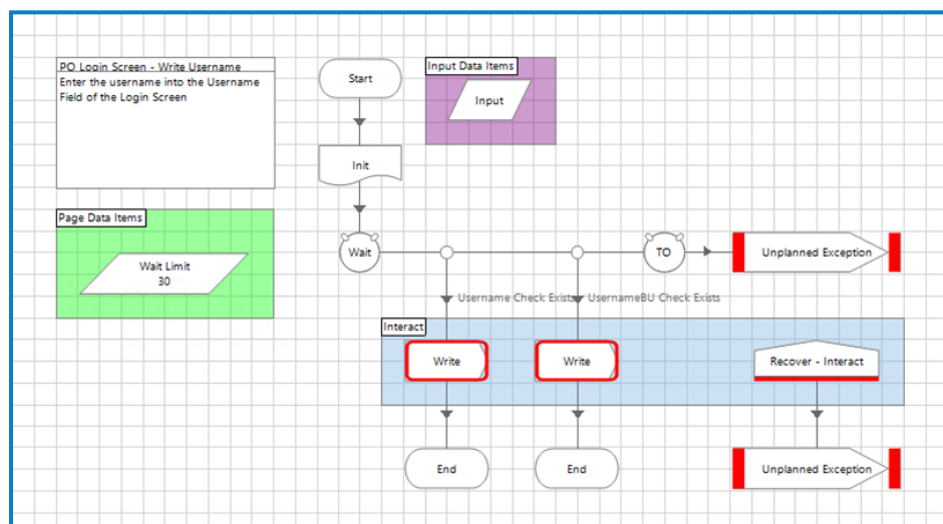
The screenshot shows the 'Wait Properties' dialog box. It has a title bar with a question mark, minus, and close button. The 'Name' field is 'Wait' and the 'Description' is 'Created by Thoughtonomy Connect Wireframing Tool'. Below this is an 'Application Explorer' section with a tree view showing the application structure: 'Application' -> 'Main Screen' -> 'Username' -> 'UsernameBU', 'Password' -> 'PasswordBU', and 'Login Button' -> 'Login ButtonBU'. To the right of the tree is an 'Actions' section with a table with columns 'Element', 'Params', 'Condition', 'Type', 'Comparison', and 'Value'. The table contains two rows: 'Username' (Check Exists, Flag, = (Equal), True) and 'UsernameBU' (Check Exists, Flag, = (Equal), True). Below the table are 'Move Up', 'Move Down', 'Add', and 'Remove' buttons. To the right of the table is a 'Data Explorer' section with checkboxes for 'Page' (unchecked) and 'Data Type' (checked), and a 'View All Items' checkbox (checked). Below the table is an 'Inputs' section with a table with columns 'Name', 'Datatype', and 'Value'. The table is empty. Below the inputs is a 'Timeout (timespan/secs)' field set to '[Wait Limit]'. At the bottom right are 'OK' and 'Cancel' buttons.

Element	Params	Condition	Type	Comparison	Value
Username		Check Exists	Flag	= (Equal)	True
UsernameBU		Check Exists	Flag	= (Equal)	True

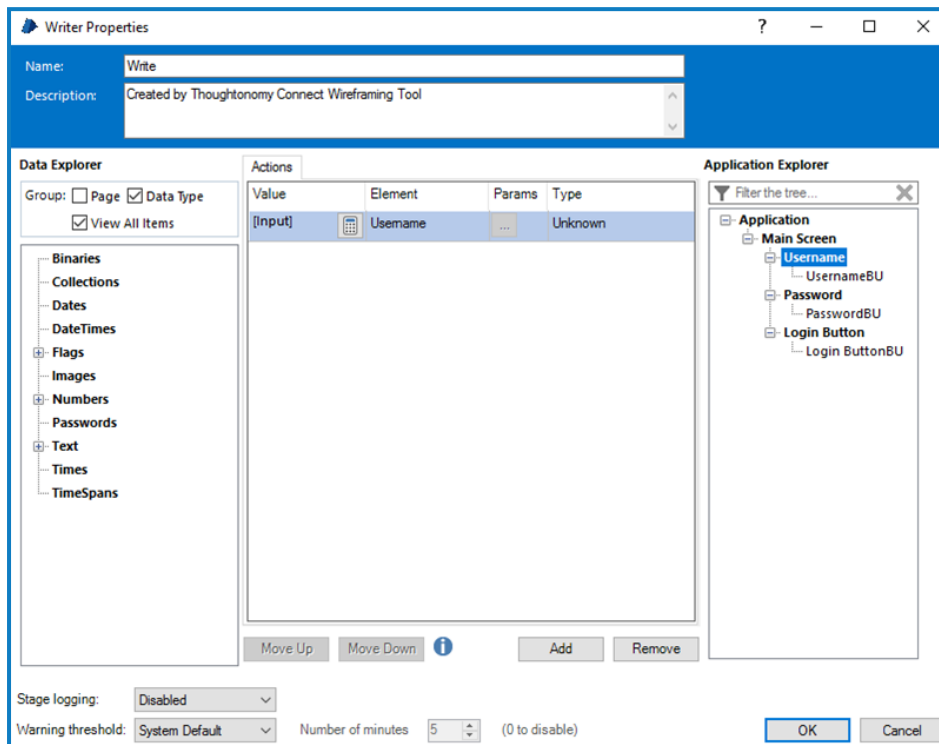
Once the properties are saved, two anchor points are added to the diagram which need to be connected to the two, in our example, Write Stages.



Once connected and tidied up, the Breakpoint for the Wait Stage can be removed.



It is important that when configuring the Write Stages that the correct elements are used which correspond to the two Wait Stage conditions we set, so in our example the left condition is 'Username Check Exists' so using the Primary element. Whilst the right shows 'UsernameBU Check Exists' the Backup element. The left Write Stage is therefore configured.



Completing the second Write Stage in a similar way but using the Backup element, then removing the two Breakpoints completes this wireframe Action page.

