
AUTOMATION ANYWHERE ENTERPRISE 11 LTS

MetaBot Designer - User Guide

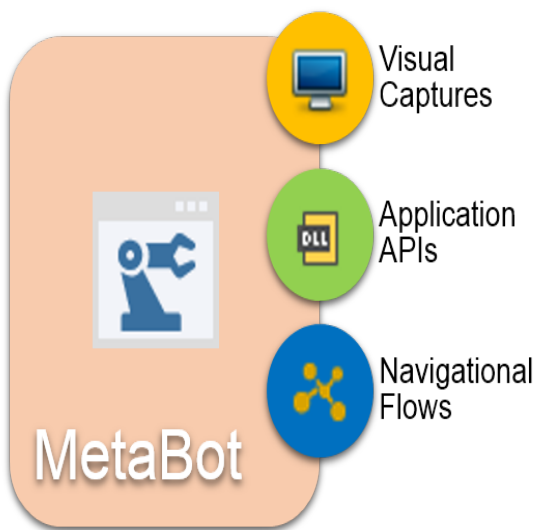
Section: MetaBots - Getting Started

AAE Client MetaBot Designer Overview

A MetaBot is a highly re-usable automation blueprint of an application. This automation blueprint can be constructed using:

- **Visual captures** - These are GUI components (screens) of an application.
 - In MetaBot Designer, a visual capture is referred to as Screen.
- **Application APIs** - These are interfaces that allow low level operations of an application by circumventing GUI.
 - The MetaBot Designer also supports the most common format of API on Windows platform - the DLL.
- **Navigational flows** - These are pre-configured use cases of an application and leverage Visual captures and APIs.
 - In MetaBot Designer, Screens and DLLs form the Assets using which you can define and pre-configure any use case of a target application to create a navigational flow, known as Logic.

[Learn More](#)



Benefits of using MetaBots

Using MetaBots in automation tasks have the following benefits:

- MetaBots are highly re-usable; create once, use everywhere. They show up in automation command library in AA Enterprise and can be leveraged by any automation task.
- Enterprises can leverage MetaBot library to standardize org-wide automation in a rapid manner.
- MetaBots ensures systematic, accelerated automation ROI.
- MetaBots help to eliminate common navigational errors in complex automation tasks.
- MetaBots help to automate without requiring access to live application
- MetaBots can be easily calibrated to newer versions of applications to ensure compatibility.

The MetaBot Designer

The MetaBot Designer helps you to conceptualize, create, manage and upload MetaBots.

With MetaBot Designer, you can:

- Create MetaBots.
- Add Assets - Screens and DLLs to your MetaBots.
- Manage your MetaBot Assets.
- Configure and Calibrate the Screens.
- Create custom objects on the Screens.
- Create Logic to represent a navigational flow using Screens and DLLs.

- Upload the MetaBots to the Control Room for use by other Enterprise Client users.
- Edit MetaBots created by other users.
- Export and Import MetaBots for use in different Control Room setups.

Installing the MetaBot Designer


The MetaBot Designer comes pre-installed with Automation Anywhere Client.

To access MetaBots, the Client user has to have the required access privileges that are provided by the Control Room administrator.

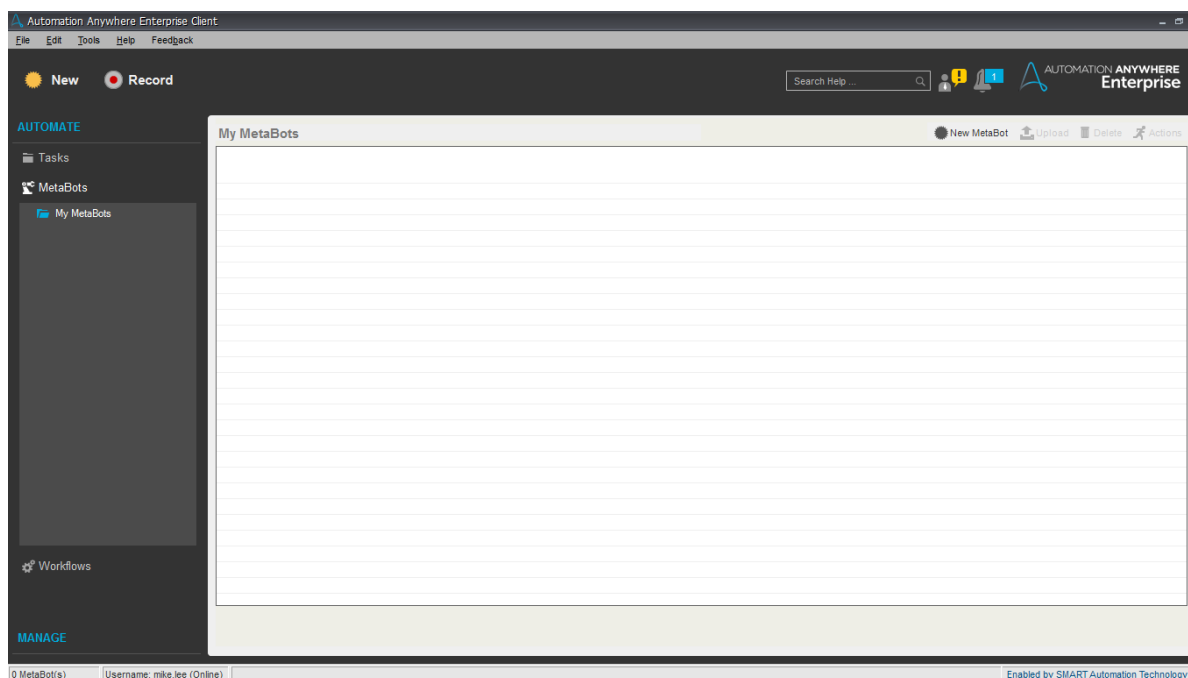
- To do a quick check that MetaBot Designer is installed and working fine, ensure that you see MetaBots under AUTOMATE on left panel.
- Click on MetaBots tab to launch the My MetaBots view.
 - If you are using MetaBots for the first time, the list does not display any file. You can simply start creating MetaBots using either New or Record option. [Learn More](#)
 - However, if you have upgraded to the current version, you will see MetaBots created in earlier version(s) listed under My MetaBots.
- To access the MetaBots Designer console, you need to click on Edit. [Learn More](#)

Launching the MetaBot Designer

After successful installation of Automation Anywhere Client setup, the feature is added to the product under the Automate Tab.

 Note: You will have to exit and restart Automation Anywhere Client for the changes to take effect, if you have upgraded to the current version.

Click on 'MetaBots' to launch the Enterprise Client - MetaBot Designer.



Understanding Enterprise Client - MetaBots

A BotCreator Client with MetaBots Repository access permissions can create, record, edit, upload, download, and rename MetaBots from the Client - My MetaBots list view.

From the Enterprise Client - MetaBots view, a BotCreator can create a new MetaBot, record screens for a new MetaBot, upload and delete MetaBot(s).

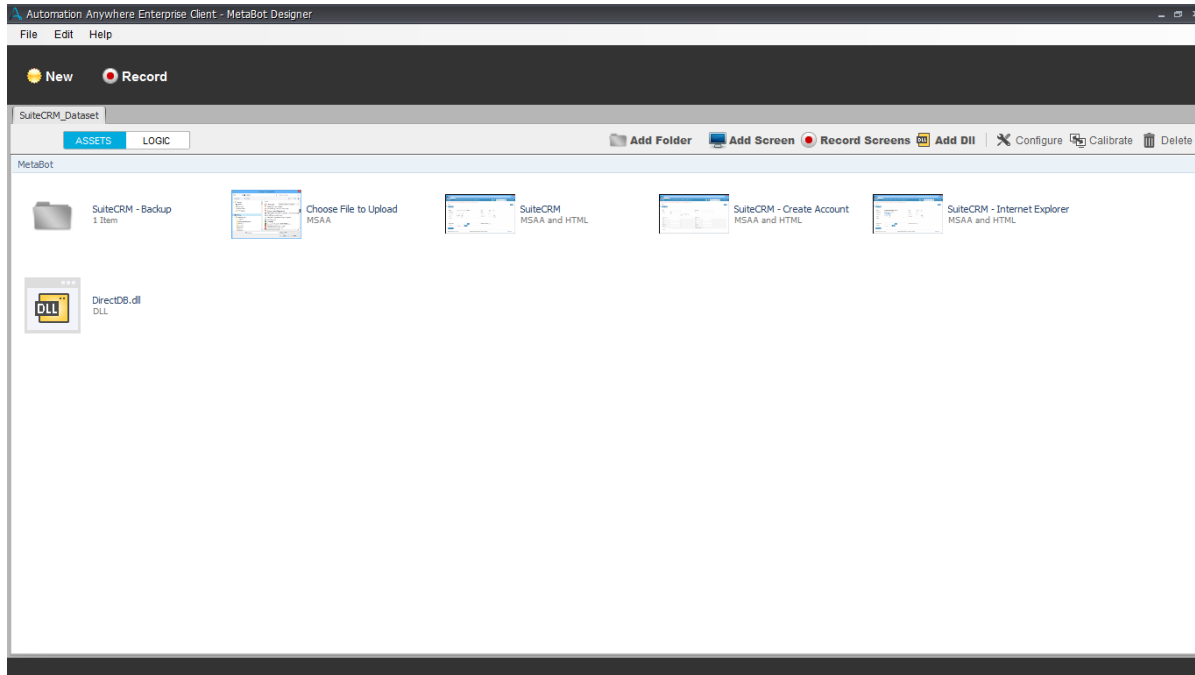
Additionally, if Version Control is enabled for controlled edits of MetaBots, a BotCreator can Check Out the MetaBot for editing and view Version History. [Learn More](#)

From the My MetaBots list view, a BotCreator can perform the below actions:

1. **New** - Use this to create a New MetaBot. [Learn More](#)
2. **Record** - Record is an alternate method of creating a MetaBot. [Learn More](#)
3. **Upload** - Upload the MetaBots to the Control Room to enable other BotCreators to download for TaskBot integration. [Learn More](#)
4. **Delete** - Use this to delete obsolete MetaBots or ones that are no longer required. [Learn More](#)

Some of these features are also accessible from the MetaBot Designer console. [Learn More](#)

Understanding MetaBot Designer

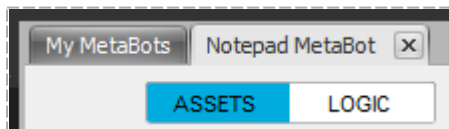


Using the MetaBot Designer, you can:

- Specify folders in an existing MetaBot,
- Add DLL's to create low level operations of an application by circumventing GUI
- Add and record new screens to a MetaBot;
- Create simple independent yet functional logic blocks.
- Also, create MetaBots and upload them to Control Room to create a repository for using/reusing MetaBots in automation tasks.

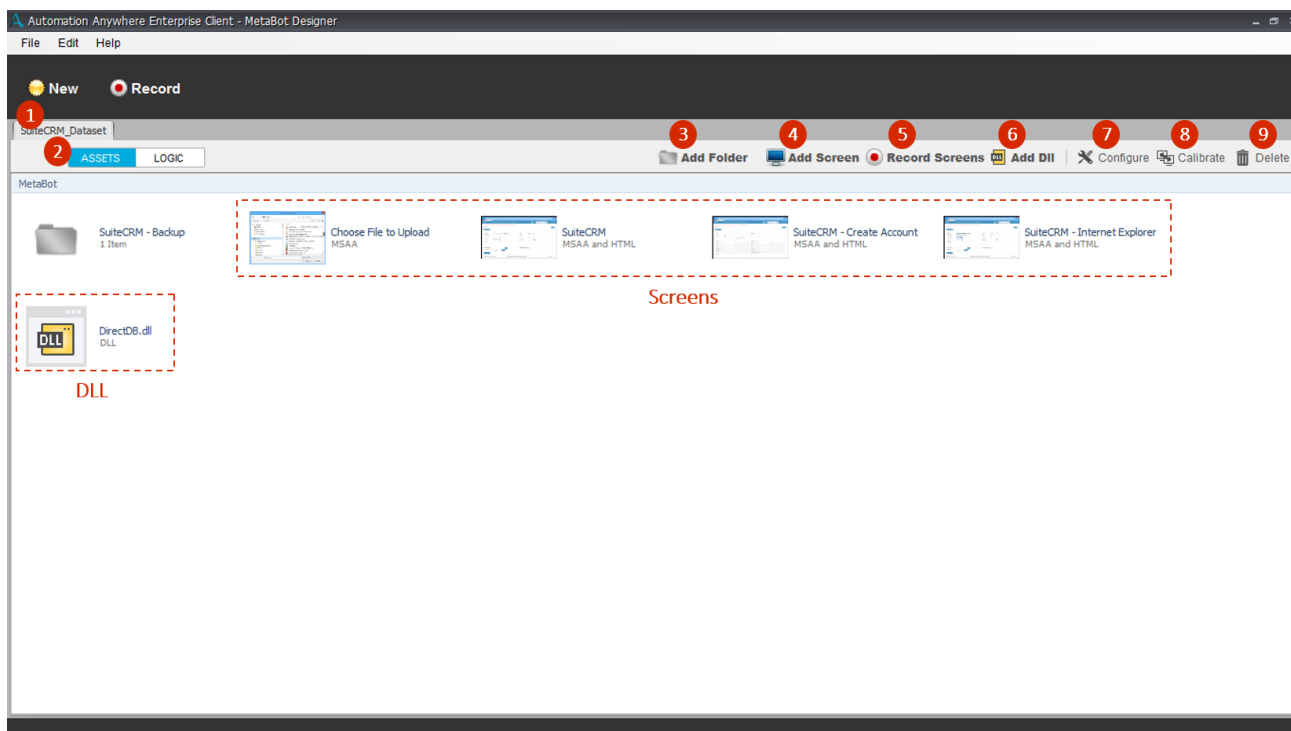
MetaBot Designer Decoded

The work space in MetBot Designer allows you to work with multiple MetaBots at the same time. MetaBots that you wish to work on can be kept open in the form of tabs.




Once you create a MetaBot, you are taken to Assets tab by default where in you can start by adding new assets (Screens/DLLs).

Understanding Assets




The following explains the options available within an Assets view:


1. **MetaBot Tab** - The MetaBot opens in its own tab. This tab is dedicated to the Assets and Logic for that particular MetaBot.
2. **Assets Tab** - When highlighted it indicates that the view is open in 'Assets'; it displays all the Screens, DLLs and Folders inherent to this particular MetaBot. It is selected by default for a new MetaBot.
3. **Add Folders*** - You can organize your MetaBot using 'Folders'. This will enable you to easily manage all your screens and dll's that are to be uploaded/have been uploaded.
[Learn More](#)
4. **Add Screen*** - When you require to capture a single screen for an application executable that is running, this feature is extremely useful.
[Learn More](#)

 Note: You will have to invoke the application screen before using this feature. Also, if the application is closed, you will be prompted to open the required application.

**If your Screens are set at lower resolutions; e.g. 1024 X 768, the 'Add Folder' and 'Add Screen' options can be accessed from the 'Edit' menu.*

5. **Record Screen** - When you need to capture multiple screens of the related application/webpage at one go, use Record Screen. Every screen / Menu item / Popup / context menu that you interact with during the recording gets captured.
[Learn More](#)

 Tip: Use Record Screen to record all the Screens/UI elements (Menu item / Popup / context menu etc.) while you interact with the application in a workflow mode. These UI elements cannot be captured using Add Screen.

 Note: If an application has multiple exe's, you are required to create a separate MetaBot for each.

6. **Add DLL** - If you need to use an 'Application Programming Interface' (DLL) within your MetaBot, you can add it to the MetaBot using 'Add DLL'. Remember though, you cannot include special characters in DLL names.

[Learn More](#)



Note: The Screens, DLLs and Folders are displayed in the order they were added.

7. **Configure** - Use this to edit properties for the recorded/added screen. Here you can provide aliases such as a 'Screen Name' and a 'Screen Title'. You can also select an object to define its properties such as Name, Path, Value, ID, Class, Index, States etc and the 'Play Mode' to be used when running tasks. Some of these properties help to uniquely identify an object during playback. You can thus use configure to improve the reliability of your automation.

[Learn More](#)

8. **Calibrate** - Since an application may get updated continually during its lifecycle with improvements and newer features, your captured screen and its object properties might need a re-look after every update of the application. In MetaBots Designer you can use 'Calibrate' to instantly compare an existing screen with a newer screen to identify changes if any.

[Learn More](#)

9. **Upload** - The MetaBots can be 'uploaded' to the Server i.e. uploaded to the Control Room, which acts as the central library from where fellow MetaBot Designers can pick up the MetaBots necessary to their task(s). The client with MetaBot privileges can upload and deploy the MetaBots to the server.

[Learn More](#)

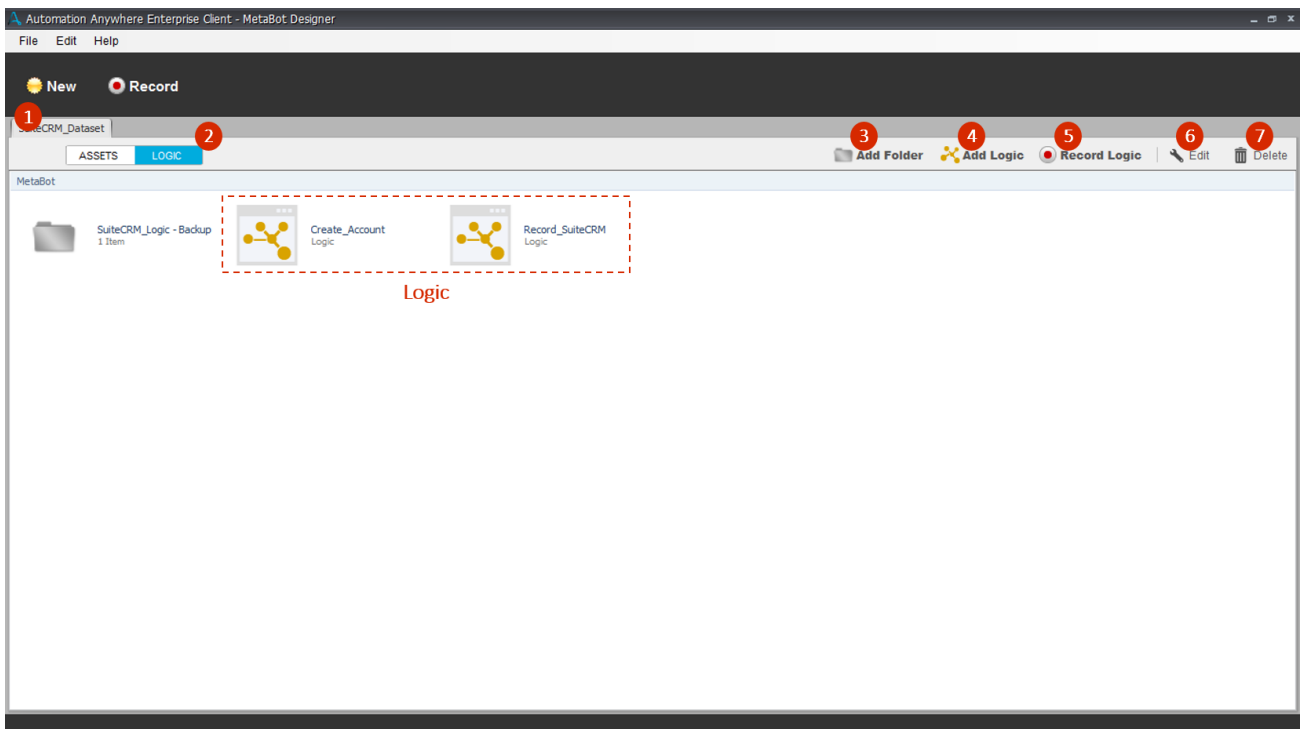
10. **Delete** - Use this to delete MetaBots that are no longer required.



Note: Deleting a MetaBot from the MetaBot Designer doesn't delete it from Control Room.

Once you have captured the desired assets, move on to create Logic using those assets.

Understanding Logic



A Logic is an independent yet functional unit of a process that represents a part navigational flow of an application and can be integrated into automation tasks as and when required.

You can use Assets (Screens and DLL's) to design a Logic block. Subsequently, you can upload the Logic Blocks to Control Room so that they can be downloaded to Development/Runtime Client(s) with appropriate MetaBot privileges.

The following explains the options available within a Logic view:

1. **MetaBot Tab** - The MetaBot opens in its own tab. This tab is dedicated to the Assets and Logic for that particular MetaBot.
2. **Logic Tab** - When highlighted it indicates that the view is open in 'Logic'; it displays all the Logic Blocks and Folders inherent to this particular MetaBot.
3. **Add Folder** - Similar to Assets; for instance, you can add Logic Blocks that are functionally similar to a folder.
4. **Add Logic** - Use this to create your navigational flows in the Logic Editor.
[Learn More](#)
5. **Record Logic** - Use this to record the logic flow and automatically save Screens in Assets.
[Learn More](#)
6. **Edit** - Use this to edit an existing navigational flow.
7. **Upload** - Use this to publish (upload) your (new or edited) Logic to the Control Room.
[Learn More](#)
8. **Delete** - Use this to delete obsolete Logic.



Note: The Logic and Folders are displayed in the order they were added.

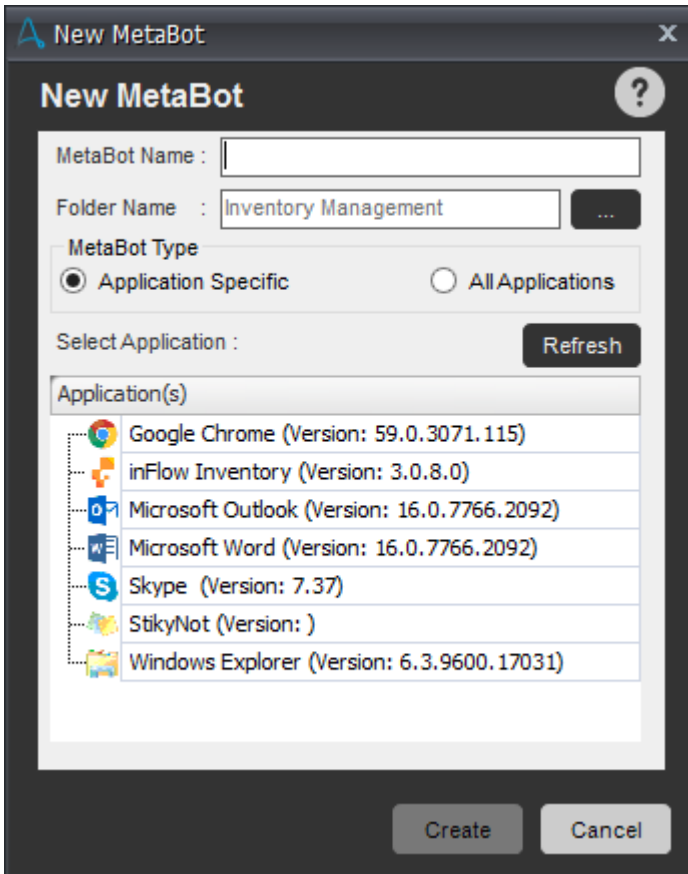
Creating a MetaBot

Let's create your first MetaBot. It will allow you to add vendors to your Inventory Management System.

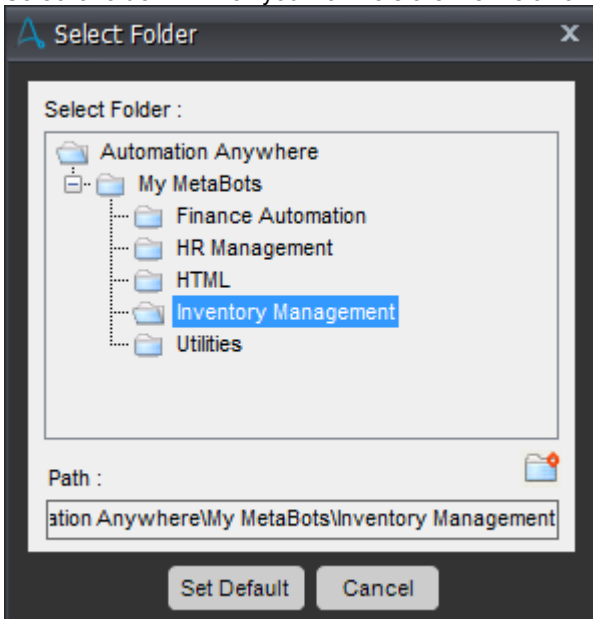
1. Open the MetaBot Designer and click on **New**.





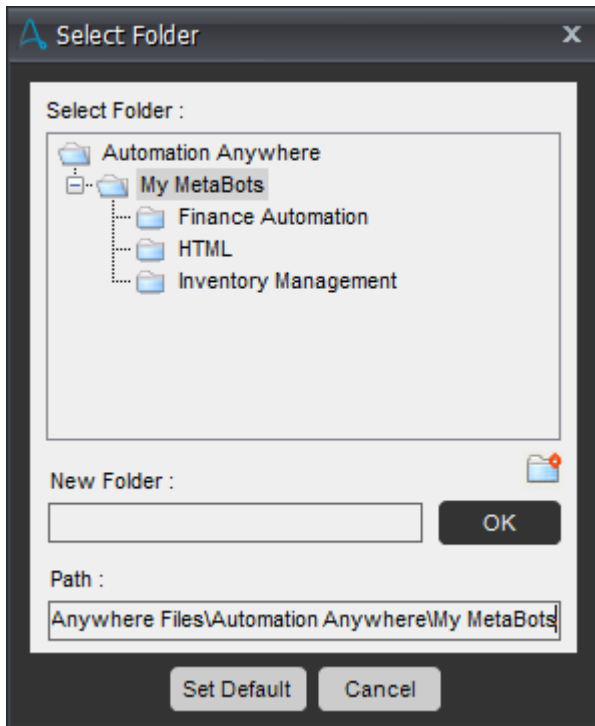
2. A smaller window, **New MetaBot**, opens which lists all currently running applications on your machine.



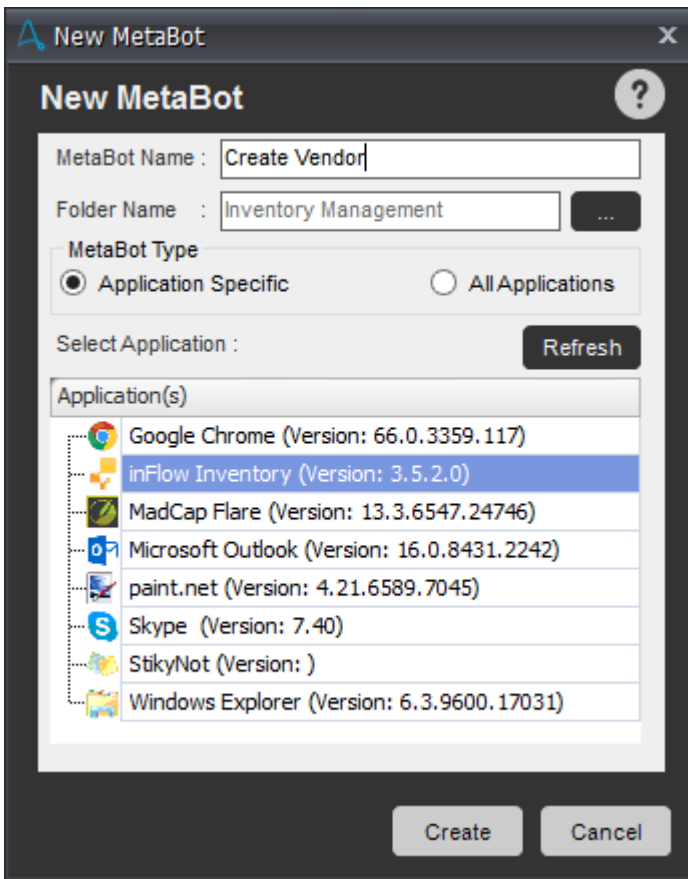
3. Opt to create either a MetaBot with a single application or with multiple applications. When you choose **Application Specific** option, you can create automation using screens only from that particular application. Whereas when you choose **All Applications** option, you can create automation using a combination of screens from different applications per your automation flow.
4. Select a folder in which you want to store the MetaBot by clicking the browse button. This launches the **Select Folder** window:




 **Tip:** Save your MetaBots to logically created folders. These can be created by clicking  :

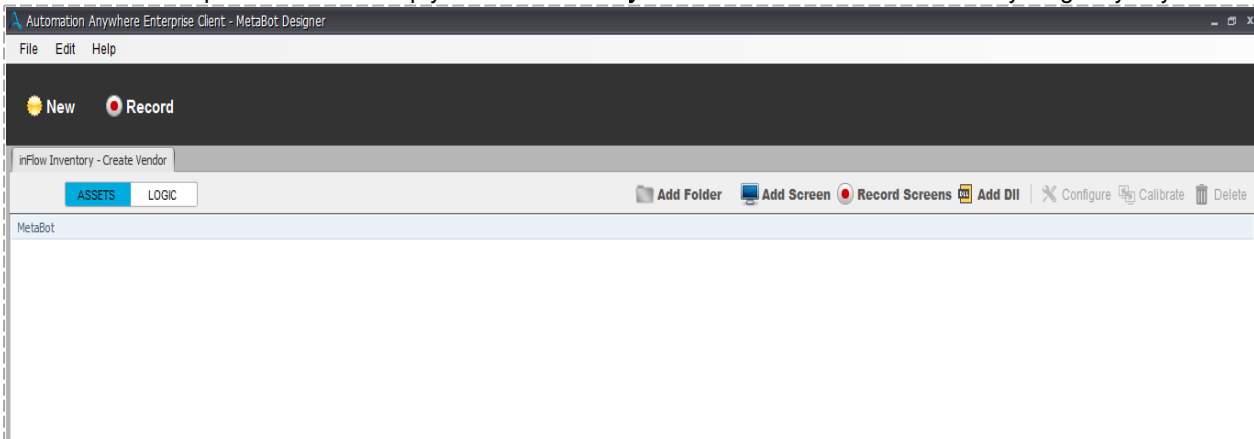


- To select a default folder to save your MetaBots click **Set Default**.
 - Click **OK** to return to the **New MetaBot** window.
5. For easy understanding, lets create a MetaBot using a single application. Select **Application Specific** .
 6. From the list, select an application (inFlow Inventory in this case) for which you want to create a MetaBot. Type in a name - "**inFlow Inventory - Create Vendor**."
 7. Click **Create**.



 **Tip:** If you forgot to launch the application before starting to create a MetaBot for it, just open it now and click on **Refresh** on the New MetaBot window. Your application will appear for selection in the list.

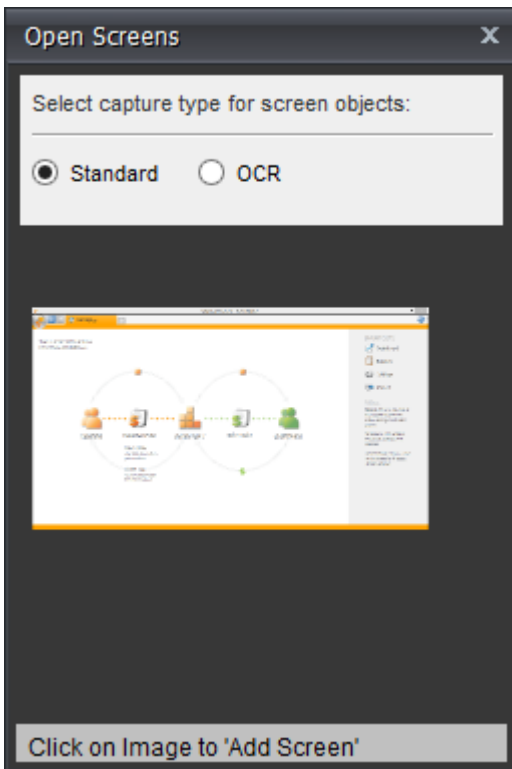
8. This creates and opens a new but empty “inFlow Inventory - Create Vendor.”. It cannot do anything for you yet.



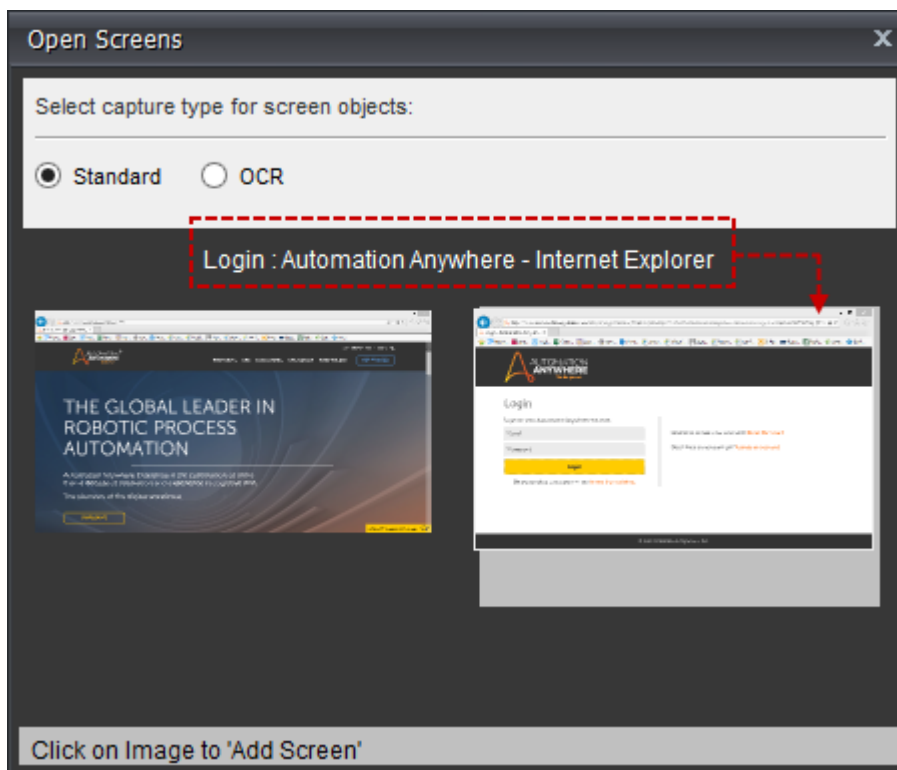
9. To create a vendor in the application, we need its screen asset. Click on 'Add Screen'.




10. On the center of your screen, a small window **Open Screens** is displayed. This window will show all instance of currently open screen of the target application.




- If multiple instances of an application are open, you will see multiple screens. For example, if you have two instances of IE open, you will see two separate screens displayed in **Open Screens**.



11. Select the capture type for screen objects - **Standard** or **OCR**.

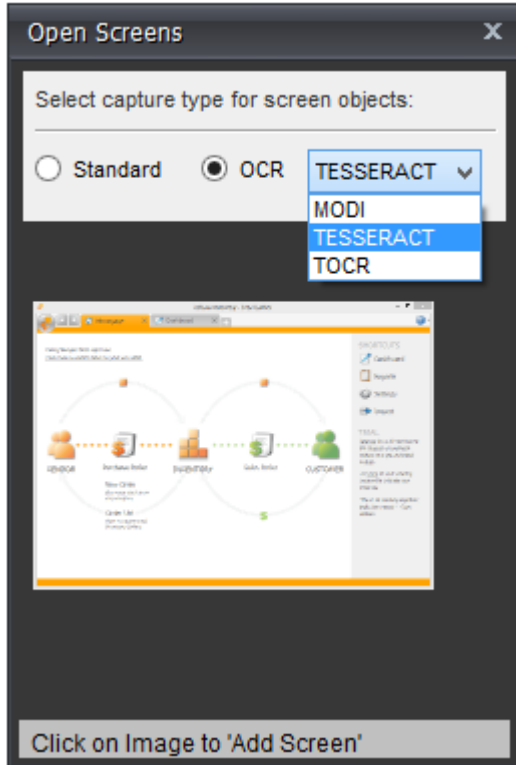
 **Note:** This option is available only for creating a screen and not when you record a screen.


- Select **Standard** when you are using technology such as MSAA, html etc. This option is selected by default.
- Select **OCR** when you want to capture objects from application images which are exposed over applications such as Citrix or RDP. While capturing screens with OCR ensure that the screen **DPI** is set to **Standard** or **100%**.

 **Note:** If you want to create a Logic for OCR Screens, it is recommended that you use Import Dataset command.

- When you select OCR, parent object properties are linked to child objects automatically while creating the Screen. You can choose to change the linking while configuring the screen for greater accuracy.

You can also choose the OCR engine to capture objects from screens:

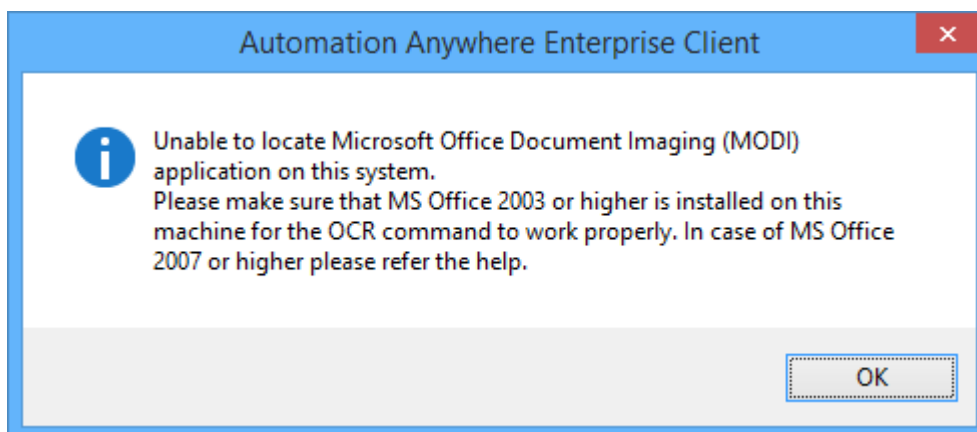


 **Note:** The OCR engine that you choose will be used for executing automations. It cannot be changed when you edit a MetaBot screen. Hence, the selected OCR engine should also be installed on the Bot Runner machines to ensure your automations do not fail.

- To use an OCR engine, you must ensure it is installed on your machine. **TESSERACT** is the default OCR engine. You can also opt to capture your screen objects using **MODI** or **TOCR**.

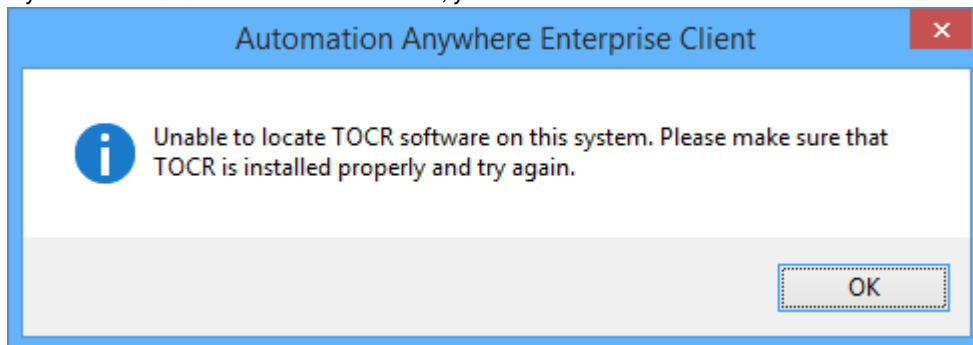
If you are using Tesseract, you will have to install Visual C++ Redistributable for Visual Studio 2015. You can download it from Microsoft's [website](https://www.microsoft.com/en-us/download/details.aspx?id=48159). To use an OCR engine, other than Tesseract, you must ensure it is installed on your machine.

- If you select **MODI** and it is not installed, you are shown:



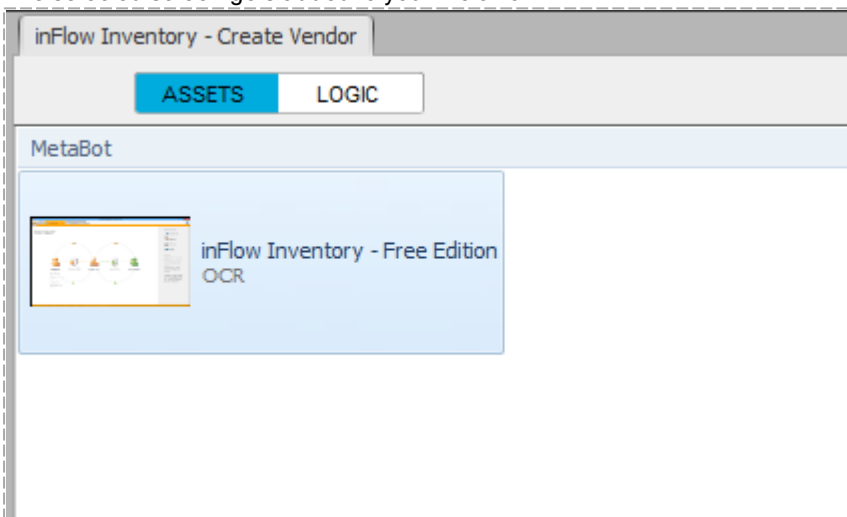
When you use **MODI** as your OCR engine, ensure that:

- Microsoft Office 2003 or later is installed on your computer.
- The sub-component "Scanning, OCR, and Indexing Service Filter" (under Microsoft Office Document Imaging) is selected during installation for Microsoft Office 2007.
- The component is installed separately if you are using Microsoft Office 2010 and above. Click [here](#) for details.
- If you select **TOCR** and it is not installed, you are shown:



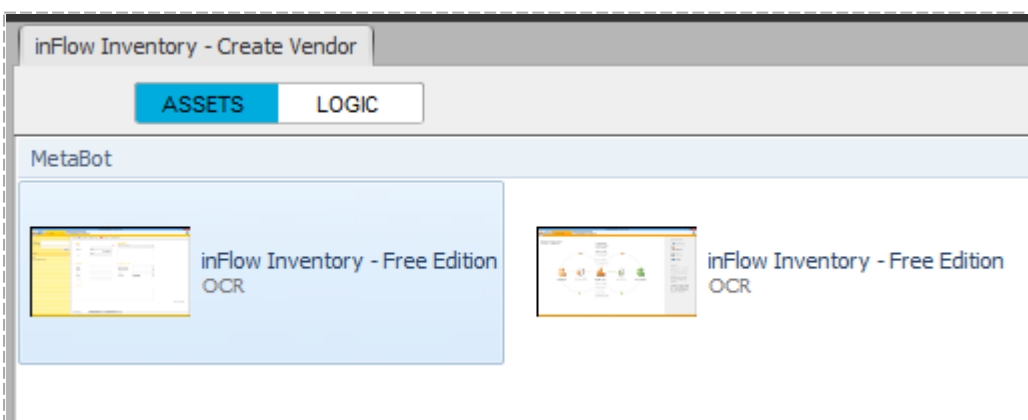
12. Click on the inFlow Inventory home screen instance to proceed.

13. The selected screen gets added to your MetaBot.



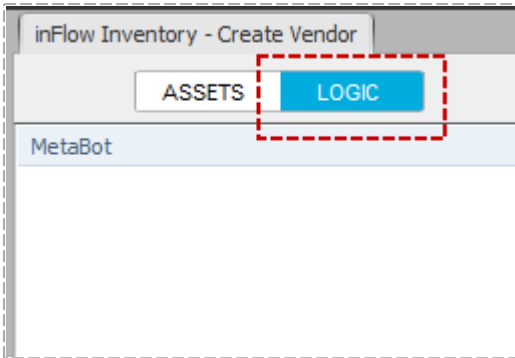
14. The next step would be adding more screens to your **Assets** library. Use either the 'Add Screen' or 'Record Screen' option. Refer [Adding and Recording Screens](#) for details.

15. Let's add another screen to **Assets**. This screen will be used to add vendor names to the inventory repository.



16. You can configure these Screens to edit object properties which help in uniquely identifying the associated object during automation execution. Refer details in [Configuring MetaBot Screens](#).

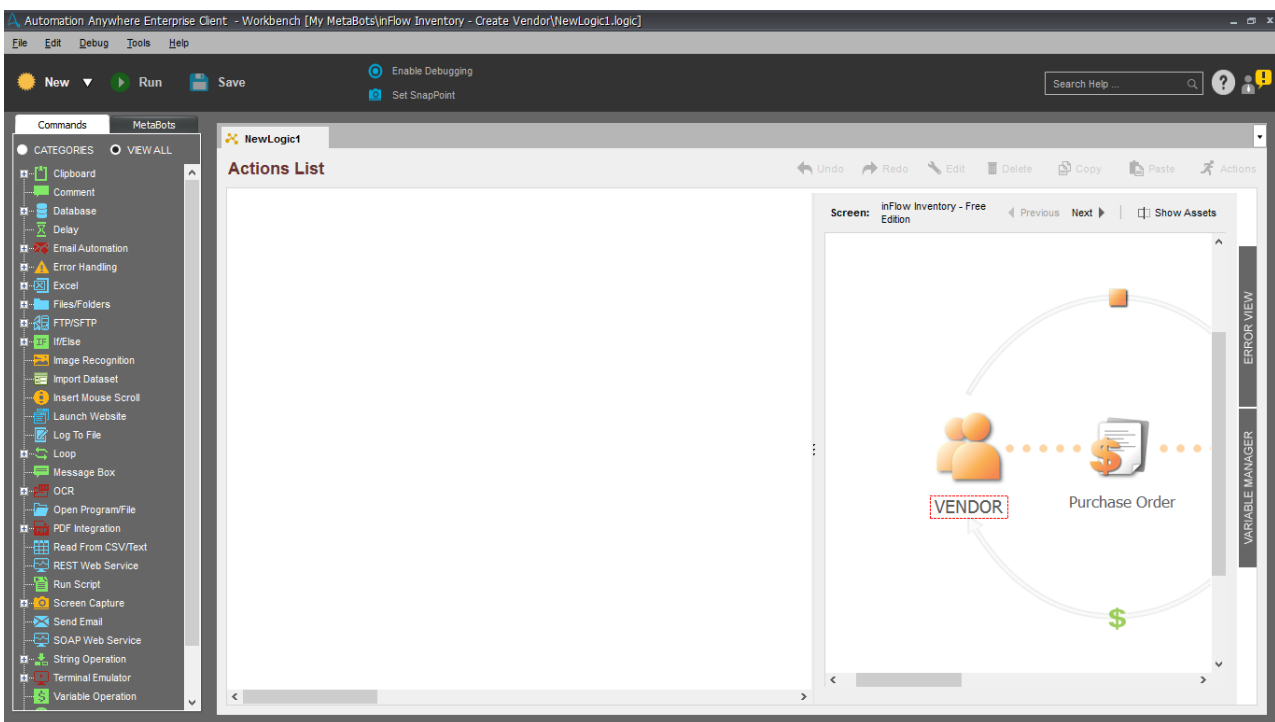
17. Let's use these screens to create a logic to create a Vendor. Click on **Logic** tab. Refer [Using the Workbench to create Logic](#) for details.



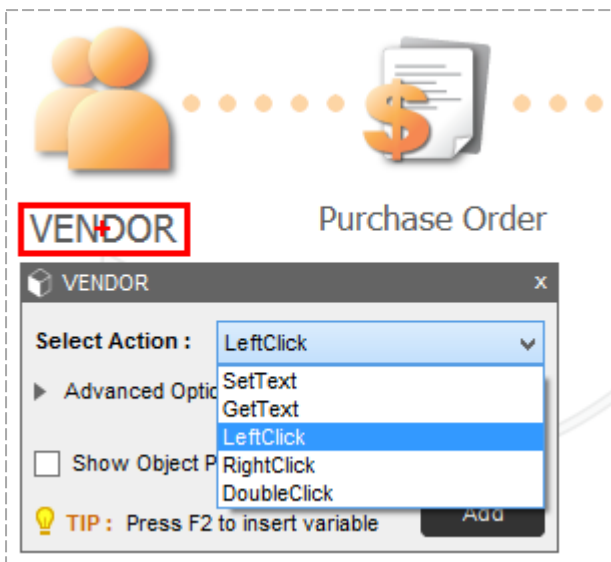
18. Click **Add Logic**.



19. This opens the **Workbench**, a place where you define the navigational flow by selecting screens/dlls, add commands, define logic, save and play the flow to see if it works as expected.

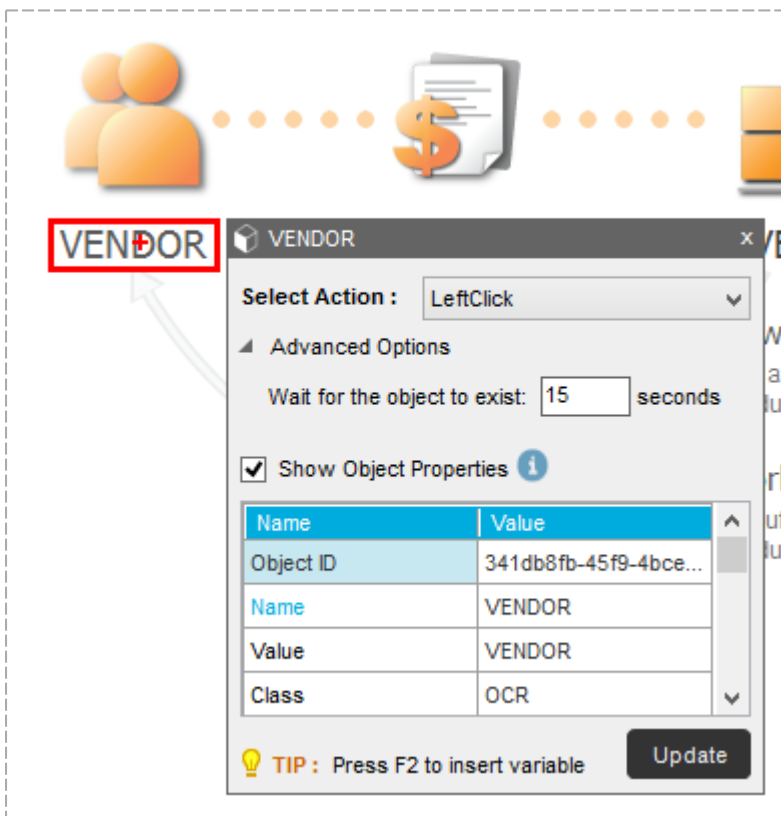


20. Click on **Vendor** in the screen that was captured first. This will open a properties window near the selected object that will allow you to configure **Actions**.

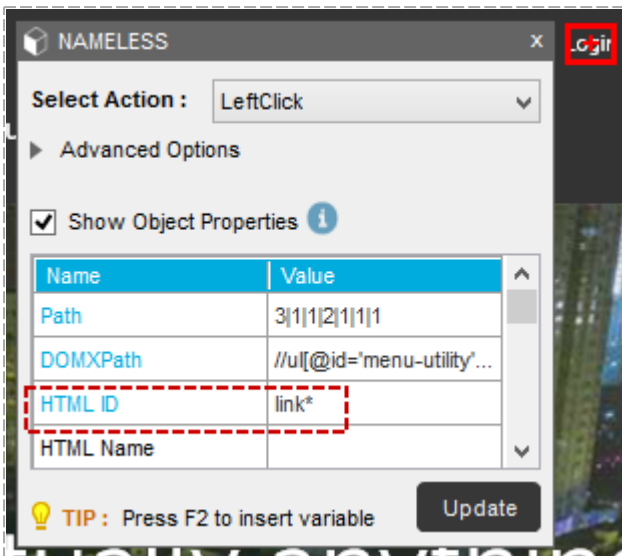



21. Select **LeftClick** from the list of actions.
22. In **Advanced Options** input the Wait time for the object to load. Select **Show Object Properties** to view the properties that will be used to search the object during play time.

The Object Properties are configured based on the Search Criteria selected when configuring screens. The object property selected for search is shown in blue font. For details refer [Configuring MetaBot Screens](#).

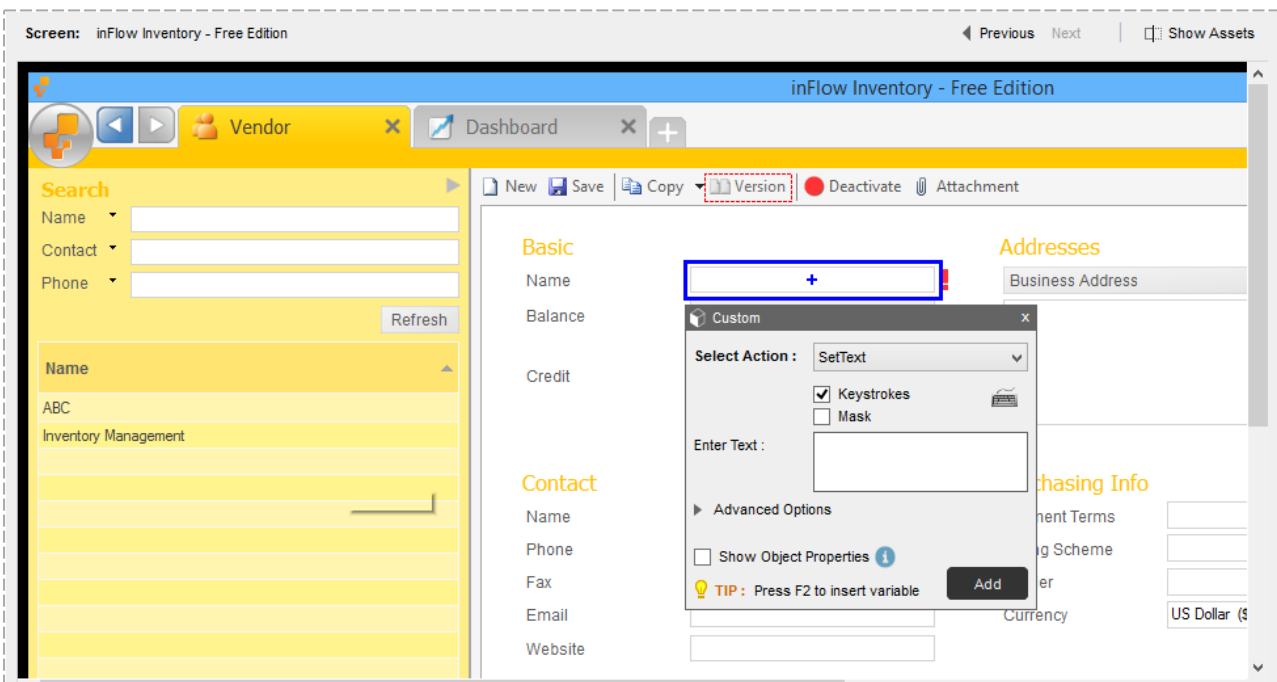


Tip: If the selected object has dynamic properties in its search criteria, it is recommended to use '*' in place of the actual value. For instance, change the value of property 'HTML ID' as shown:

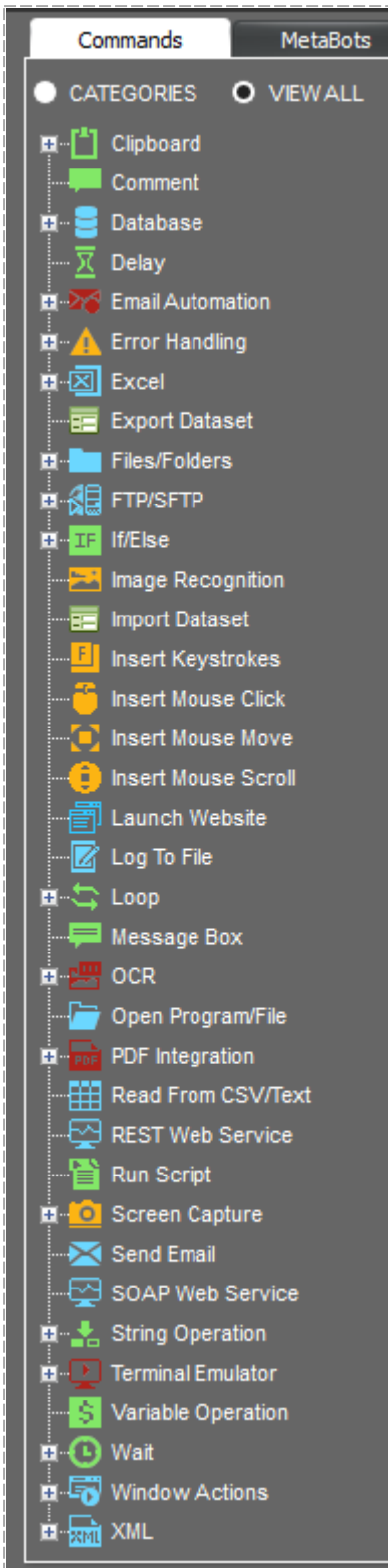


 Tip: If no name is displayed for the selected control, you can give an easily identifiable name to it. You can also rename the selected control if required. [Learn More](#).

- Subsequently, add other screens by clicking 'Next'. For instance, the above action will navigate you to the create vendor screen wherein you can create a vendor and add details:

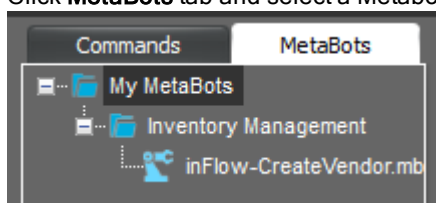


24. If your Logic requires that you add commands, select ones that are most appropriate for your logic. [Learn More](#)

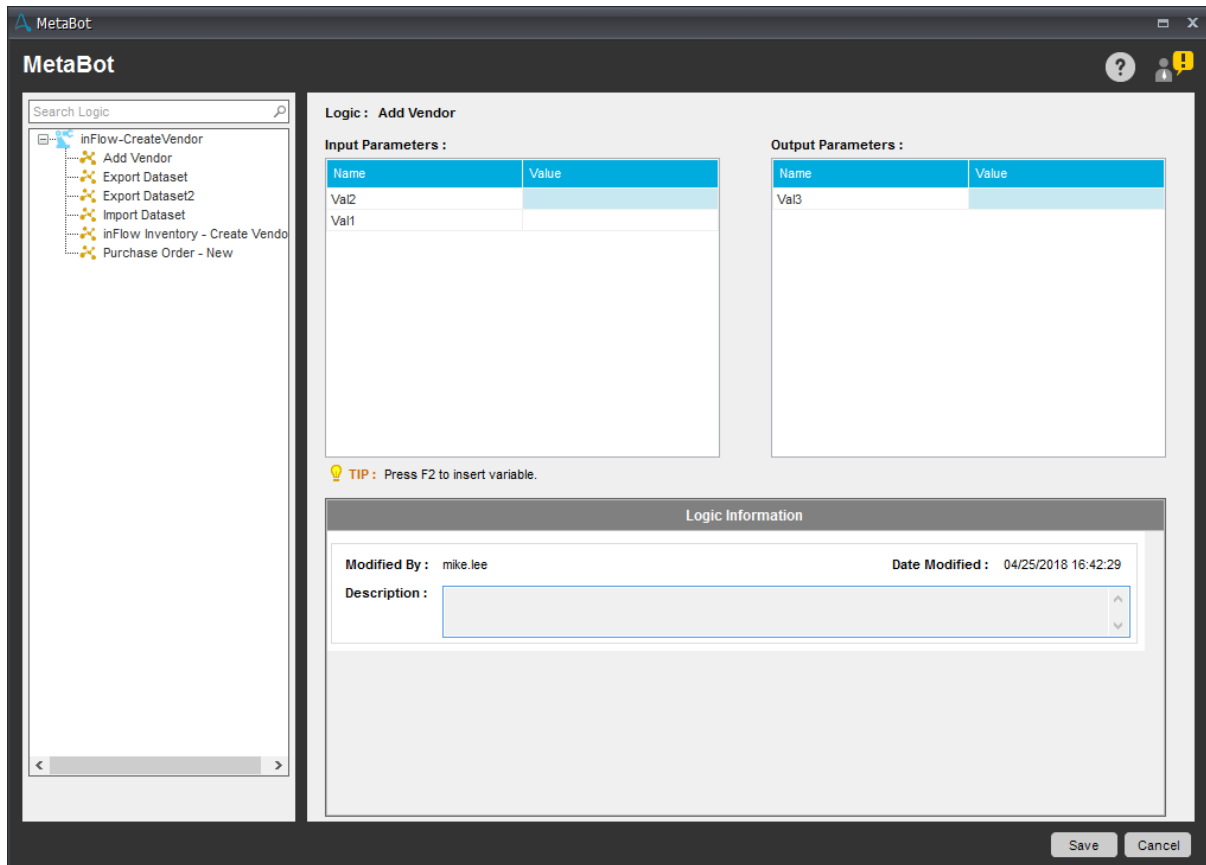


25. You can also choose to add another logic from the MetaBot that you have selected.

a. Click **MetaBots** tab and select a Metabot from the list:



b. This launches the MetaBot window from which you can select a Logic:



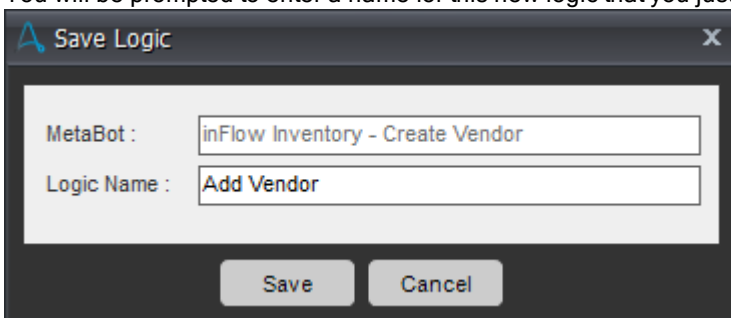
Note: You can select only those Logic that are available for the MetaBot from which you want to add. You cannot select Logics from other MetaBots. Those are available when you [Create a new Logic from the Workbench](#)

26. If needed, you can also add DLLs. [Learn More](#)

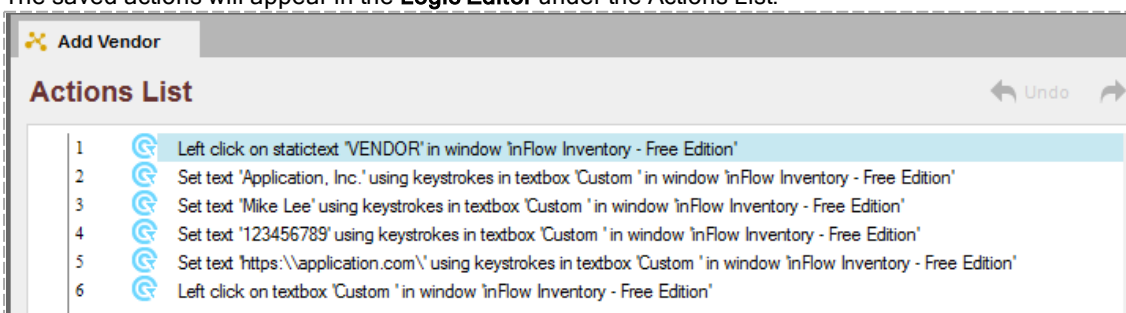
27. Once done, click **Save**.



28. You will be prompted to enter a name for this new logic that you just created. Click on **Save** to finish.



29. The saved actions will appear in the **Logic Editor** under the Actions List.




30. Let's run this logic to see if it works as expected. Click **Run**.

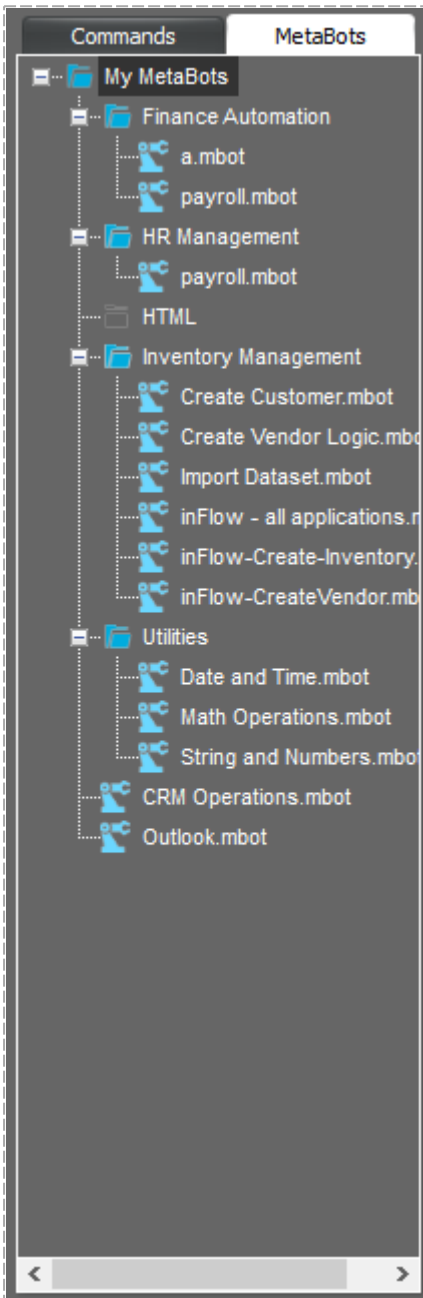


31. You have successfully created a MetaBot with a logic to create a Vendor in your inventory repository.

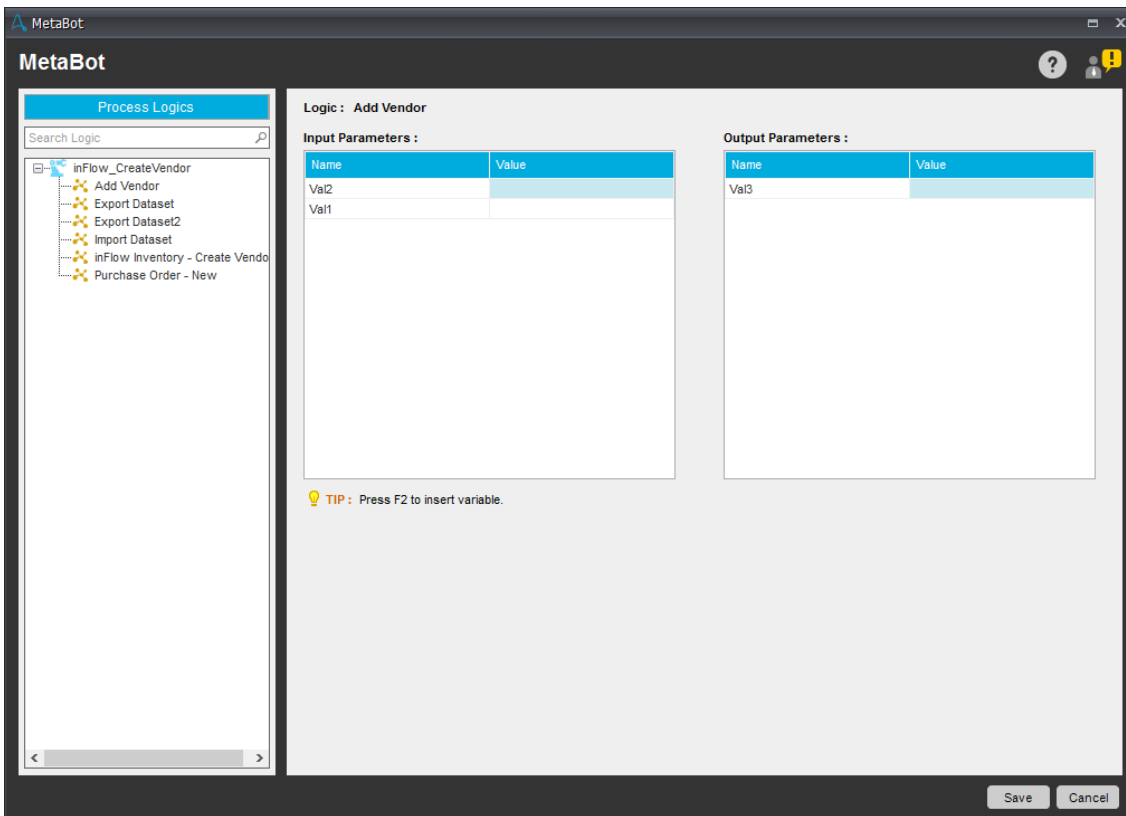
32. We will now use this Metabot in an automation task. Assuming Automation Anywhere Enterprise Client is also installed on your machine and supports MetaBots, open the Workbench to create a new automation task.

33. Select inFlow Inventory - Create Vendor from the list of MetaBots on the top left and drag it onto the action list.

 **Note:** These are available to other MetaBot users after the MetaBots have been **Uploaded** to the Control Room and later Downloaded to the Client.



34. This invokes the MetaBot UI where you can browse and select from the list of available **Logics**. Select the Add Vendor logic, add required **Input Parameters** as well as **Output Parameters**, and click **Save**.

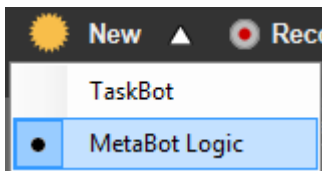


Note: You can also directly create a Logic inside a logical folder of the selected MetaBot in the Workbench.

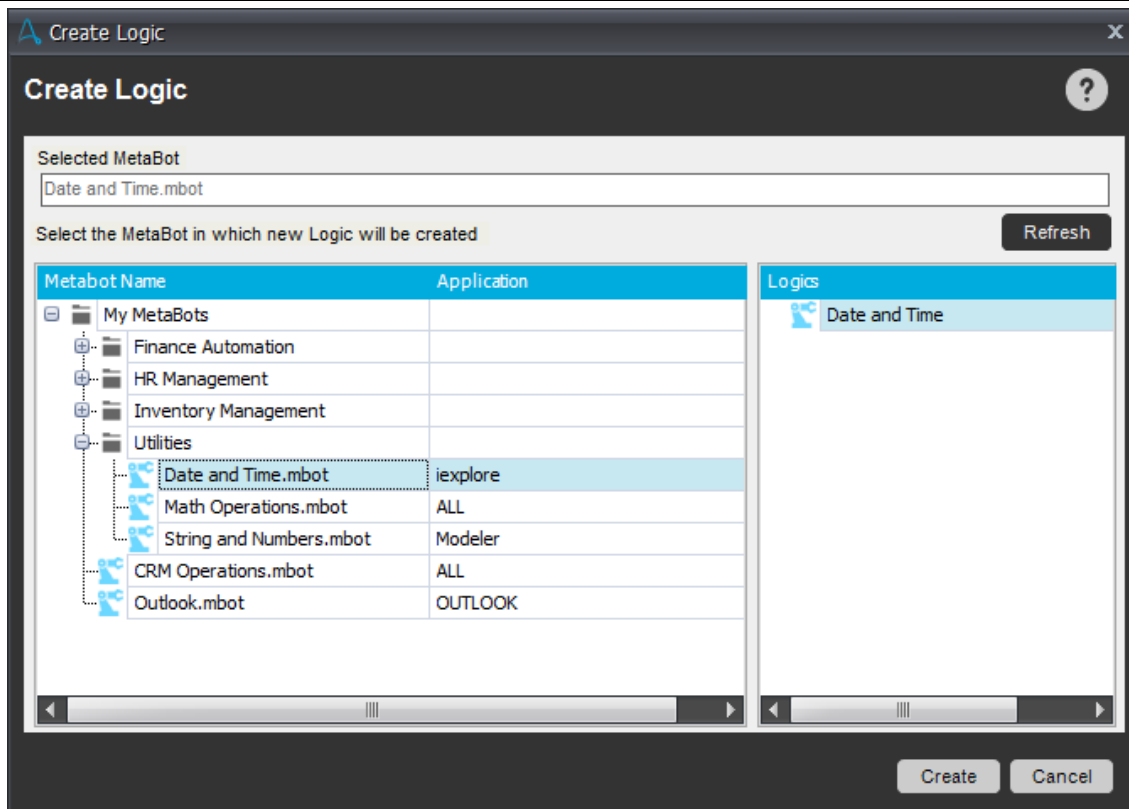
- To **create** a Logic, click **New**,



Note: To create a Logic by default, click on the arrow, select Metabot Logic and then click **New**:

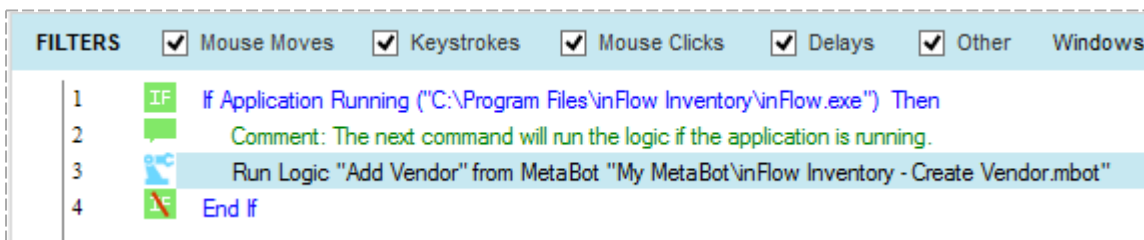


- This launches the **Create Logic** window. Select a MetaBot from the list of **My MetaBots**:



- The **Create Logic** window displays a list of **MetaBots**, the **Application** for which it is created, and list of **Logics** if available. Refer [Using the Workbench to create Logic](#) for details.

35. The logic gets added to your TaskBot.



Tip: If you specify any **Input/Output parameters** while creating a logic in the Logic editor, you can see them here. [Learn More](#)

36. Save the task and run.

Tip: You can launch the application first and then use the logic.

37. You have successfully used a MetaBot to create a repository of vendors in your inventory management system.

Adding and Recording a MetaBot

You can create MetaBots using the MetaBot Designer either by adding an empty MetaBot or recording a series of steps of a workflow.

Creating a New MetaBot


This creates an empty MetaBot for the selected application. You can then progressively add screens to this MetaBot at any point of time using [Add Screen](#) and [Record Screens](#).

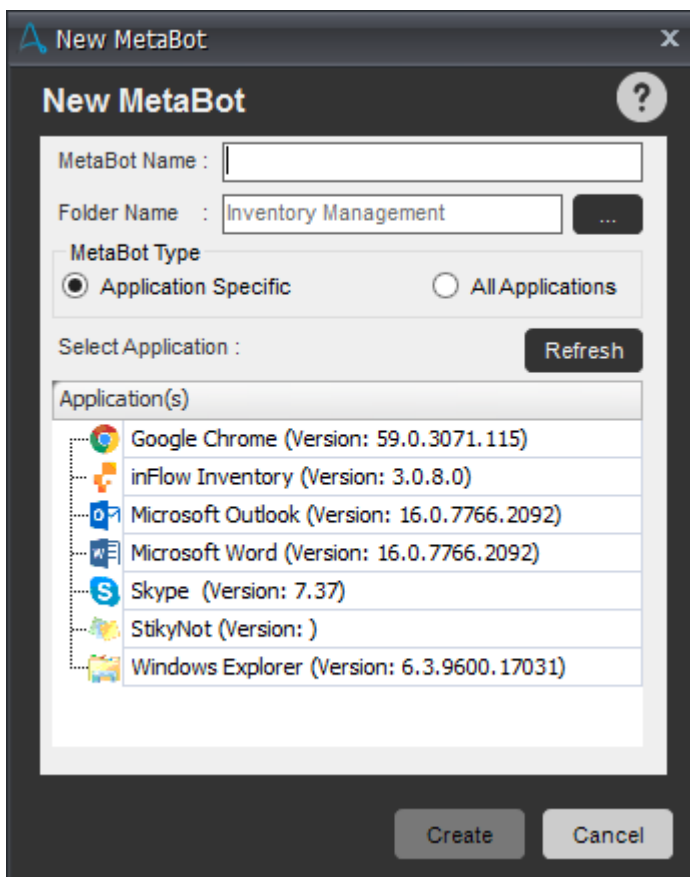
Creating a MetaBot using 'New'

1. Click New MetaBot

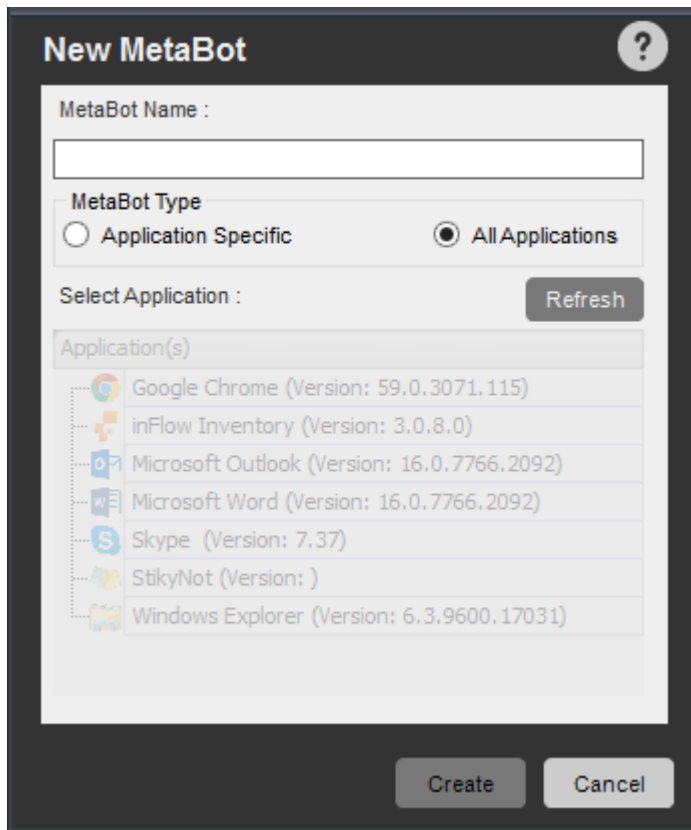


2. Select **Application Specific** to create a MetaBot for a single application. Else select **All Applications** to create an "application agnostic" MetaBot i.e. a MetaBot that makes use of multiple applications. Specify the name that reflects the purpose of creating the Metabot.

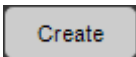
 Note: Click **Refresh** if your application is not listed. Alternately, despite 'Refresh' if it is not listed, verify if the application is running in 'Admin' mode. If yes, restart the MetaBot Designer in 'Admin' mode.



- When you select **All Applications**, the Application(s) list is disabled:



3. Click **Create** to save the MetaBot.



- The MetaBots are saved in the My MetaBots folder.

 Tip: Refer the detailed steps given in [Creating a MetaBot](#).

Creating a MetaBot using 'Record'

You can also create a **new** MetaBot by capturing the entire process in sequence.

Recording is especially useful when you want to capture a workflow. Here, you can opt to

- Record only screens
or
- Record screens with the workflow in the form of logic.

1. Record Screen(s)

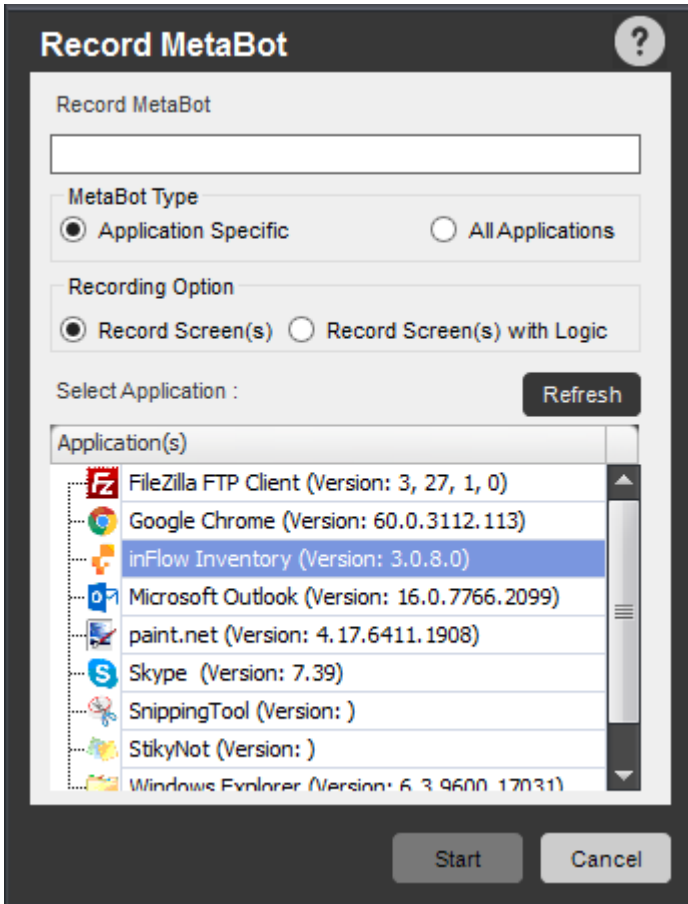
Use this option when you want to capture only screens with relevant object properties.

1. Click record

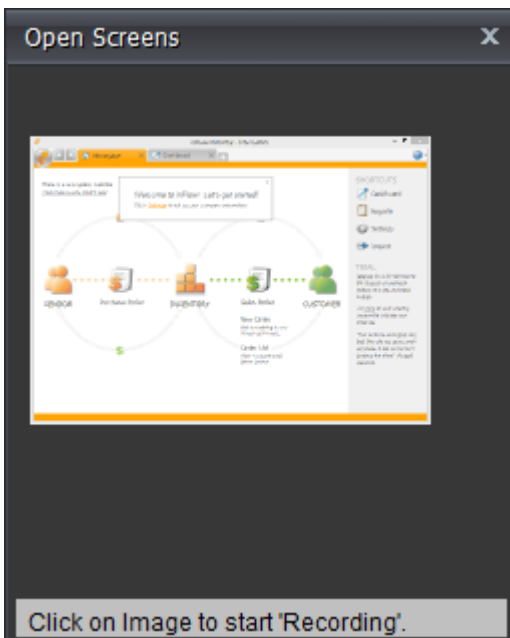


2. Specify either **Application Specific** or **All Applications** based on your workflow to capture GUI properties.

3. Select 'Record Screen(s)' option in the **Record MetaBot** window.



4. Click on the image in Open Screens.

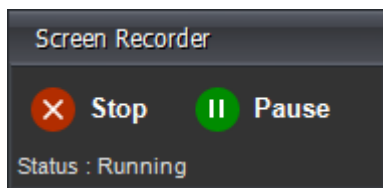


- MetaBot Designer begins recording the selected screen of the application:

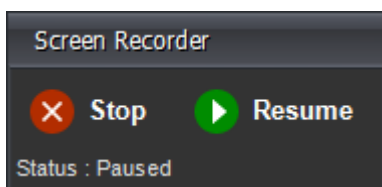


Note: Each click will record the screen.

- You know you are in recording mode when you see:



- Click **Pause** if you wish to stop recording for sometime but resume from where you left off.
- Click **Resume** to continue recording:

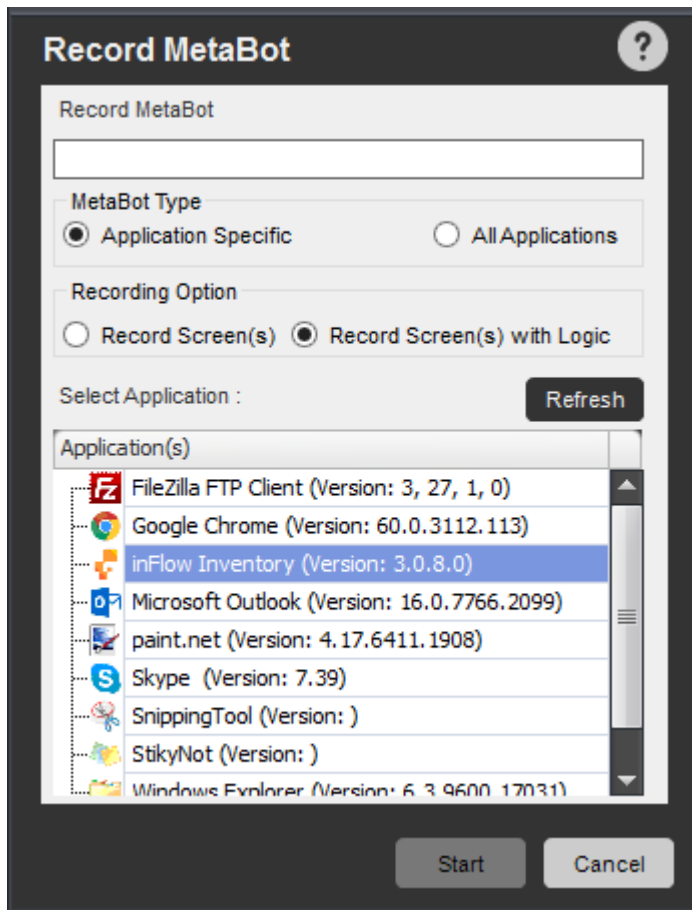


Note: If an application has multiple exe's, you are required to create a separate MetaBot for each.


2. Record Screen(s) with Logic

Use this option when you want to save the workflow in the form of Logic directly instead of configuring, calibrating and then designing a logic flow. This will also add recorded screens to the Asset library.

Follow Steps 1 through 6. However, instead of selecting 'Record Screen(s)', select '**Record Screen(s) with Logic**':



When you hit Stop, the Logic Editor is launched. Here, you can choose to edit the workflow and save as Logic.

 Tip: You can also record logic with screens for an existing MetaBot using the 'Record Logic' option in 'Logic' tab. [Learn More](#)

Adding Screens to a MetaBot using 'Add Screen'

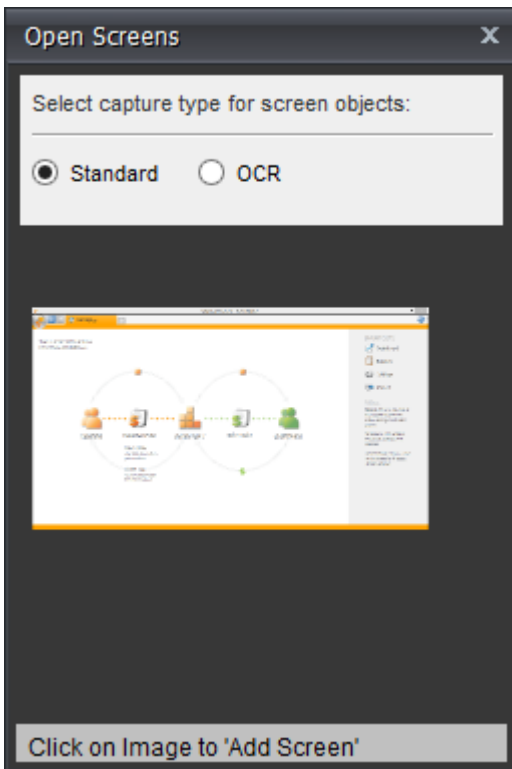
Use this to capture a single screen for an application that is running. This will add a screen to the existing MetaBot.


To add a Screen,

1. Click 'Add Screen'

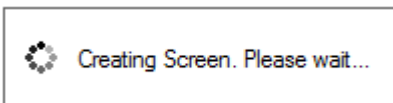


2. Select the capture type for screen objects - **Standard** or **OCR** and Click on the image in the **Open Screens** window. Refer details on screen capture type in [Creating a MetaBot](#)



 **Note:** OCR selection is available only for **Add Screen** option and not for **Record Screens**.

3. Wait for the Screen to capture:



4. The screen is added to the current MetaBot.

Recording Screens in a MetaBot using 'Record Screens'

Recording is especially useful when you want to capture a workflow.

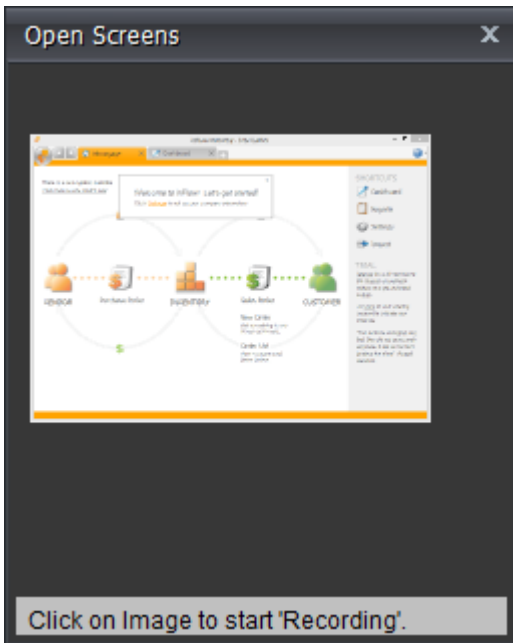
Use Record Screen to record all the Screens/UI elements (Menu item / Popup / context menu etc.) while you interact with the application in a workflow mode. These UI elements cannot be captured using Add Screen.

To record a Screen:


1. Click 'Record Screen'



2. Click on Open Screens window



3. The screens that are captured are saved to the current MetaBot

 **Tip:** It is recommended that you keep the zoom level of your screens to 100% while creating, recording and playing your MetaBots.

Updating MetaBots

Update your MetaBots by adding to or modifying the existing one. You can add/record screens, add dll's, folders and re-upload to Control Room. You can re-configure and re-calibrate captured screens.

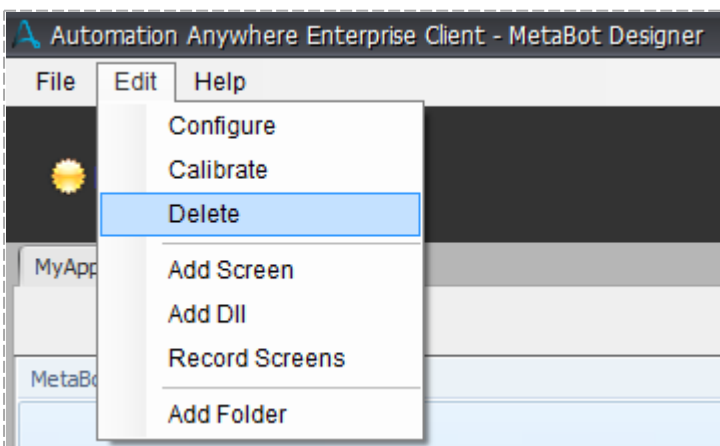
i.e. carry out all functions that you would when creating a new MetaBot.

Deleting MetaBots

You can delete a MetaBot or Screens and/or Dlls from a MetaBot.

Delete in one of the following ways:

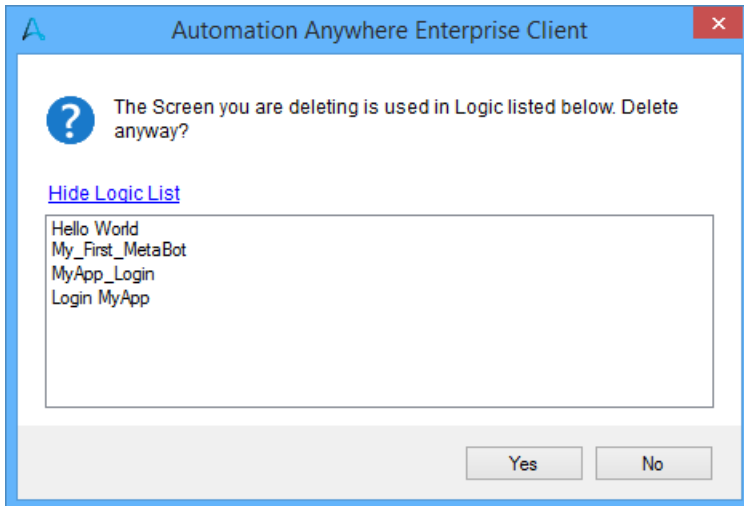
1. From the 'Edit' menu:



2. Using 'Delete' button:



Note: You can verify whether the 'Screen' to be deleted is used in multiple logic from a 'Logic List':



Configuring MetaBot Screens

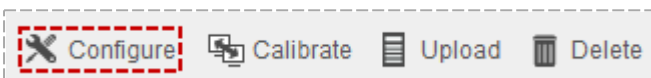
Essentially, you need to create MetaBots in such a manner that they cover all possible run time scenarios. This will ensure that the tasks in which they participate run without any glitches. Thus, simply capturing a screen may not help in most cases. This is where the Configure feature comes into picture.

Use 'Configure' to edit object properties for the recorded/added screen. Here you can provide aliases such as a 'Screen Name' and a 'Screen Title'. You can also select an object to define its properties such as Name, Path, Value, ID, Class, Index, States etc and the 'Play Mode' to be used when running tasks.

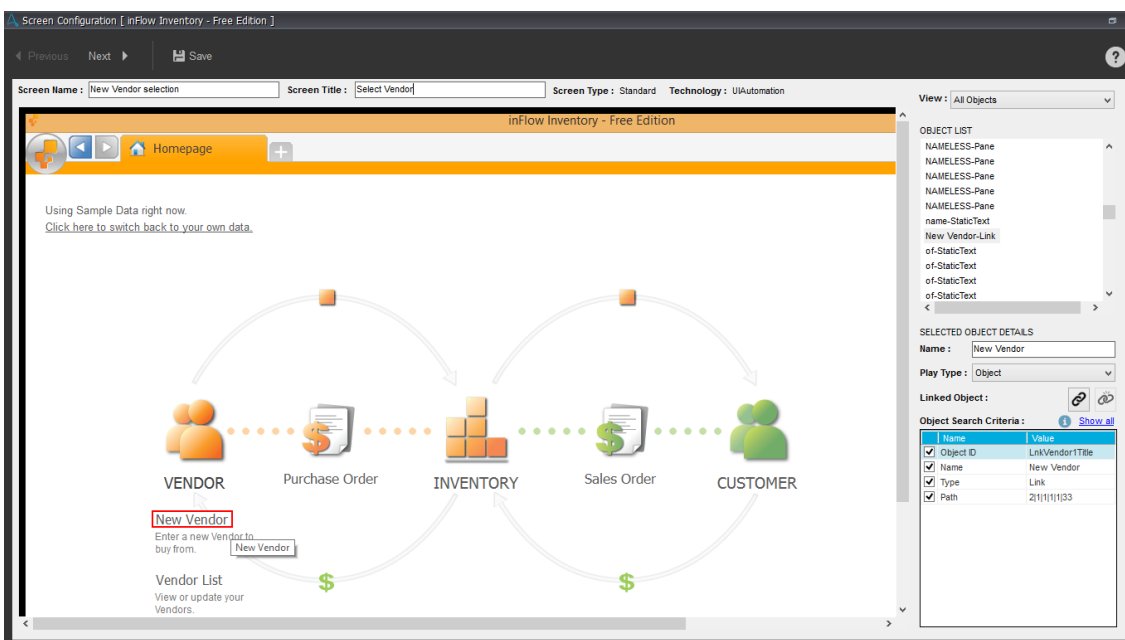
The object properties have a very important objective. A set of these properties help to uniquely identify the associated object during playback. As you would have noticed, the product intelligently selects some of these properties by default for every selected object to get you going through most of the scenarios. However, for complex cases, you can always play with these properties to make your MetaBots truly reliable. For instance, you can customize certain objects using the '[Custom Object](#)' option.

1. Configure a Screen


1. Select a screen.
2. Click Configure




3. This launches the **Screen Configuration** editor which has options to go to **Previous** and **Next** screen, select a screen **View** and **Object Details**. It also shows the **Screen Name**, **Screen Title**, **Screen Type** and **Technology / OCR Engine** used to capture the screen.



4. Highlight the control which you wish to use during run-time.
 - The object properties are populated in the right panel of the screen.
5. Optionally, provide a **Screen Name**.

 **Note:** **Screen Name** is the one that we provide to the added screen, while **Screen Title** is the one that appears on the screen window that is configured.

- You can rename the Screen if you feel that it would be easier for a user to identify with something else.
- Also, it is recommended that you provide a generic screen title to ensure compatibility with all situations

 **Note:** Use ' * ' to add a generic title.

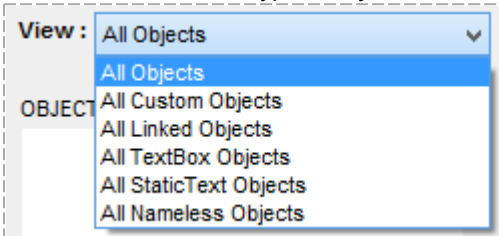
6. **Screen Type** allows you to figure out the object capture type that was used - **Standard** or **OCR** while adding the selected screen.

7. Depending upon the object capture type, the system displays **Technology** or **OCR Engine**. For Standard captures, **Technology** is shown as **UIAutomation**. For OCR automation, the system displays the OCR engine used to capture the object. For details, refer the section on adding screens in [Creating a MetaBot](#).

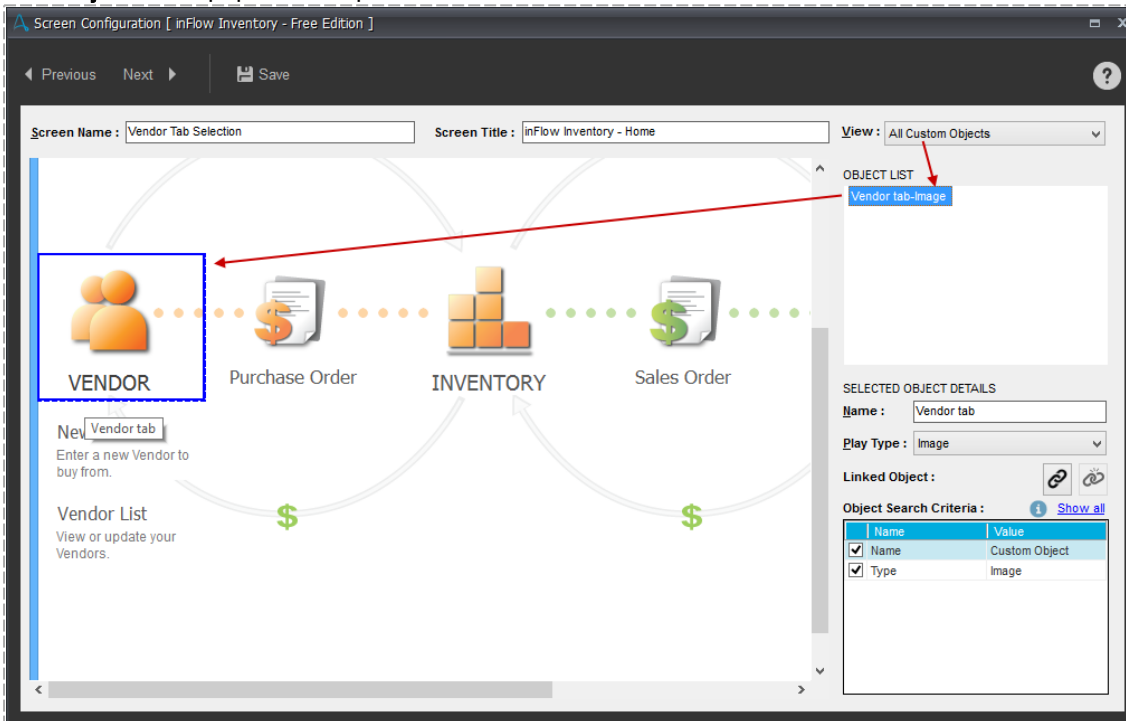
2. Configure Object Properties

You can configure the object properties using **View**, **Selected Object Details**, and **Object Search Criteria**. Here, you can rename the objects, select a play type, manually link one object with another and define the search criteria that will be used for executing the automation.

1. To customize the **View** type of object that has been captured, select any one from the list:



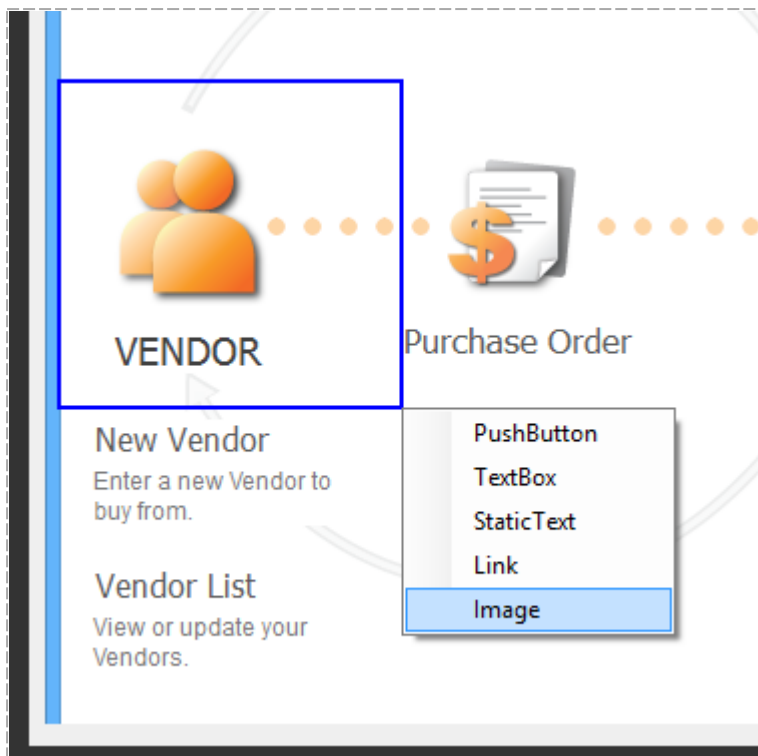
2. The **Object List** is populated as per the View selected.



- **Custom Objects** - These are the ones that you define especially when the object properties are not captured while recording/capturing a screen.

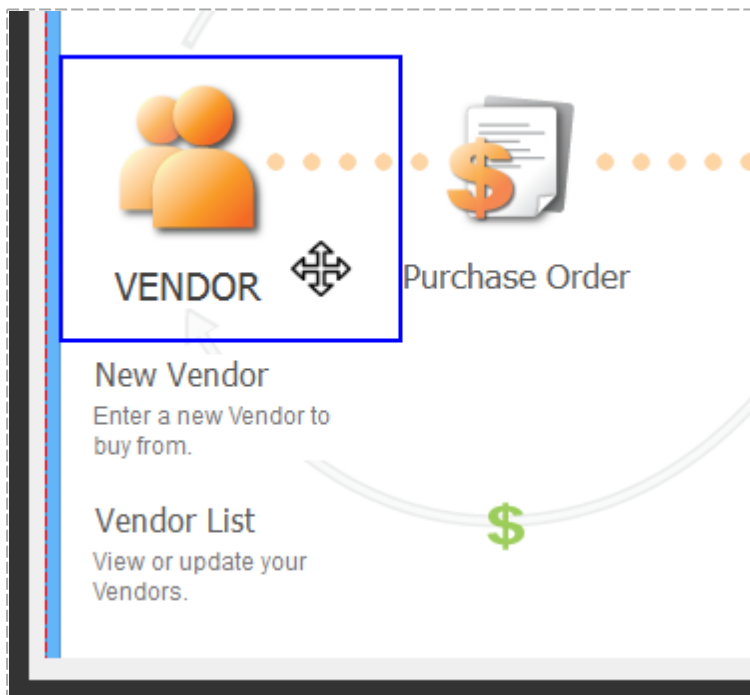
To classify an object as custom, simply drag and select the area; choose whether to treat the object as Push Button, Text Box, Static Text, Link or Image.

Note: All custom objects are highlighted with a blue outline when you **select** an object whereas the non-custom objects are highlighted in red outline. Similarly, when you **mouse-over** the objects custom objects are highlighted with blue-dotted line while non-custom objects are highlighted with red-dotted line.

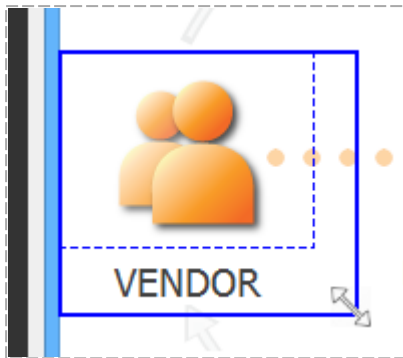


When custom objects move or are re-sized, you will have to update relevant Screen(s) to reflect the same. You can achieve this by moving and/or re-sizing those during configuration.

- a. To move the customized object to a different location, click and drag it to the target location.





- b. To re-size a customized object, hold down the click button and drag the pointer as required.

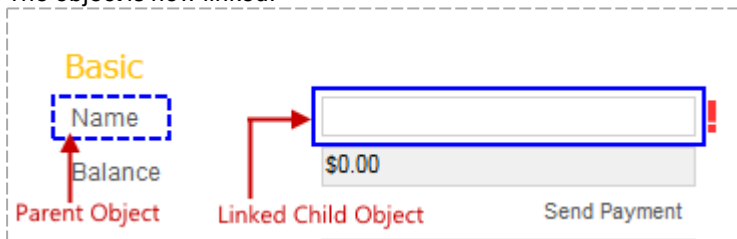


- **Linked Objects** - These are the ones which are linked to a particular object in the screen. Objects are linked to other objects or UI elements that can be searched easily during automation execution.


For objects that are captured using **OCR**, the linking is done automatically by the system and hence you do not need to link those objects manually. However, for screens that are captured using Standard technology, you need to link objects manually. [Learn More](#)


 Note: If you have html elements in your screen and you want to link those, it is recommended that you configure them as custom objects and then use linking.

- To link an object with another, select the easily identifiable object in the screen and click the  icon in **Linked Objects**. The easily identifiable object is the Parent object while the one that is linked to it is the Child object.
- Move your cursor to the object that you wish to link and click the link
- The object is now linked:

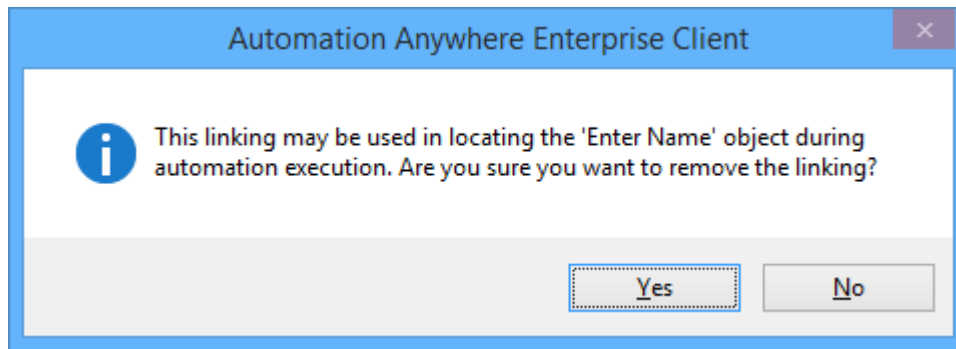


- You can link multiple child objects to a single parent object.
- You can also change the linking between parent and child objects. When you do so, the Bot Creator who edits the automation that uses these objects is shown a message. [Learn More](#)

 Tip: The properties of the parent and child objects can be seen separately in the properties window when you create a Logic. When you execute the automation, the child objects are searched on the play type of the parent object. [Learn More](#)

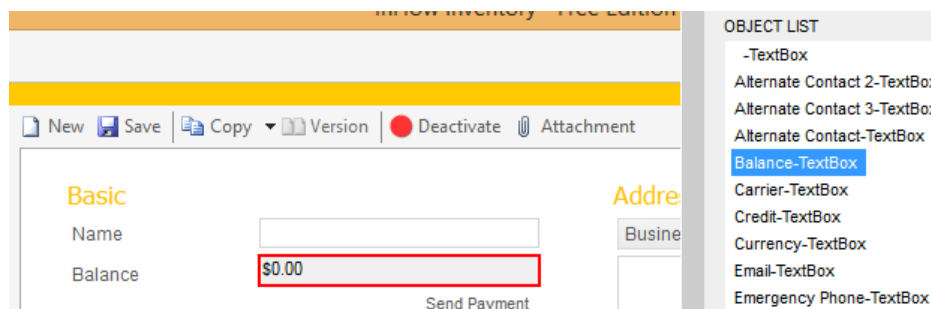
- You can also de-link the objects that were linked. Click on the  icon in **Linked Objects**.

Be aware that while de-linking objects you must ensure that they are not used in your automations.

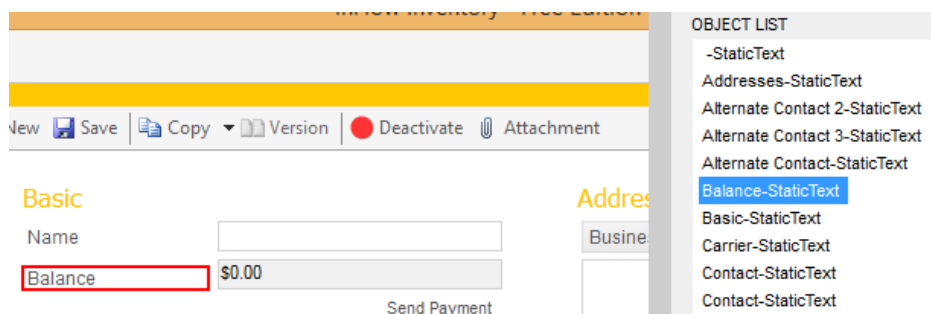


If the objects are already being used in automation and you click **Yes**, messages are shown when those automation are opened in edit mode by the Bot Creator. [Learn More](#)

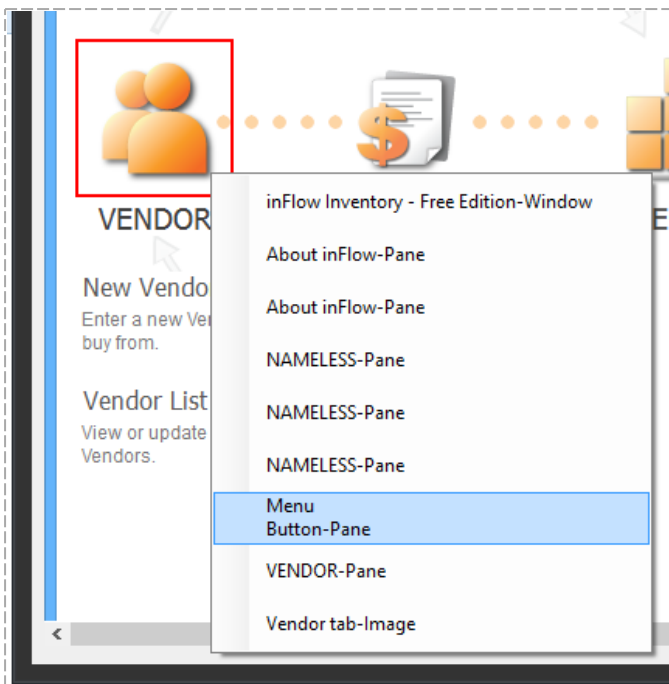
- **TextBox Objects** - These are the objects that appear as text-boxes in the captured screen.



- **StaticText Objects** - These are the ones that appear as text in the captured screen.



- **Co-located Objects** - These are the ones that are neighboring to the currently selected object belonging to the same parent control.



Note: These objects are not visible in the **View** filter.

- **Nameless Objects** - These are the ones which have not been categorized in any of the other object types. These objects are the ones that appear on the screen, but are not configured. These have unique system generated names as it enables easy identification during **Screen** and **Import/Export Command** configuration.

Important: From Enterprise Client Metabots version 11.0 onward, these Nameless objects are defined as TextBox or StaticText based on the object type.

The Nameless objects captured before an upgrade to the 11.0 version continue to exist as Nameless. However, it is recommended that you assign names to those for easier identification in Import or Export automation.

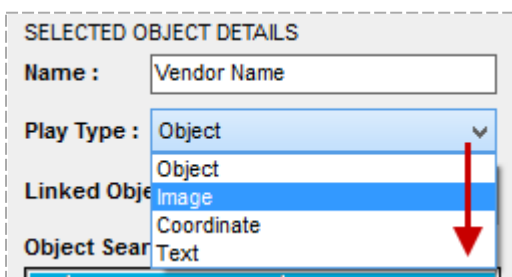
3. Customize Selected Object Details


After configuring object properties, you can choose to customize the object details per your automation requirement. In the **Selected Object Details** panel you can provide **Name** of the object, select a **Play Type**, and select the **Object Properties** to support the object Search Type.

1. In the **Selected Object Details** panel, you can optionally specify an alias in the Name text-box for the same reason as renaming Screen/Title names.


Note: Here, 'Name' is the selected object name. Specify user friendly / easily identifiable name for the object. This is helpful when you use this screen in a task for automation.

2. Select a **Play Type** - whether Object, Image, Coordinate or Text.



 Note: Observe that the play type is enabled automatically depending upon the selected object type. You may however select the one that suits the purpose of the Logic.

- **Object** - Use this as the play type for object selected on the basis of its object properties. This could be useful when the selected objects are dynamic in nature; in the sense that they keep shifting their positions in the target application. This is considered the most reliable form of automation as it is performed on the UI based elements of the object.


 Tip: If object based automation execution is not possible, then you can opt for the other play modes - Image, Coordinate, or Text.

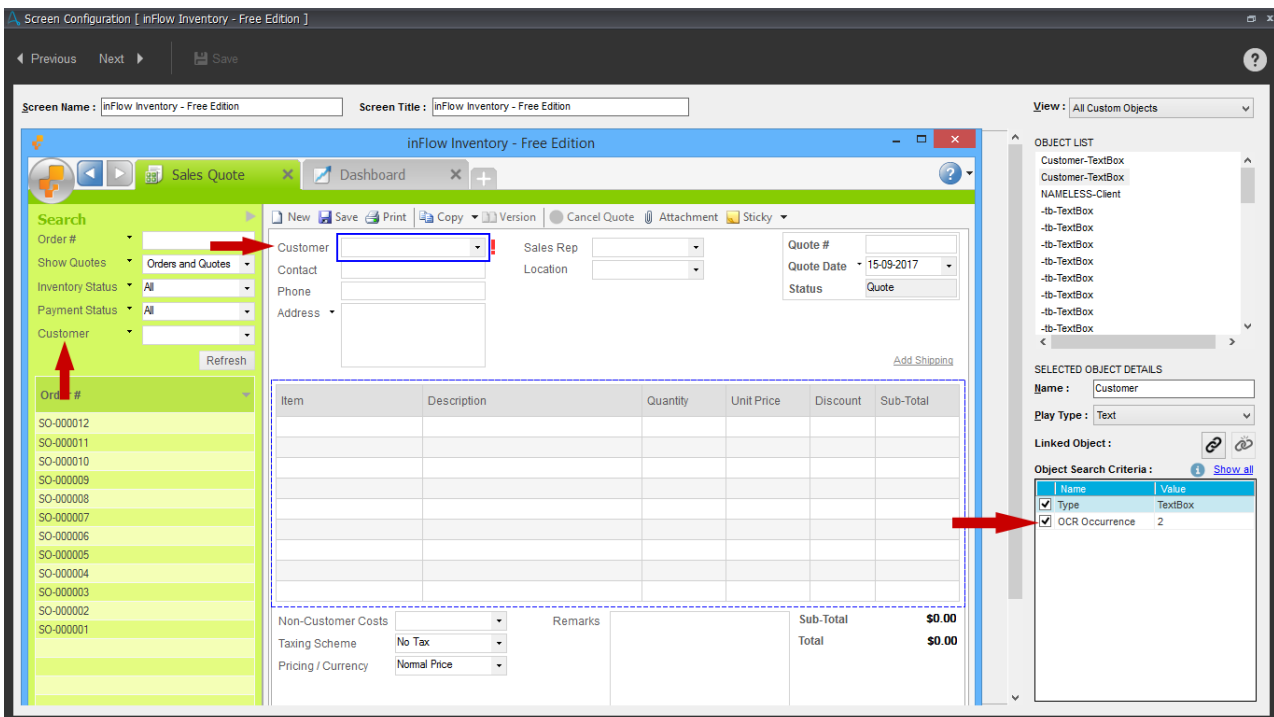
- **Image** - Use this as the play type for objects selected on the basis of its image properties. This could be useful when object based automation is not feasible or fails for certain reasons. For example, image based automation works for Citrix, RDP, and Legacy applications such as delphi. You can also use this option when you want to run automation to extract text from screens that were captured using OCR. [Learn More](#)
- **Coordinate** - Use this as the play type for objects selected on the basis of its coordinates properties. This could be useful when selected object is available at the same co-ordinates in the target application.
- **Text** - Use this as the play type for objects selected on the basis of its text properties. This could be useful when you want to extract text from an image based object in the target application. Since the text is extracted using OCR engine while configuring screens, only that OCR engine is used to execute the automations. Hence, the selected OCR engine should also be installed on the Bot Runner machines to ensure your automations do not fail.

3. Select **Properties** type that you wish to use during play time from **Object Search Criteria**. You can choose to view select properties or view them all. Some properties are selected by default.

Note that for objects that are not linked manually to other objects, its own search properties are used whereas for objects that are linked to other objects using the **Link** option, the selected object will be searched based on search properties of the linked object during automation execution.

You can also change the **Values** of the properties, if required to make your MetaBots more reliable.

 Note: Properties could differ depending upon the type of object/control captured. For example, for objects captured using OCR, the object properties show the number of times an object appears in the screen:



The screenshot displays the 'Screen Configuration' window for 'inFlow Inventory - Free Edition'. The main window shows a search interface with a 'Customer' dropdown menu highlighted by a red arrow. On the right, the 'OBJECT LIST' shows various object types, and the 'SELECTED OBJECT DETAILS' panel shows the 'Customer' object with a 'Play Type' of 'Text'. Below this, the 'Object Search Criteria' table is visible, with a red arrow pointing to the 'OCR Occurrence' property, which is checked and has a value of '2'.

| Name | Value |
|--|---------|
| <input checked="" type="checkbox"/> Type | TextBox |
| <input checked="" type="checkbox"/> OCR Occurrence | 2 |

4. Save when you have all screens covered for configuration.

 Save

Tip: Use the 'Previous' and 'Next' buttons to access other screens in the MetaBot without exiting Application Configuration.

 and 

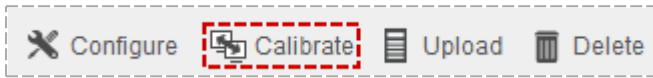
Calibrating MetaBot Screens

An application can undergo continual change during its life-cycle with improvements and newer features. MetaBot Designer's Calibration feature allows you instantly compare an existing screen with newer screen to identify those changes.

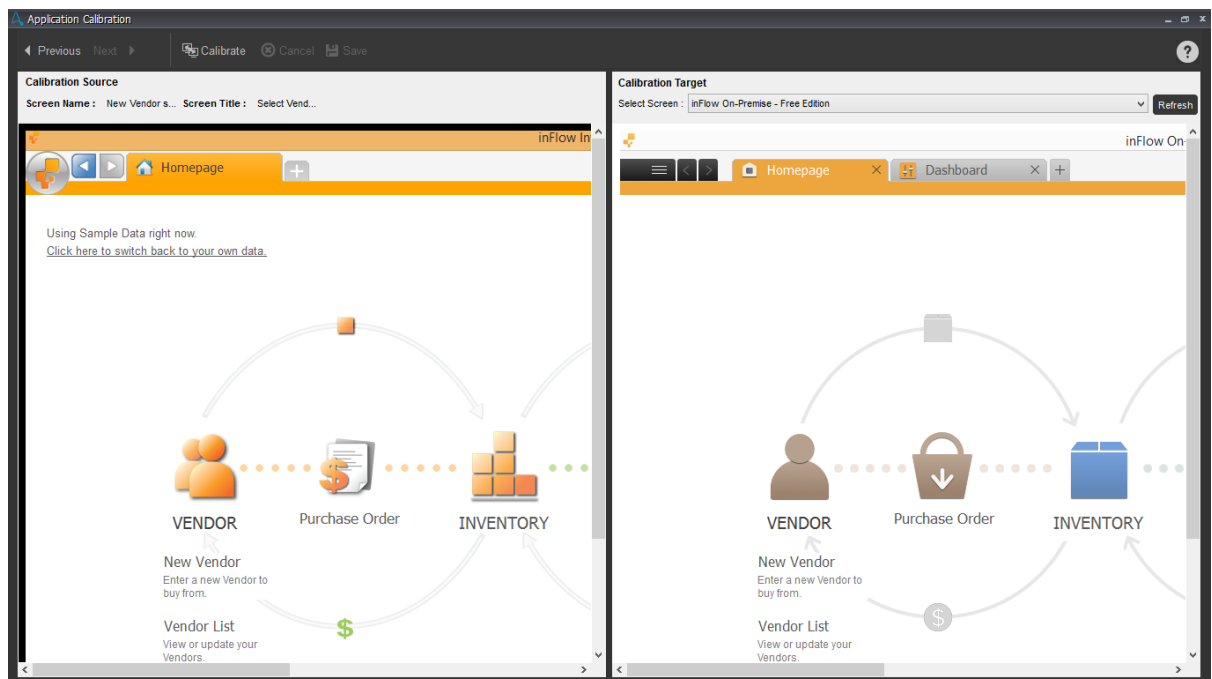
With a single click you can then upgrade the existing screen and upload it so that all the automation tasks using that screen can leverage the newer features.

Guide to Calibrating a Screen

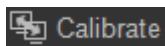
1. Select a screen.
2. Click **Calibrate** to launch the **Application Calibration** editor



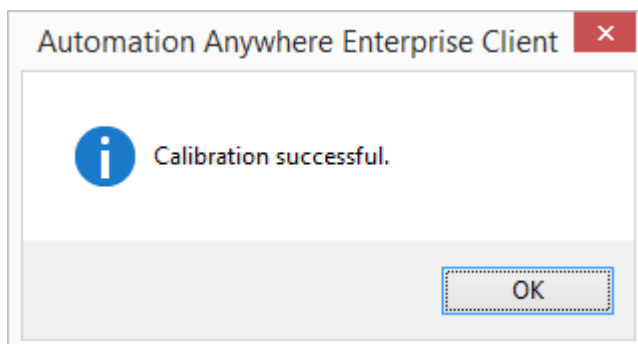
- The calibration screen is divided into two panels: The one on the left - **Calibration Source** indicates your current screen (which needs to be calibrated) and one on the right **Calibration Target** (usually the latest screen) .



3. In the **Calibration Source** panel, the screen that has been selected from the Metabot is displayed.
4. In the **Calibration Target** panel the latest screen that will be used to calibrate is displayed.
5. Click **Calibrate**.

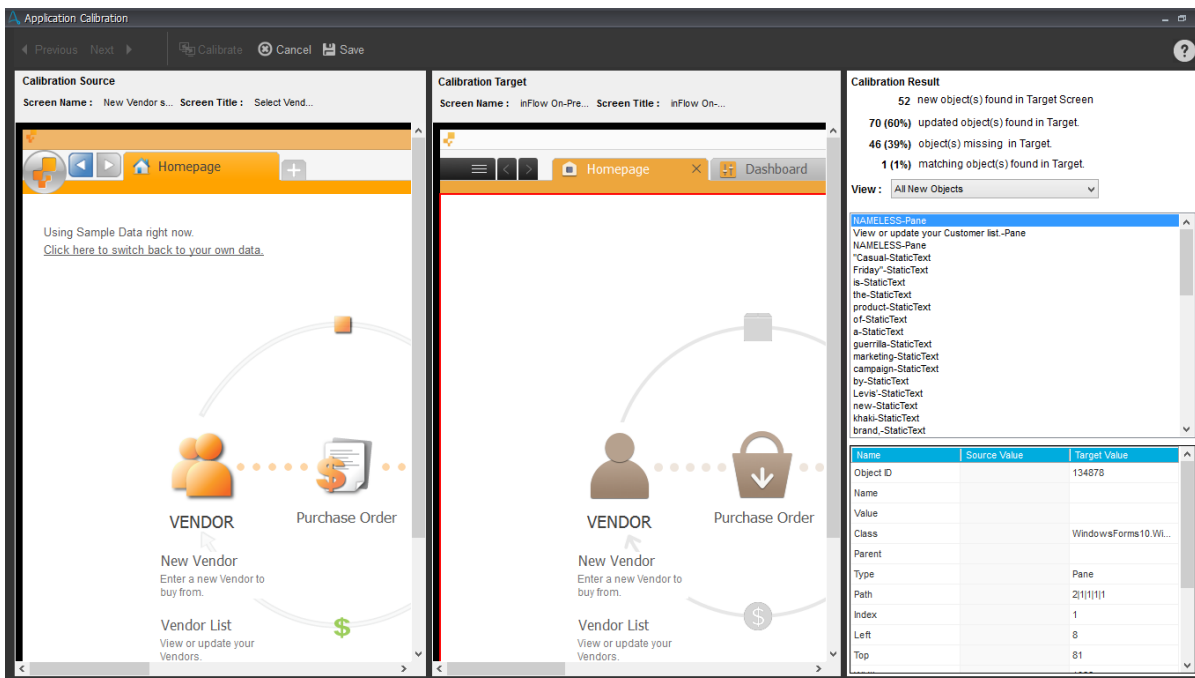


- You know the Calibration is successful when the following message is displayed:

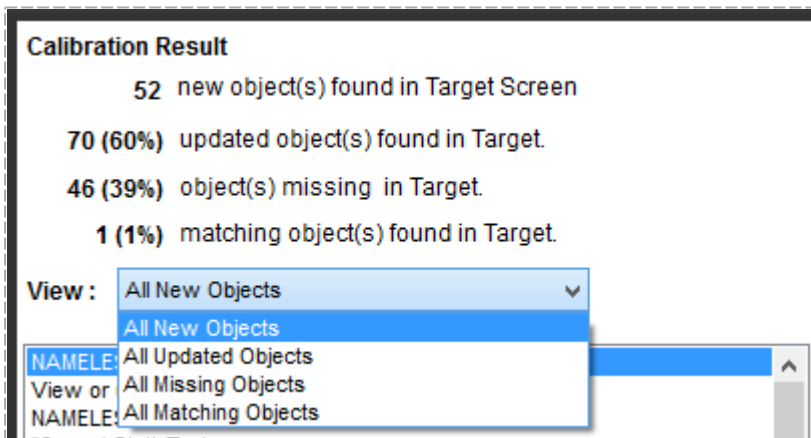



 Note: You cannot calibrate screen that pop-up or do not have a title.

- The properties displayed under the screens are calibrated accordingly and displayed in the 'Calibration Result' panel:



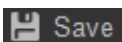
Here you can choose to view calibration in four 'Views':



 Note: MetaBot Designer provides you quick summary (in numbers and percentages) of the newly added objects, updated ones, missing from targeted screen and matching objects that were found.

- Once you complete the calibration, you can actually compare the screens by clicking on the objects on either panes. The objects will be highlighted accordingly. Any unchanged object will get highlighted in both panes. And if it doesn't get highlighted, it indicates a change between your current screen and the calibration source.

5. Save when done.




- 6. Next, you can opt to retain or overwrite existing screen if a newer one is detected. You can also choose to carry forward any existing custom object, if configured.

Automation Anywhere Enterprise Client ✕

This will replace your existing screen (Source) with the newer screen (Target).
Do you want to continue?

Carry forward all associated custom objects to the newer screen.

Yes No

 Tip: Use the 'Previous' and 'Next' buttons to access other screens in the MetaBot without exiting Application Calibration.


◀ Previous and Next ▶

Adding Folders to a MetaBot

You can bundle similar screens and DLLs together using folders. This will help you to logically manage MetaBots for ease of understanding and use.

i.e. Keep all the Screens/DLLs related to new employee record creation together in a folder "New Employee".

MetaBots can have any number of folders. You can also perform certain operations on a folder i.e. configure, calibrate, upload and delete.

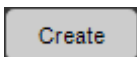
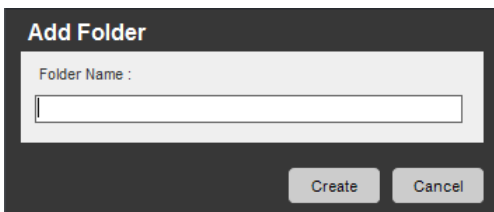
 Note: You can add 'Folders' to 'Assets' and 'Logic'.

Guide to Creating a Folder

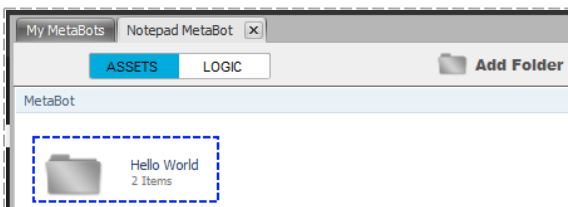
1. In the MetaBot Designer, click on Add folder





2. In the 'Add folder' window specify the folder name. Ensure that you use a name that reflects the type of folder created. Save by clicking 'Create'



- This is how a folder is displayed in the work-space window:



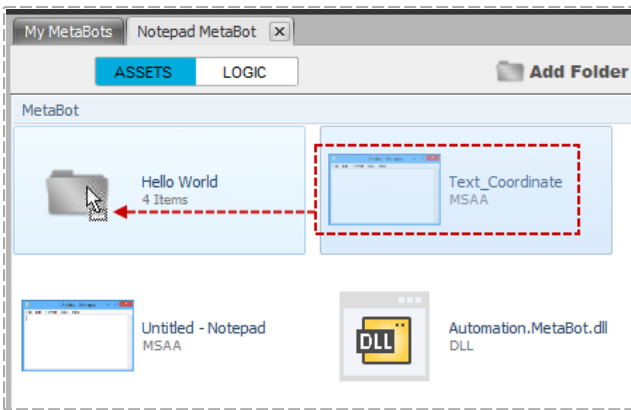
- Double click the folder to access related screens and dlls.

 Note: Use the back button () if you wish to return to the main/previous window.

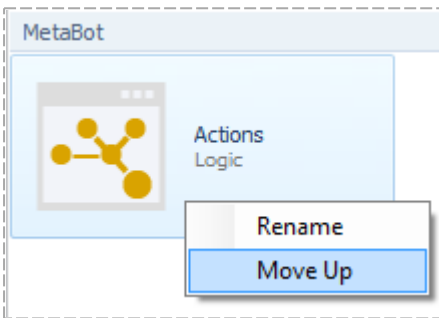
Moving Assets and Logic within Folders

You can move your Assets (Screens & DLLs) and Logic within Folders of a MetaBot.

- To move an Asset (even Screen or DLL individually) and/or Logic, simply drag and drop it to the target folder.



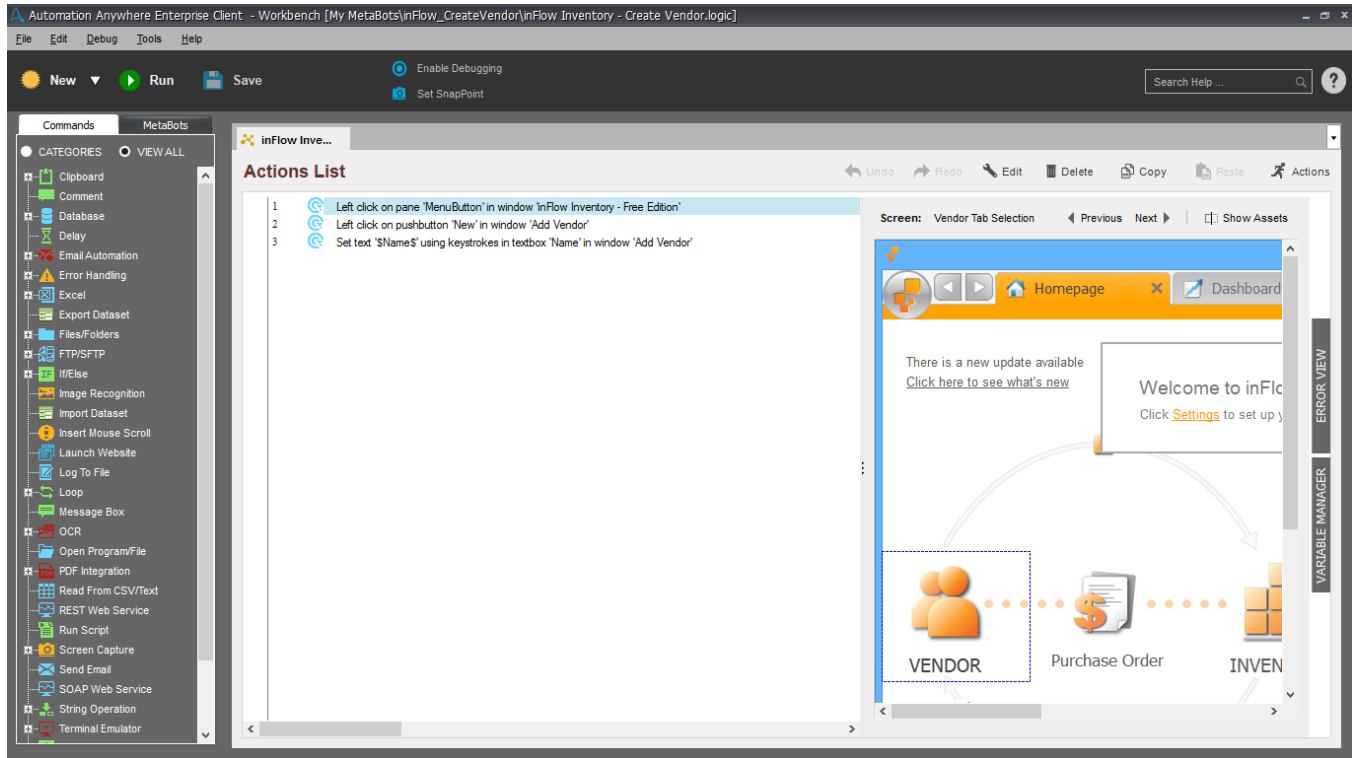
- To move an Asset (even Screen or DLL individually) and/or Logic, a level up, select the option 'Move Up' in the context menu:



Remember - to move to a folder you need 'drag and drop' action; but to move a level up you require to use the context menu.

Using the Workbench to create Logic

Use the Automation Anywhere **Workbench** to create simple manageable independent navigational flow - Logic that can be integrated into other automation TaskBots / MetaBot Logics as and when required. Logic is a pre-configured use case of an application that leverages **Assets** (Screens and DIIs).



The Workbench

Use the following components to create a Logic.

1. **New** - Use this to create a new MetaBot Logic



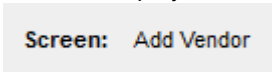
2. **Run** - Use this to run the Logic in edit mode of the Workbench



3. **Save** - Save a new or an edited Logic



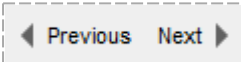
4. **Screen** - Displays name of the selected screen provided during screen configuration



5. **Captured Object** - This space displays the captured Assets (Screen or DII).



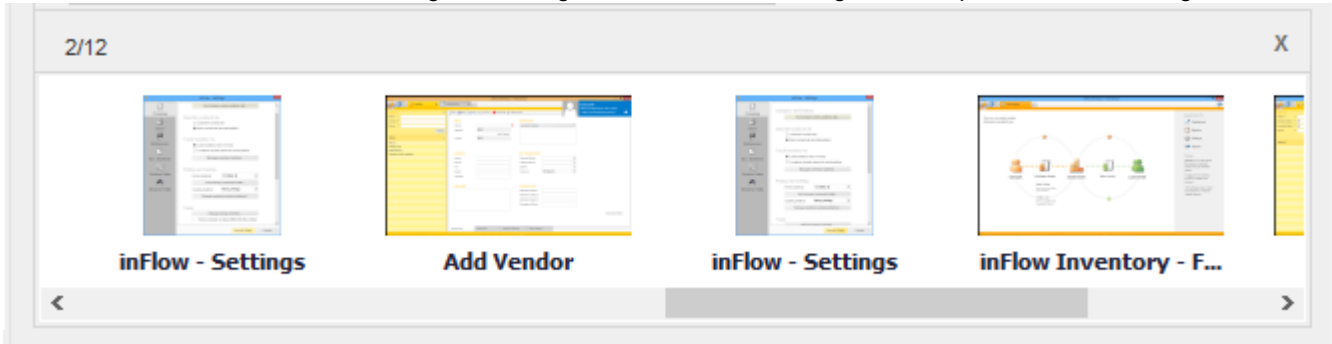
6. **Previous/Next** - Use this to navigate between captured Assets (Screens and DIIs).



7. **Show/Hide Assets** - A toggle control used to display or hide captured Assets (Screens and DIIs).



8. **Assets carousel** - Displays captured/recorded screens and DIIs in the chronology they were added. It invokes on clicking 'Show Assets'. Offers a visual means to viewing and adding Assets. Use this to configure the required Asset to the logic.

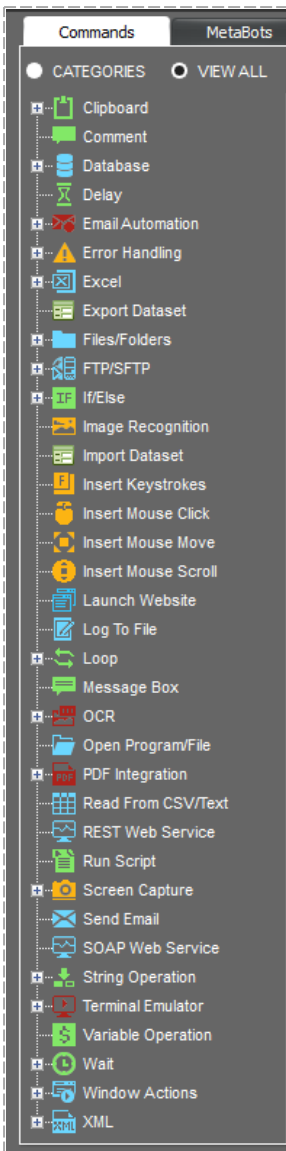


9. **Edit button** - Use to edit an existing Command or Action.

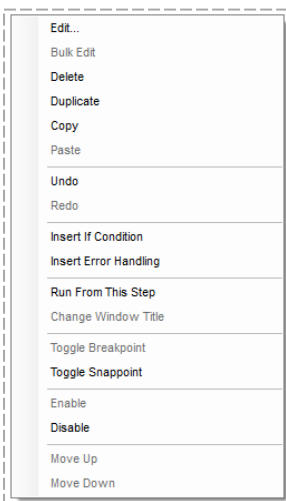


Note: 'Run', 'Save' and 'Edit' are enabled when an Action/Command is configured.

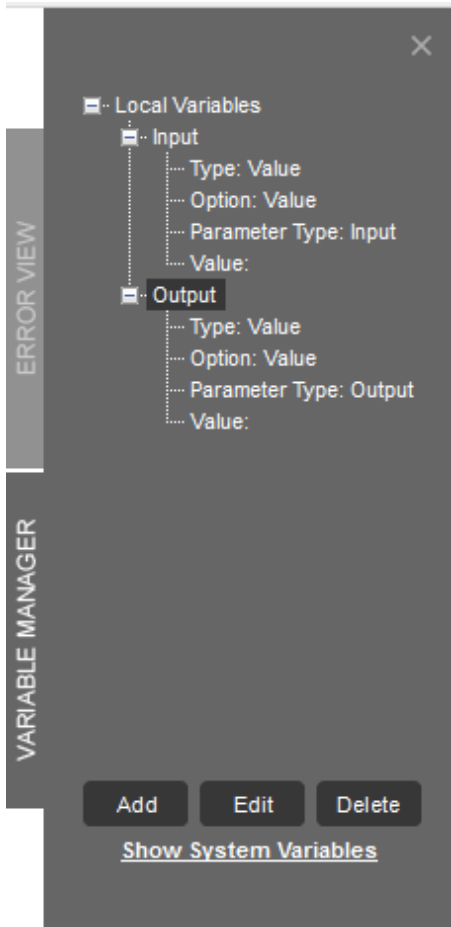
10. **Commands tab** - Use to add Commands to the Logic.



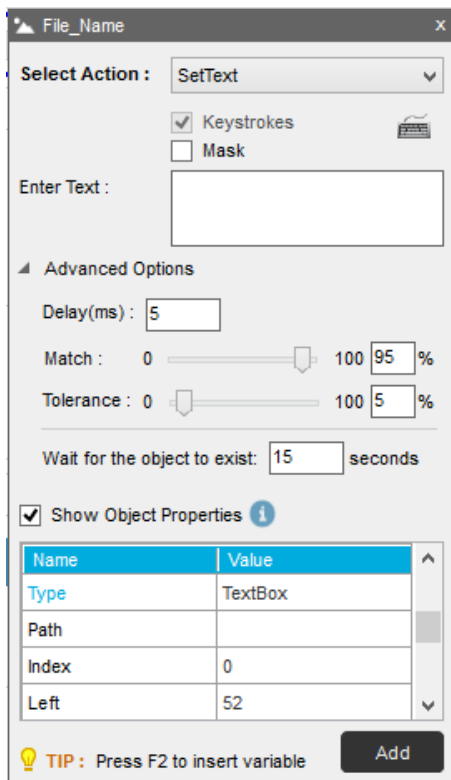
11. **Context menu** - Use to perform any of the actions given in the list:



12. **Variable Manager/Error View** - Use the 'Variable Manager' to manage variables and the 'Error View' for viewing and fixing task errors.



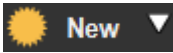
13. **Properties Window** - Use the floating 'Properties Window' to add and/or update actions.



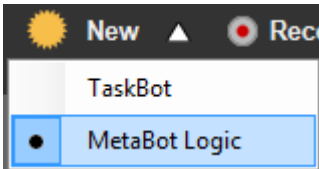
Creating a Logic

A Logic is a combination of 'Commands' and 'Actions'.

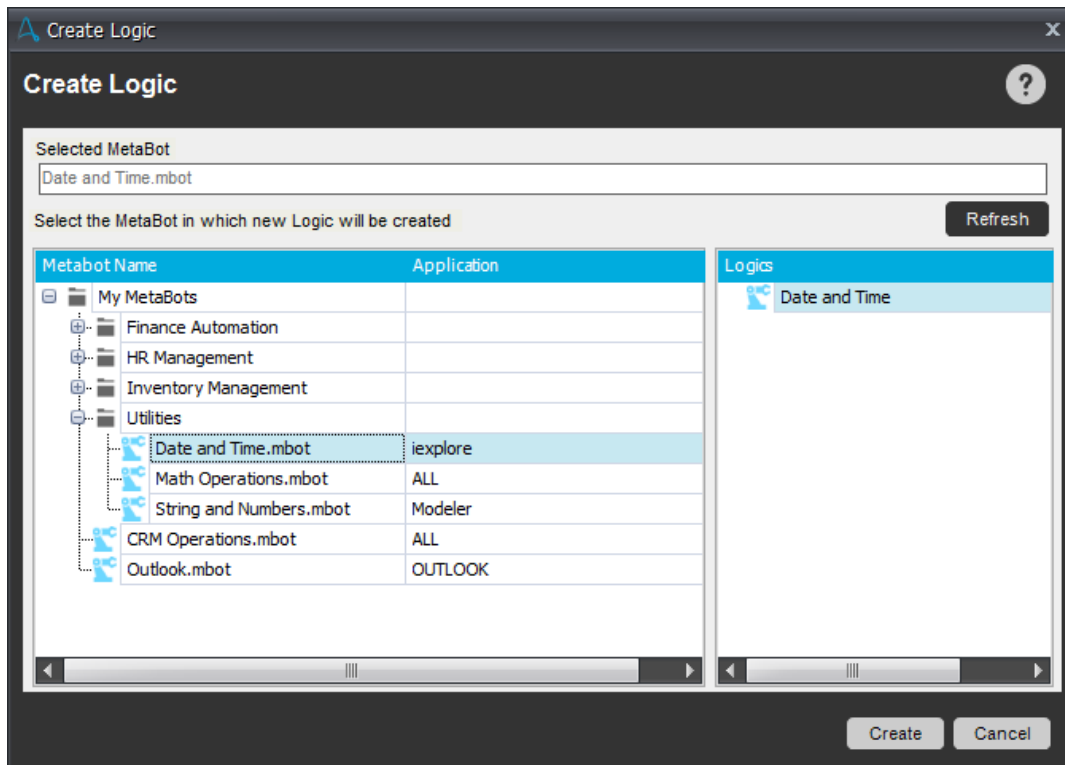
1. To create a Logic, click New



Note: To create a Logic by default, click on the arrow, select Metabot Logic and then click **New**:

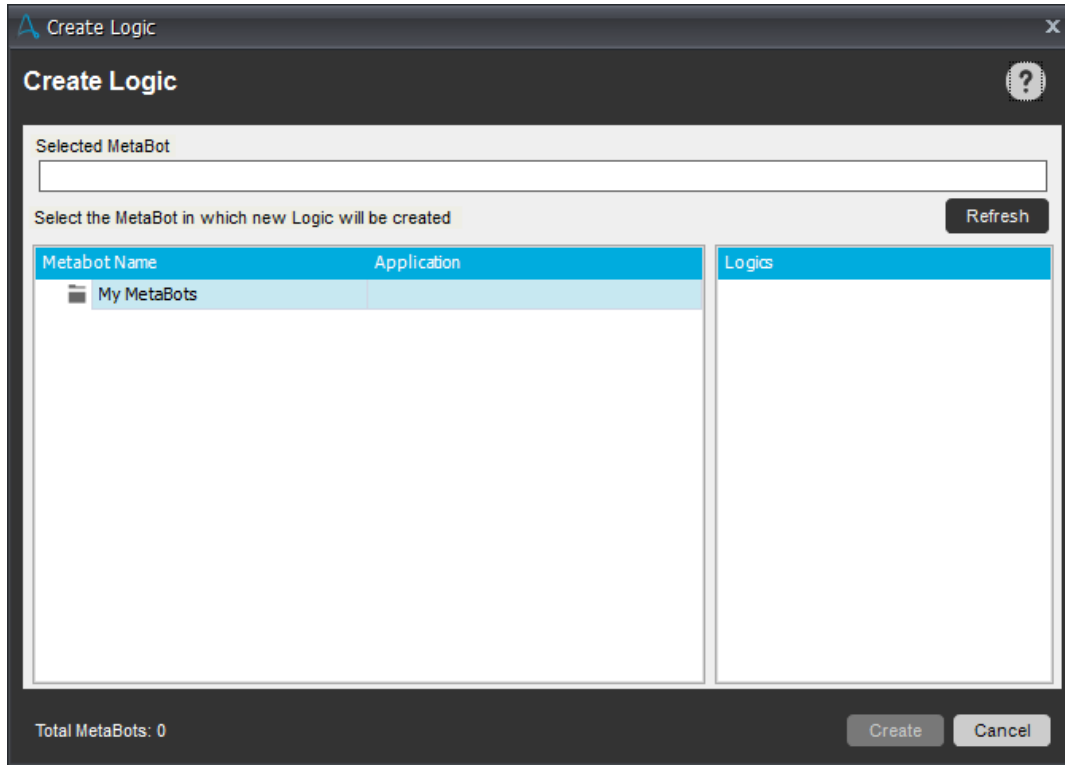


- This launches the **Create Logic** window. Select a MetaBot from the list of **My MetaBots**:

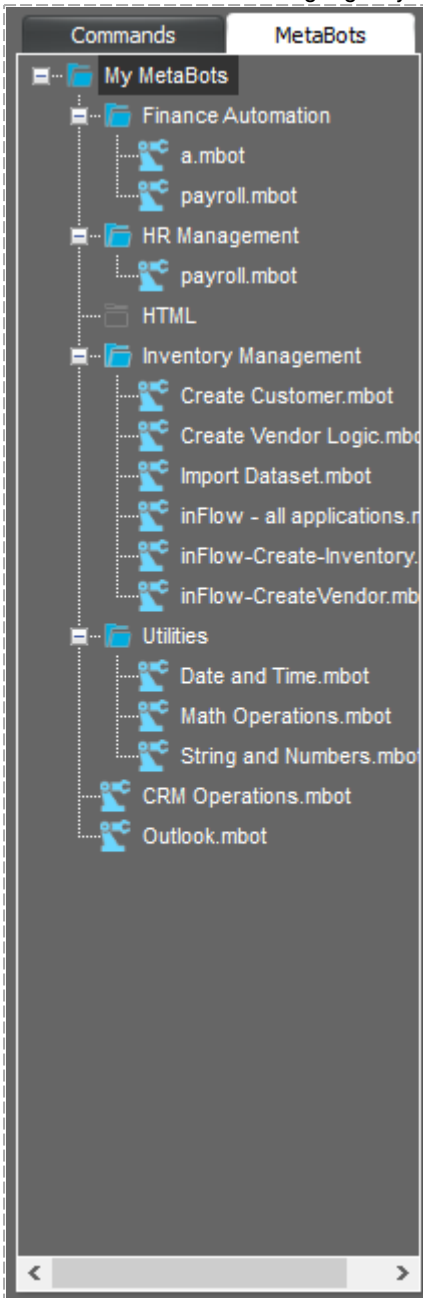


- The **Create Logic** window displays a list of MetaBots, the application for which it is created, and list of Logics if available.

- If no MetaBot is present in the MetaBots folder, the Create Logic window shows only the **My MetaBots** folder:

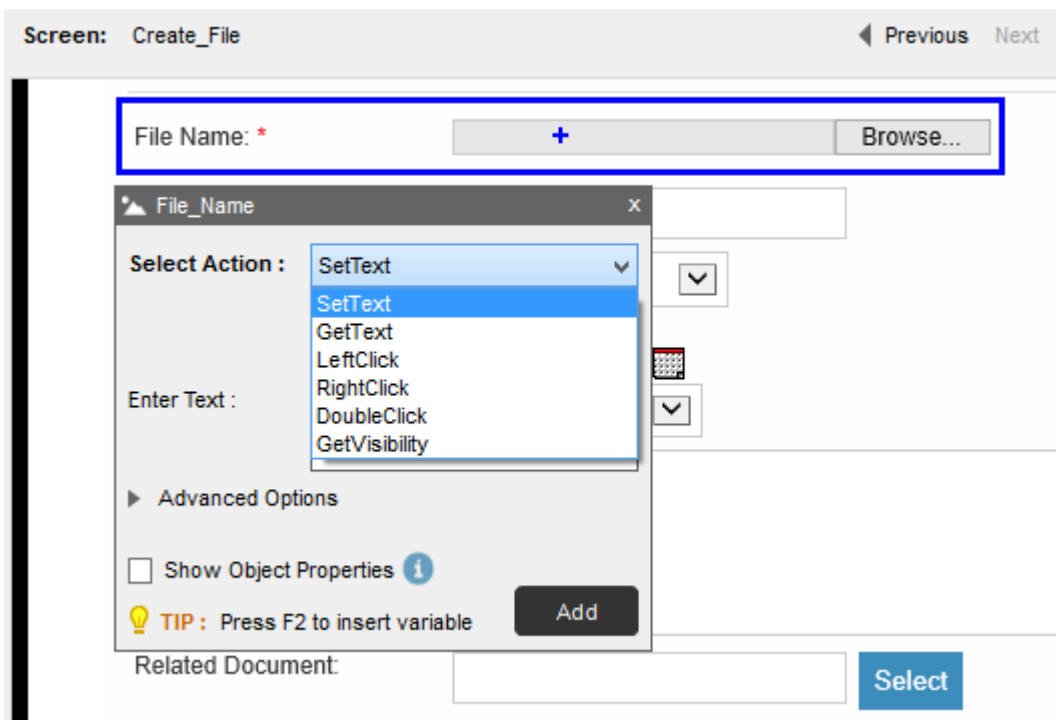


2. You can also add an existing logic by selecting a Metabot from the MetaBots panel:



Refer [Using MetaBot Logic in Bots](#) for details.

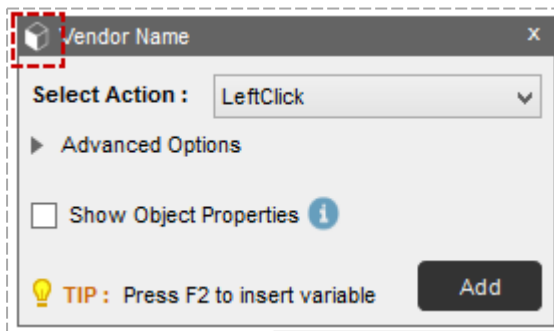
3. Configure commands and assets as per your automation work-flow.
4. To Add and/or Update various **Actions** to build your Logic, click on the captured Screen/DII.
5. To add actions in **Screens**, select relevant action in the floating **Properties** window that appears when you click on an object.



Refer [Selecting Actions in Logic Editor](#) for details on actions allowed in the Logic Editor

- The action selection depends upon the **Object** and **Control Type** selected for 'Screen' and Class and its API selected for 'DLLs'.
- Also, the properties and relevant actions are controlled by the **Play Mode** that has been selected while configuring the screen.

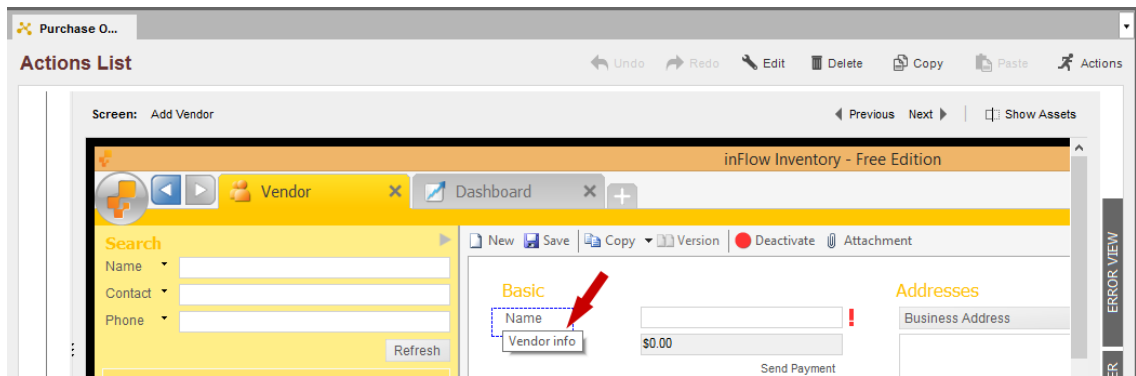
Tip: You can identify the 'Object Type' and it's 'Play Type' based upon the icon that appears in the title bar of the Properties window.



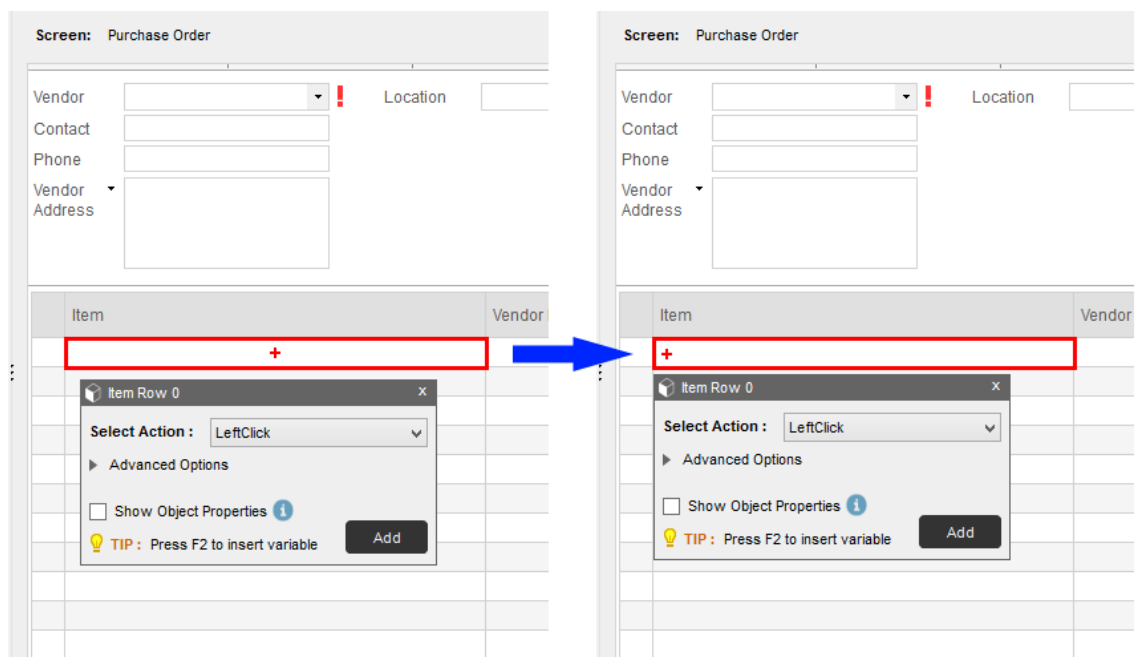
- All **non-custom** objects are highlighted with a **red outline** while **custom objects** are highlighted with a blue outline to allow you to easily identify the object type when you select them.

Tip: You can also hover your mouse pointer to identify the object type - custom objects are highlighted with blue-dotted line whereas non-custom objects are high-lighted with red-dotted line.

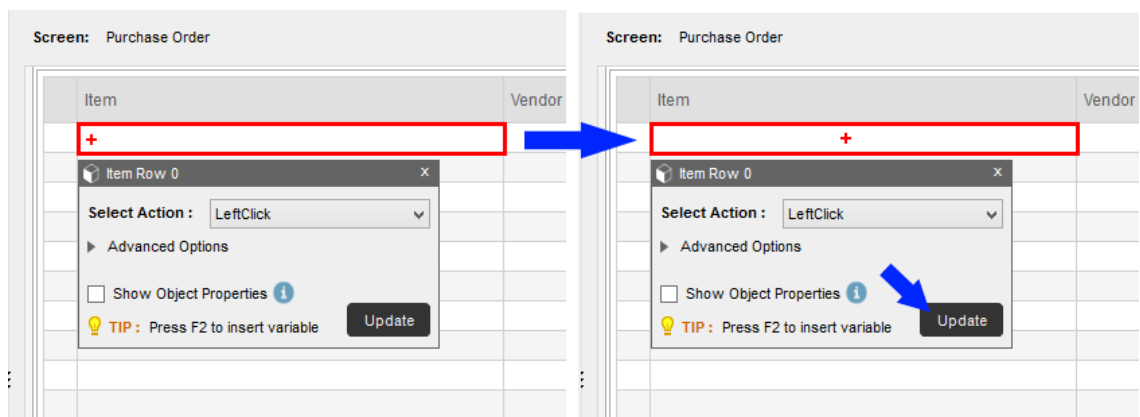
- The name of the object is also shown as tool tip. This labeling allows you to identify and select the required object:



- System performs the selected action at the location of the cross hair. You can place the cross hair within the corresponding object by dragging it. You can also place the cross hair anywhere outside of the object also. This can be helpful when you want to perform the chosen action relative to the selected object.
 - Drag the cross-hair to select the precise location where you want to perform an action:



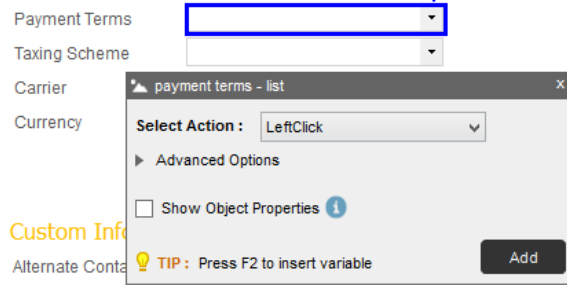
Note: If you **reset** the cross-hair position, ensure that you click **Update** to capture the click's latest position as shown:



- For objects that are difficult to recognize, you can perform a Left Click, Right Click or Double Click action relative to the target object's position. You can also use **linked objects** to easily identify difficult to search objects.
 - **Relative click** is used when your target object is difficult to search and you want to configure the bot to click relative to that particular object.

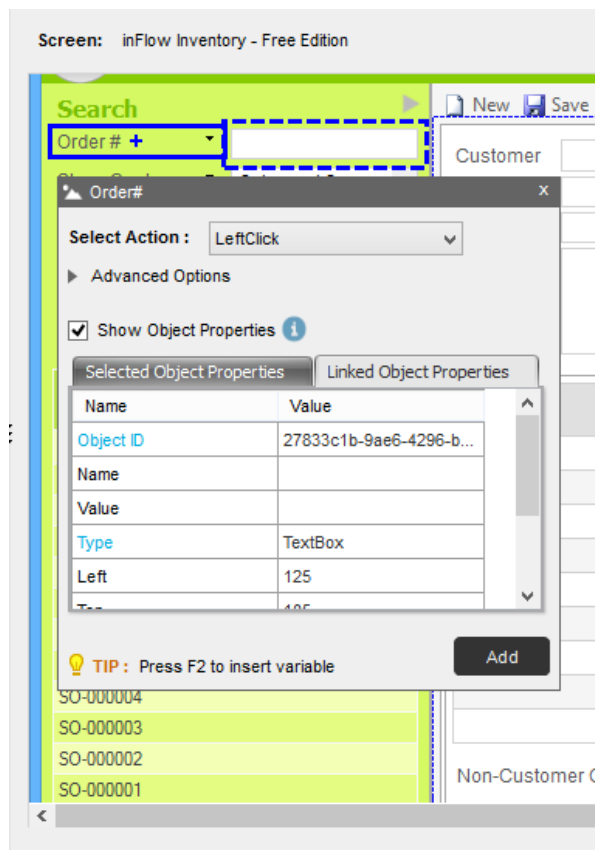
- You can drag the cross-hair to **change the click position**. This could also be outside the highlighted object's boundary.
For example, in the following illustration the space between two objects is very less. For more accuracy, you can choose a relative click positioned away from the subsequent object.

Purchasing Info

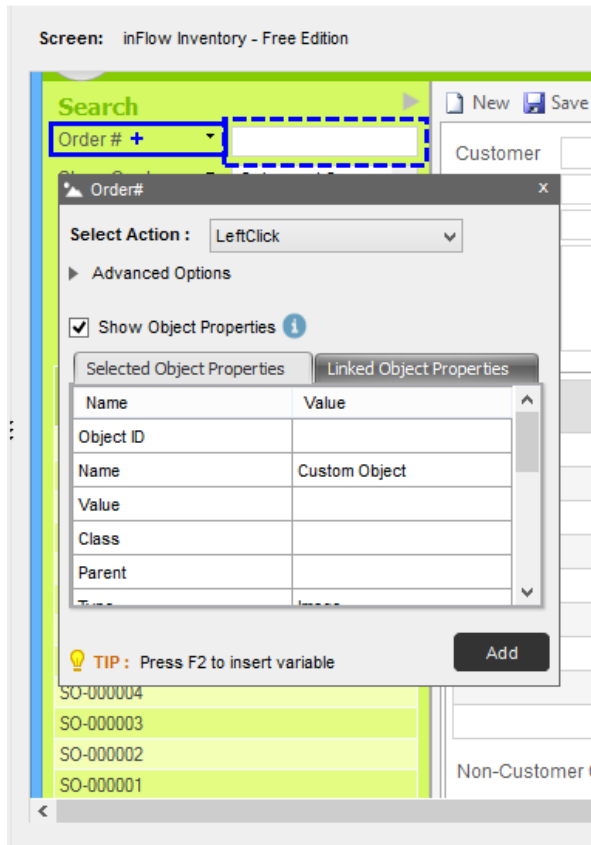


Note: The cross-hair color is **blue** for custom objects and **red** for standard objects.

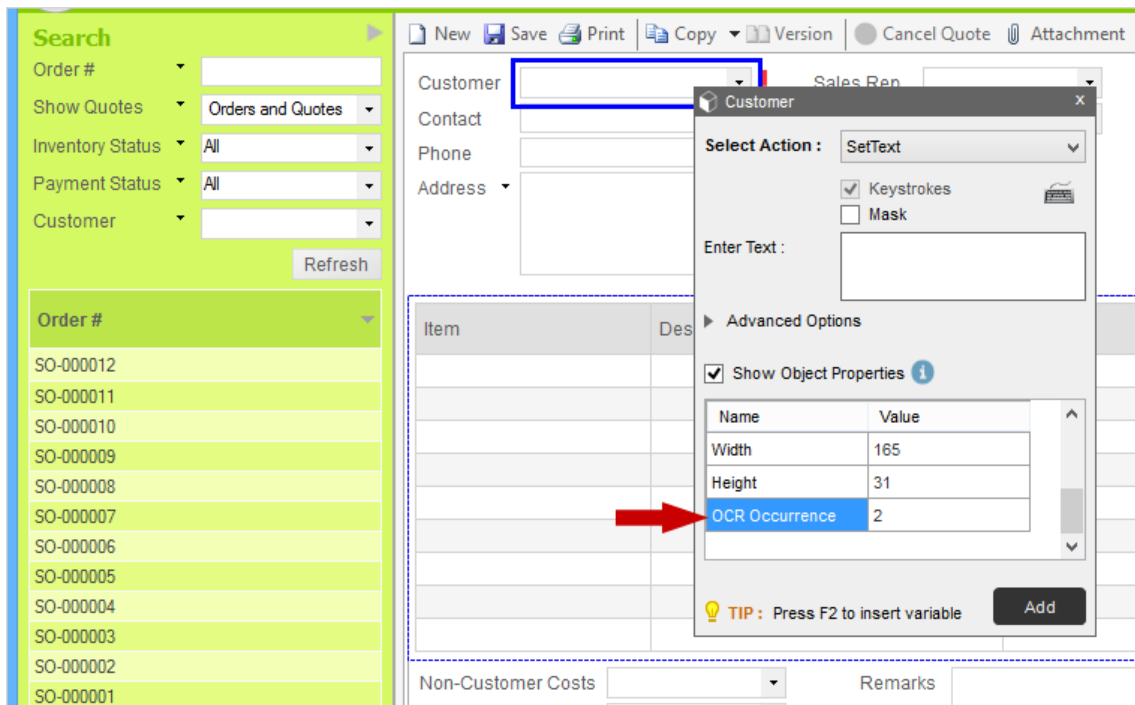
- If you update the action from click to some other action, for example SetText, you cannot perform relative click. The cross-hair will not be visible.
- If you have **linked** easily identifiable objects to other difficult to search objects during configuration, the objects will be searched based on the parent object properties/play type during automation execution. You can view these properties in the properties window when you select **Show Object Properties** for the parent and child objects.
 - The linked (child) object properties are shown by default:



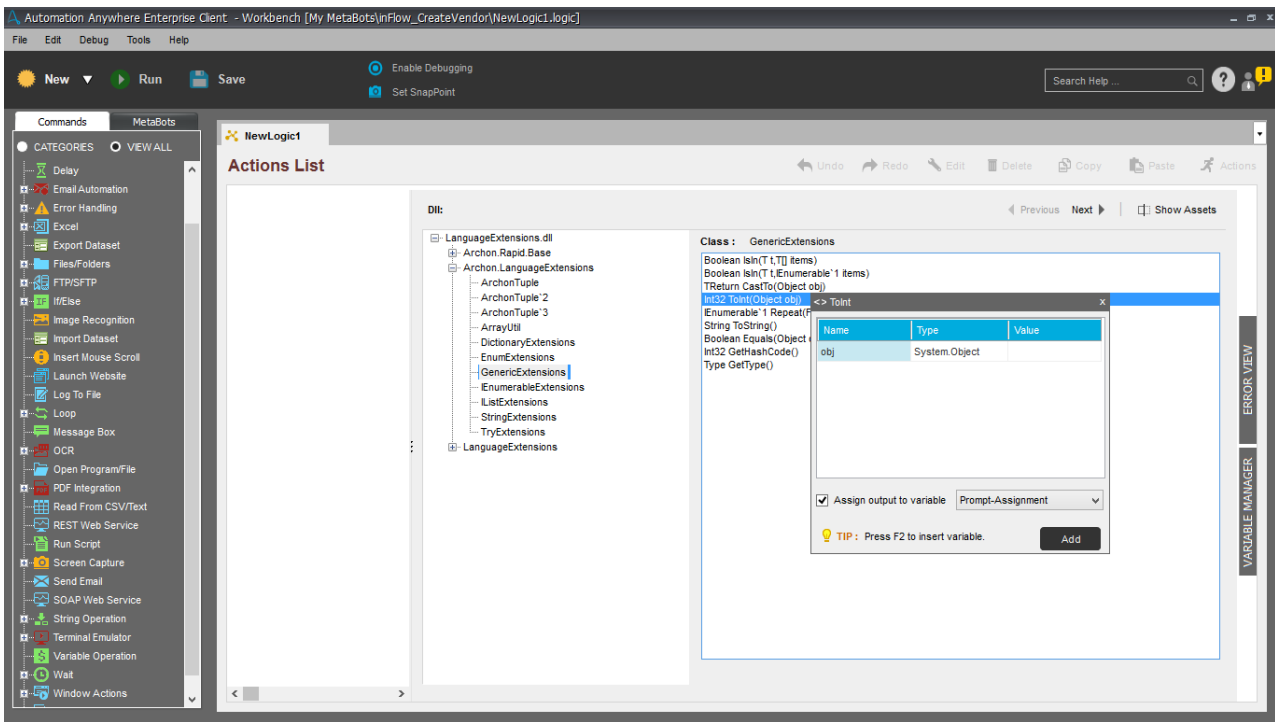
- b. To view the selected (parent) object properties, click on the tab corresponding to the linked object properties:




- For objects that are captured using OCR, you can view and edit the number of times it appears in the screen in the **OCR Occurrence** property:



6. To add actions to a DII, select relevant action in the floating Properties window that appears when you click on an API class.

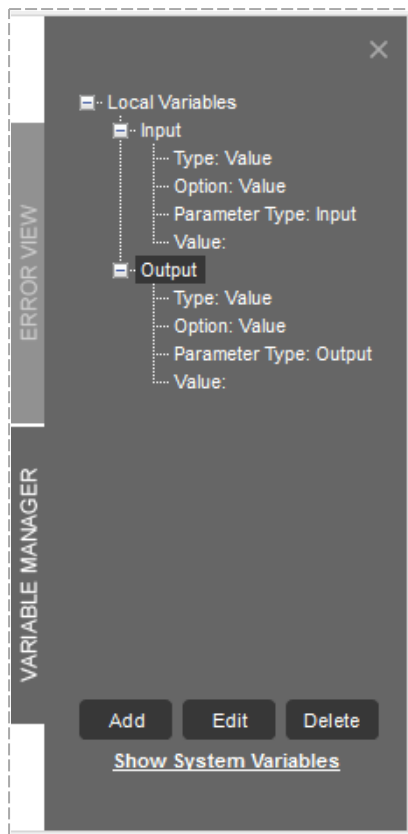


- Expanding your desired DLL will reveal its name-spaces and related classes.
- Select a class from the list. The supported APIs for the selected class are displayed on the right pane.
- From the list of APIs select the one for which you need to input a value.

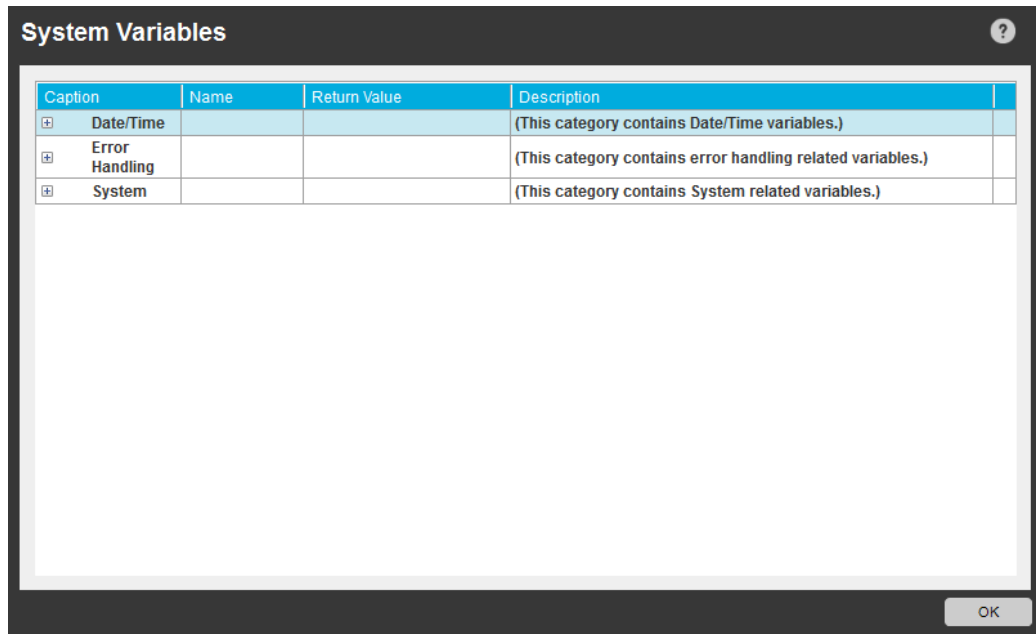
 **Tip:** Use an Array type variable to input values if you want to assign multiple values to a single parameter. [Learn More](#)

- You can also choose to assign the output value to a variable. To add and edit variables in the Logic, click on Variable Manager tab. [Learn More](#)

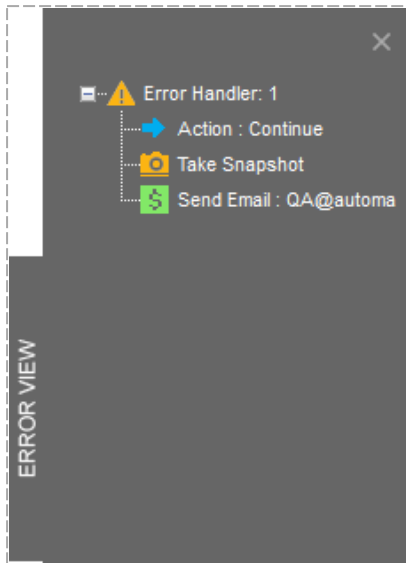
In Logic, variables are storage locations for known or unknown information. When building Logic, variables play an important role in maintaining or calculating information.



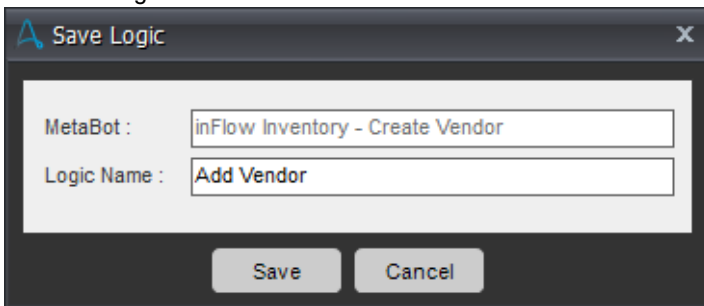
- Also assign System Variables, if required, while configuring the Logic. [Learn More](#)



- You can use the **Error View** to manage errors that occur in your Logic. Use the Error Handling command to manage those.



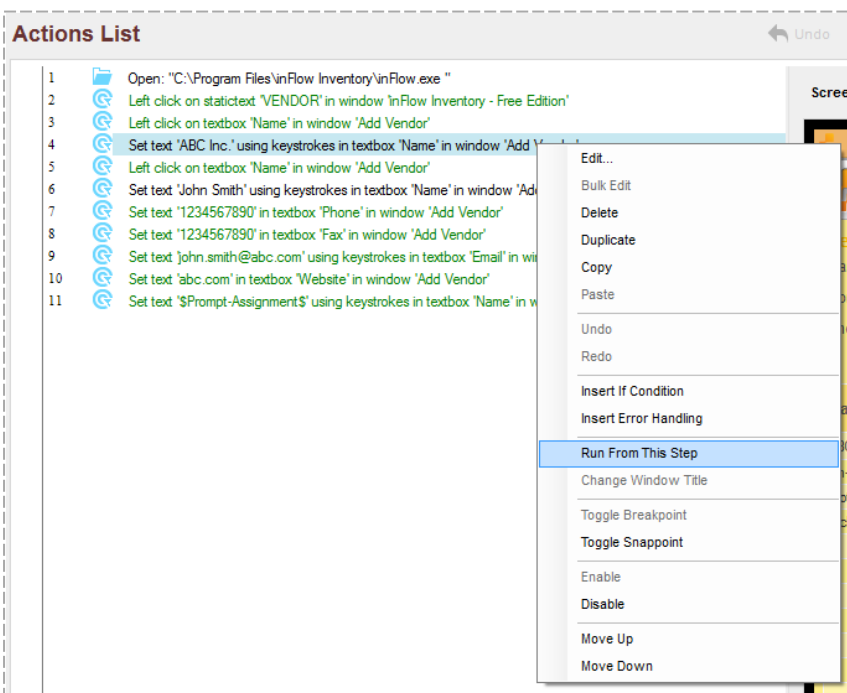
- Save the Logic.



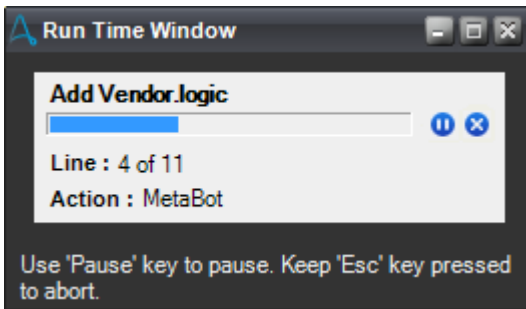
- You can now test run the Logic by clicking **Run**.




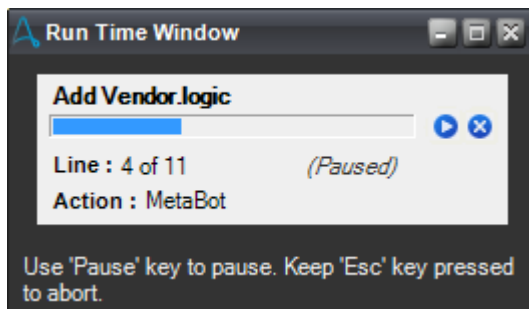
Tip: Use 'Run from this step' option if you want to skip the earlier steps while doing a test run.





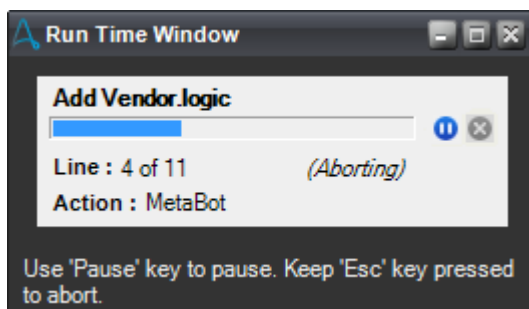
9. The Run Time window for a MetaBot appears towards right at the bottom of your screen:




- Click the Pause icon to pause the Logic run: 



- Click the Play icon to resume playing the Logic: 
- Click the Stop icon to discontinue : 

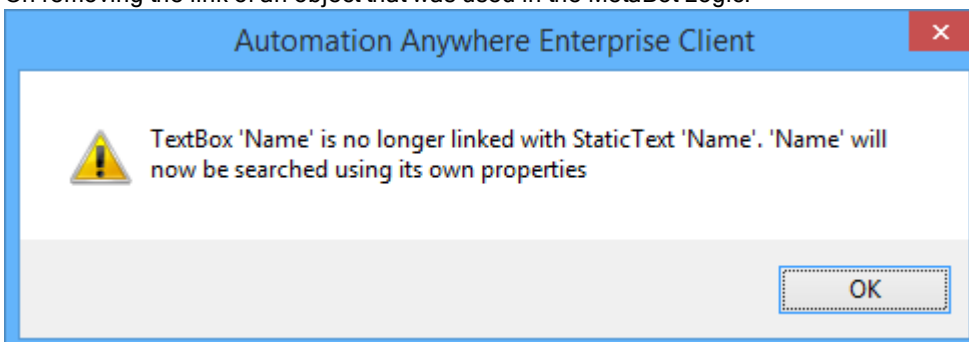


 **Note:** Once you are done creating your Logic, you can upload it to the Control Room for other MetaBot users. [Learn More](#)

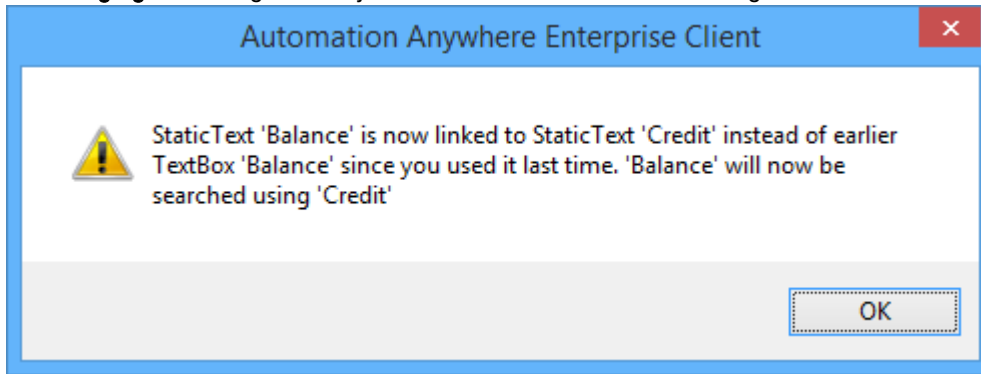
Object linking Messages

When you create automation using MetaBot Logics that are updated, you are shown messages based on the updates made:

1. On removing the link of an object that was used in the MetaBot Logic:



2. On **changing** the linking of an object that was used in the MetaBot Logic:



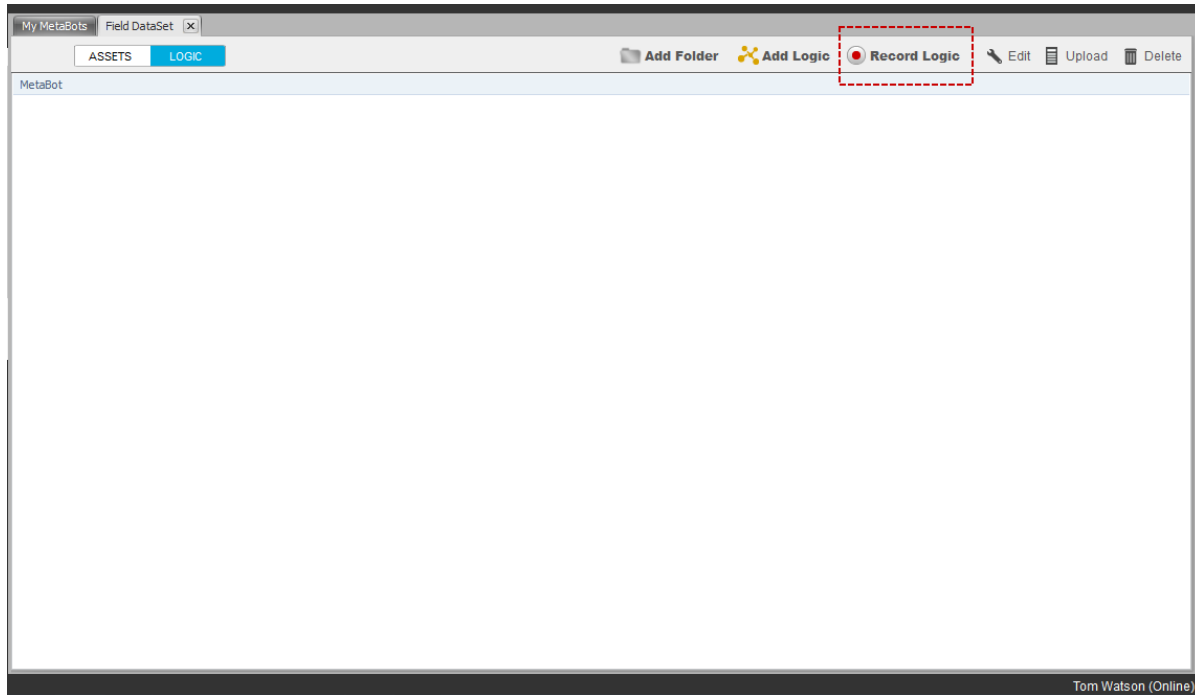
Recording Logic

At times you may want to capture the workflow directly instead of creating Screens, configuring them and then manually designing the Logic. To enable direct capture of workflow with Screens, you can opt to record Logic for an existing MetaBot.

Remember that to record logic for a **new** MetaBot, you will have to use the 'Record Screen(s) with Logic' option given in 'Record' from the MetaBot Designer panel. [Learn More](#)

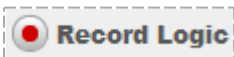
However, in order to record logic for an **existing** MetaBot, you will have to use 'Record Logic' from Logic section of the MetaBot Designer.

This topic describes the second option.

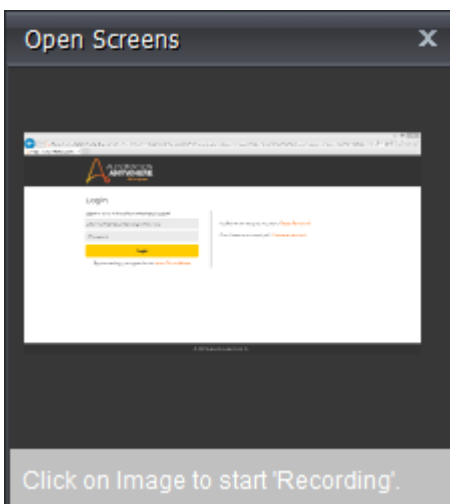


How to Record Logic

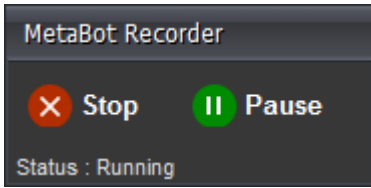
1. Open the target application.
2. Choose an existing MetaBot; the one that is relevant to that application.
3. Go to the Logic tab, select Record Logic.



4. Click on the image in the 'Open Screens' window to begin recording the Logic flow.



5. This will launch the application and the MetaBot Recorder window.



6. Perform actions as required for the Logic.



Note: The MetaBot Recorder window displays the status of the current action being performed.

7. Click on Stop once done. This will launch the Logic Editor wherein you can verify whether all actions are captured or not.
8. Modify if necessary and Save the Logic.



Note: Recorded Screens are added to your Assets library.

Selecting Actions in the Workbench

You can select various 'Actions' in the Workbench based upon the object and control type selected. Actions are allowed on HTML, .NET and Java Swing/AWT controls.

Actions allowed on HTML controls

The following Actions are allowed on HTML Controls:

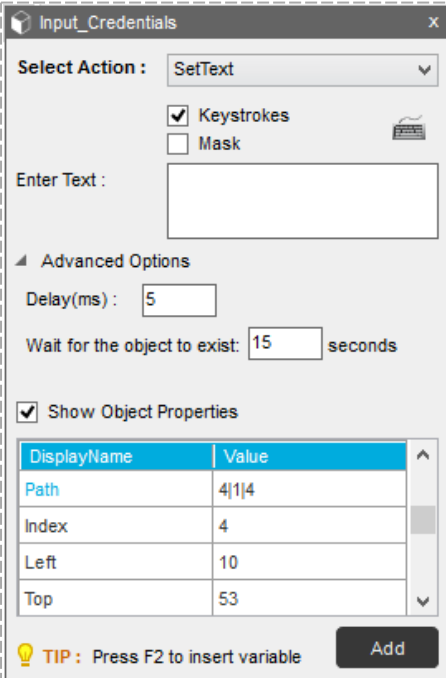
1. Click
2. DoubleClick
3. Right Click
4. Left Click
5. SetText
6. AppendText
7. GetProperty
8. GetVisibility
9. GetTotalItems
10. GetSelectedIndex
11. GetSelectedText
12. SelectItembyText
13. SelectItembyIndex
14. GetChildrenName
15. GetChildrenValue

The next section describes in brief the options available in Set Text, Append Text and Get Property.

Get Text, SetText and AppendText

Get Text, SetText and AppendText actions are available when the selected object types are Text/Text Box, Client, Password, Windows Control or Custom Objects.

While building your Logic, remember that the properties and relevant actions are controlled by the 'Play Type' configured for the selected Screen.



Input_Credentials

Select Action : SetText

Keystrokes Mask

Enter Text :

Advanced Options

Delay(ms) :


Wait for the object to exist: seconds

Show Object Properties

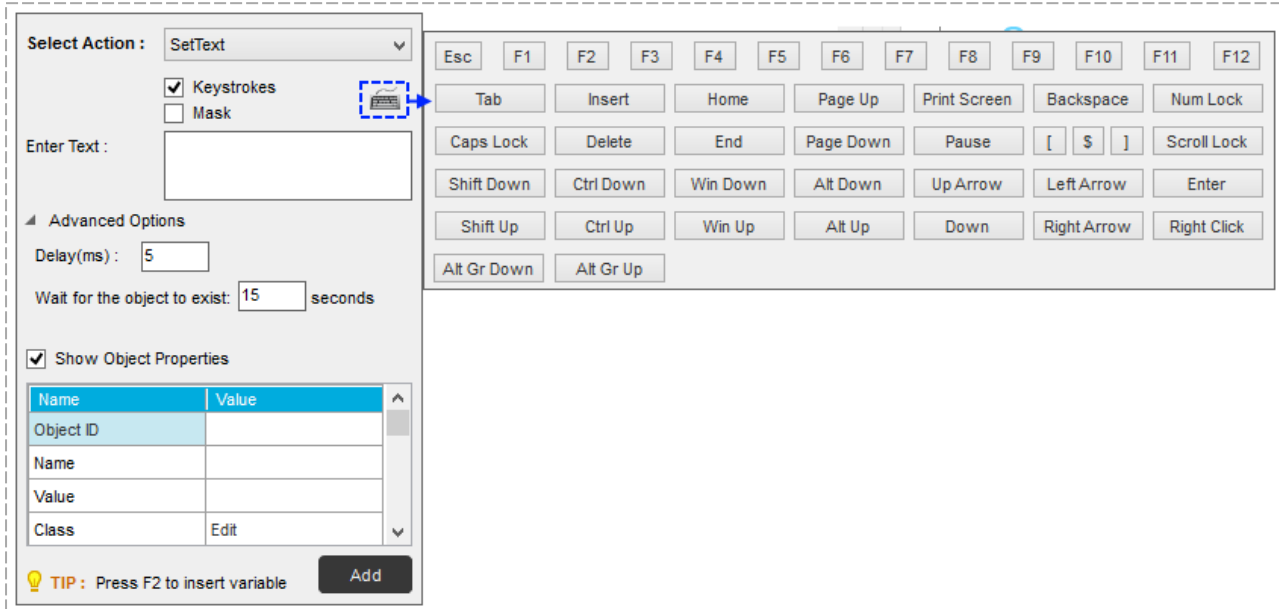
| DisplayName | Value |
|-------------|-------|
| Path | 4 1 4 |
| Index | 4 |
| Left | 10 |
| Top | 53 |

TIP : Press F2 to insert variable

- or Action type as 'SetText', you can use keystrokes option to emulate entering text using a keyboard.

 Note: In SetText, 'Keystrokes' is selected by default when the Play Type is either 'Image' or 'Coordinate'.

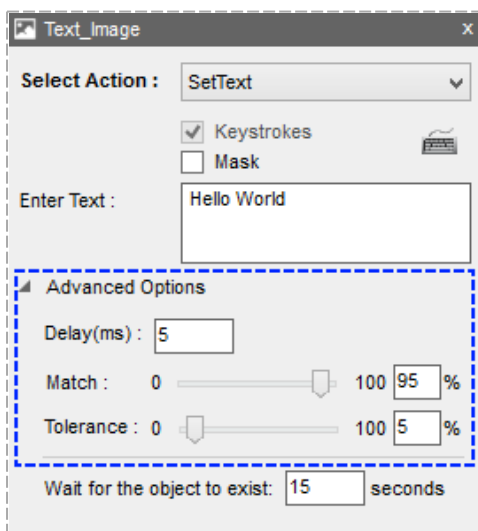
- Use virtual keyboard to enter certain special keyboard keystrokes:



- Optionally, select 'Mask' to encrypt the keystrokes.
- Enter the required text in 'Enter Text box'. Optionally, assign a variable to 'Delay'.
- **Advanced Options** : Provide appropriate 'Delay' between keystrokes in 'Advanced Options'.

Provide 'Wait time for the object to exist' to allow for object load time and ensure the logic does not fail during play time.

If play type is set to 'Image' for the control, you can configure image Match and Tolerance values in percentage.

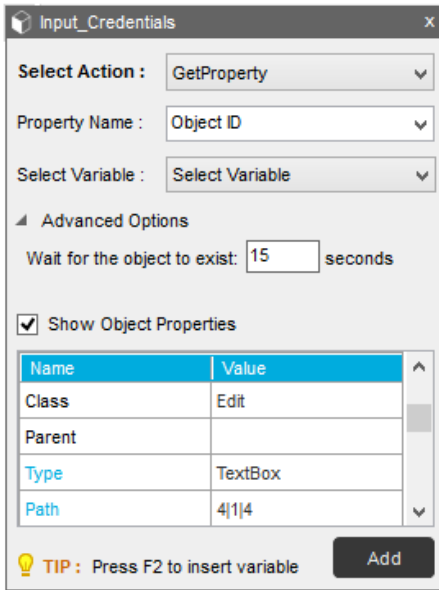


A Match value defines the extent of overall mismatch allowed between Image1 and Image2; Tolerance defines the extent of mismatch allowed between any two pixels under comparison.

Similarly, if the play type is set to 'Object' for the control, you can select your OCR engine and configure the Tolerance values in percentage.

GetProperty

Use the 'Get Property' action when you want to search the objects based on their properties during play time.




The screenshot shows the 'Input_Credentials' dialog box with the following configuration:

- Select Action: GetProperty
- Property Name: Object ID
- Select Variable: Select Variable
- Advanced Options:
 - Wait for the object to exist: 15 seconds
 - Show Object Properties:
- Object Properties Table:

| Name | Value |
|--------|---------|
| Class | Edit |
| Parent | |
| Type | TextBox |
| Path | 4 1 4 |
- TIP: Press F2 to insert variable
- Add button

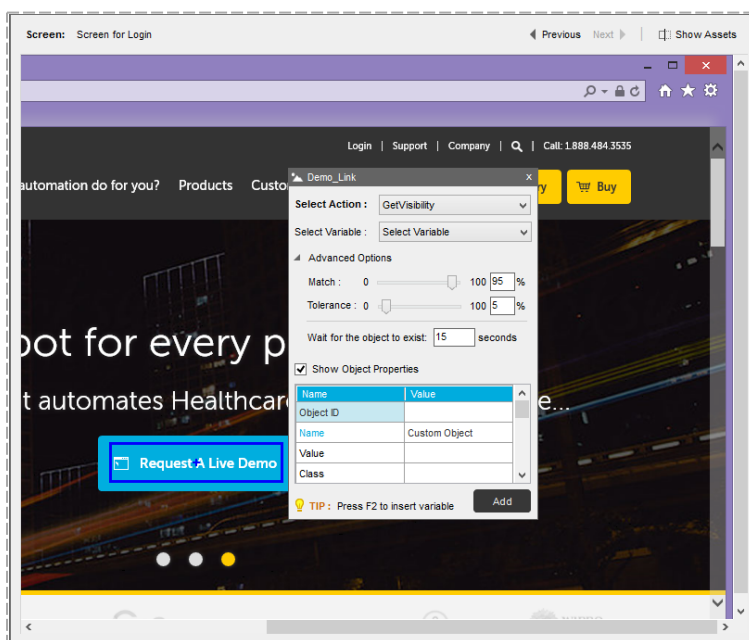
When you select action as 'Get Property', you will be able to select properties names such as Object ID, Name, Value, Class, Type, Index, Description, State, IsVisible, IsProtected etc based on the object control selected.

 Tip: Use the 'IsVisible' property to identify whether a specific object is visible or not. For custom objects, you can achieve this by using the 'GetVisibility' action; refer the next section.

GetVisibility

Use the GetVisibility action to build a logic based on an object's visibility during play time. This screen area could be a custom object or an object with Play Type Image. The 'GetVisibility' action returns the visibility status as True or False.


To add a 'GetVisibility' action in the Logic Editor, the object should be configured for Play Type 'Image'.



The screenshot shows the 'Demo_Link' dialog box with the following configuration:

- Select Action: GetVisibility
- Select Variable: Select Variable
- Advanced Options:
 - Match: 0 to 100 (95%)
 - Tolerance: 0 to 100 (5%)
 - Wait for the object to exist: 15 seconds
 - Show Object Properties:
- Object Properties Table:

| Name | Value |
|-----------|---------------|
| Object ID | |
| Name | Custom Object |
| Value | |
| Class | |
- TIP: Press F2 to insert variable
- Add button

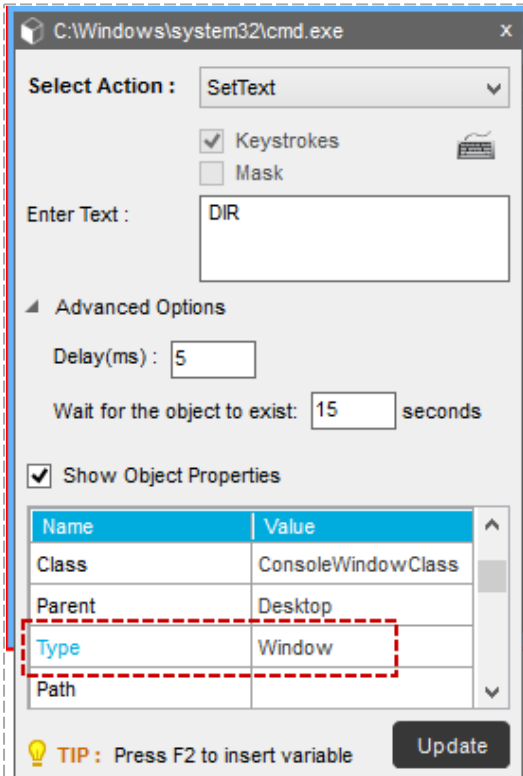
 Tip: GetVisibility can be combined with conditional commands such as If.

Actions allowed on Window Controls

The following Actions are allowed on Window Controls:


1. Click
2. DoubleClick
3. RightClick
4. LeftClick
5. SetText
6. AppendText
7. GetProperty
8. GetChildrenName
9. GetChildrenValue


SetText for Window Control - Use the action type 'SetText' for Window Controls. Select the entire window and specify the action type.



The screenshot shows the configuration for the 'SetText' action on a 'C:\Windows\system32\cmd.exe' window. The 'Select Action' dropdown is set to 'SetText'. The 'Keystrokes' checkbox is checked, and the 'Mask' checkbox is unchecked. The 'Enter Text' field contains 'DIR'. Under 'Advanced Options', 'Delay(ms)' is set to 5 and 'Wait for the object to exist' is set to 15 seconds. The 'Show Object Properties' checkbox is checked, and a table displays the object's properties. The 'Type' property is highlighted with a red dashed box.

| Name | Value |
|--------|--------------------|
| Class | ConsoleWindowClass |
| Parent | Desktop |
| Type | Window |
| Path | |

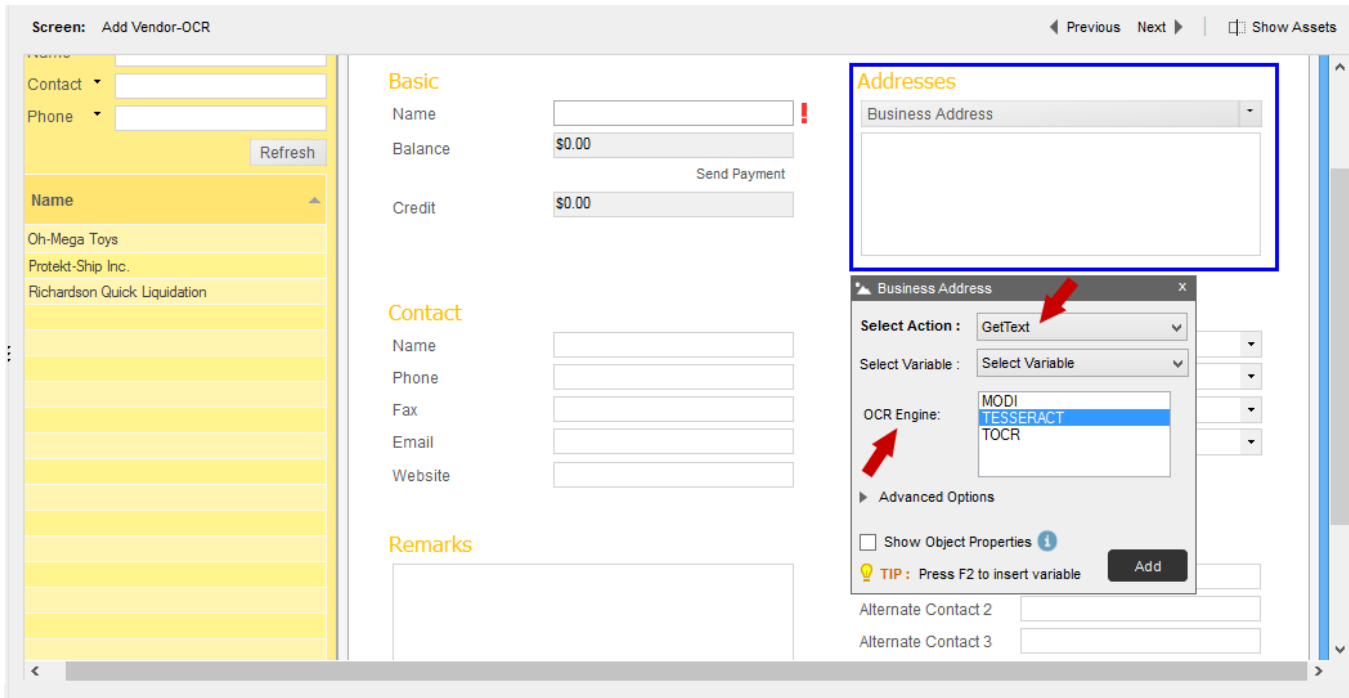
 TIP : Press F2 to insert variable Update

 Note: The Window Control for Play Type 'Object' uses keystrokes by default.

OCR screens

For screens that are captured using OCR technology, when you select play type as Image for custom objects, you are allowed actions - SetText, GetText, LeftClick, RightClick, DoubleClick, and GetVisibility.

- **GetText** - When you select **GetText** for play type **Image** for a custom object, you can choose an OCR engine to extract text while creating a MetaBot Logic. You can choose an OCR engine other than the one selected during screen configuration for optimizing your automation.



Screen: Add Vendor-OCR

Basic

Name !

Balance Send Payment

Credit

Contact

Name

Phone

Fax

Email

Website

Remarks

Addresses

Business Address

Business Address

Select Action: GetText

Select Variable: Select Variable

OCR Engine: MODI, TESSERACT, TOCR

Advanced Options

Show Object Properties

TIP: Press F2 to insert variable

Add

Alternate Contact 2

Alternate Contact 3

Section: MetaBots - Variables

Adding, Editing and Deleting Variables

Variables are storage locations for known or unknown information. When creating Logic, variables play an important role in maintaining or calculating information. Variables can help you in a number of ways, from fetching online data to transferring data between applications.

Automation Anywhere allows you to use various types of variables - locally defined or system defined in a MetaBot Logic. The following section describes how to add, edit and delete locally created variables.

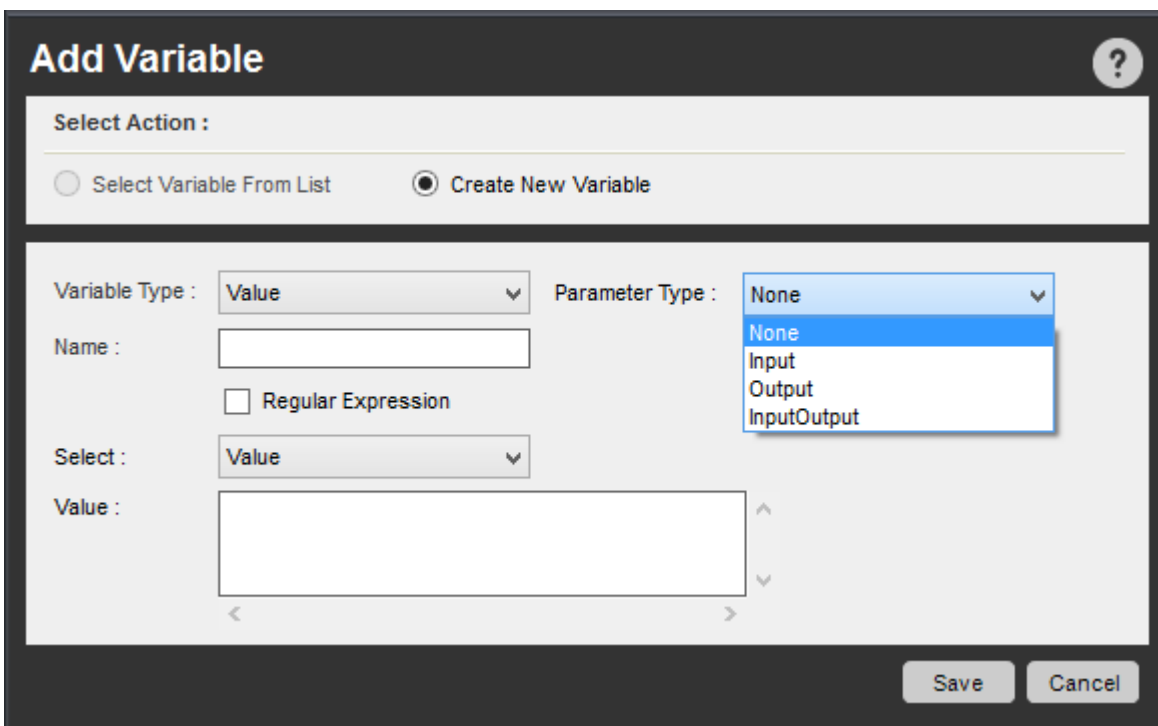
Adding Variables to a MetaBot Logic

You can use the F2 function key to list all user and system variables that are available for insertion in a MetaBot Logic.

User defined variable types

You can create **four** types of variables that can be used in a MetaBot Logic - Value, List, Array, and Random.

1. **Value** - You can use a value type variable when you need to hold a single data point and use it in multiple places. This "placeholder" value can represent either text or numeric data.



After you create the variable, you can use it by inserting the variable in several of the Logic Editor commands. When the value of the variable is modified, this value is reflected in any subsequent commands.

You can also mark a value type variable as 'Regular Expression'. These, you can use when creating TaskBots /MetaBot Logics that require pattern based searches in files, folders and window title commands.

2. **List** - You can use a list type variable when you need to retrieve multiple values, one by one. It is basically one dimensional placeholder for data.

Add Variable ?

Select Action :

Select Variable From List Create New Variable

Variable Type : List Parameter Type : None

Name : List Value :

Make Random

Select : Value

Add To List
Edit
Delete

Save
Cancel

Common uses of list variables include:

- Sending email to multiple recipients
- Passing different values inside of a loop
- Searching multiple web addresses

The values can represent either text or numeric data.

3. **Array** - Use an Array type variable for creating staging areas for data that need to be retrieved by your process as it runs. It is a two-dimensional variable that holds multiple values in a table of rows and columns.

Add Variable ?

Select Action :

Select Variable From List Create New Variable

Variable Type : Array Parameter Type : None

Name :

Select : Value

Row(s) : 1 Column(s) : 1 Initialize Values

Save
Cancel

4. **Random** - You can create two types of random variables: string and numerical. Random variables are useful when you need to generate a random, repetitive string or numerical set. The values are generated when you run the TaskBot / MetaBot Logic.

Add Variable ?

Select Action :

Select Variable From List
 Create New Variable

Variable Type : Random
 Parameter Type : None

Name :

Random String
 Random Number - Range

String Length :
 From :

To :

Save
Cancel

Common uses of Random variables include:

- String: Generating test data as input for fields or forms
- Numerical: Generating ID numbers in batch

Parameter Types

Since MetaBot Logics can be used in TaskBots, you might want to pass on certain parameters to another TaskBot / Logic (of the same MetaBot) when executing a Bot. For this, you can use variables and define their parameter types. Defining the type of parameters ensures optimum use of variables across Bots.

In MetaBot Logic, you can define **four** types of parameters - None, Input, Output, and InputOutput.

Refer [Variables - Parameter Types](#) for details.

Editing a Variable

You can edit and modify any local variable that you have created. In addition, you can edit the pre-defined variables.

To edit a variable, follow these steps:

1. Select the variable you want to edit.
2. Click the Edit button or right-click on the variable and select Edit. The Edit Variable window is displayed.
3. Modify the variable fields as necessary. You can change the variable type, the name, description or the method of determining value.
4. Click Save.

Adding or Editing Description

You can add and edit description to the Value, Array, and List type variables when the Parameter Type is Input, Output, and InputOutput. This is highly useful as it provides contextualized help without any clicks, while using any the MetaBot Utilities/Logics on TaskBots.

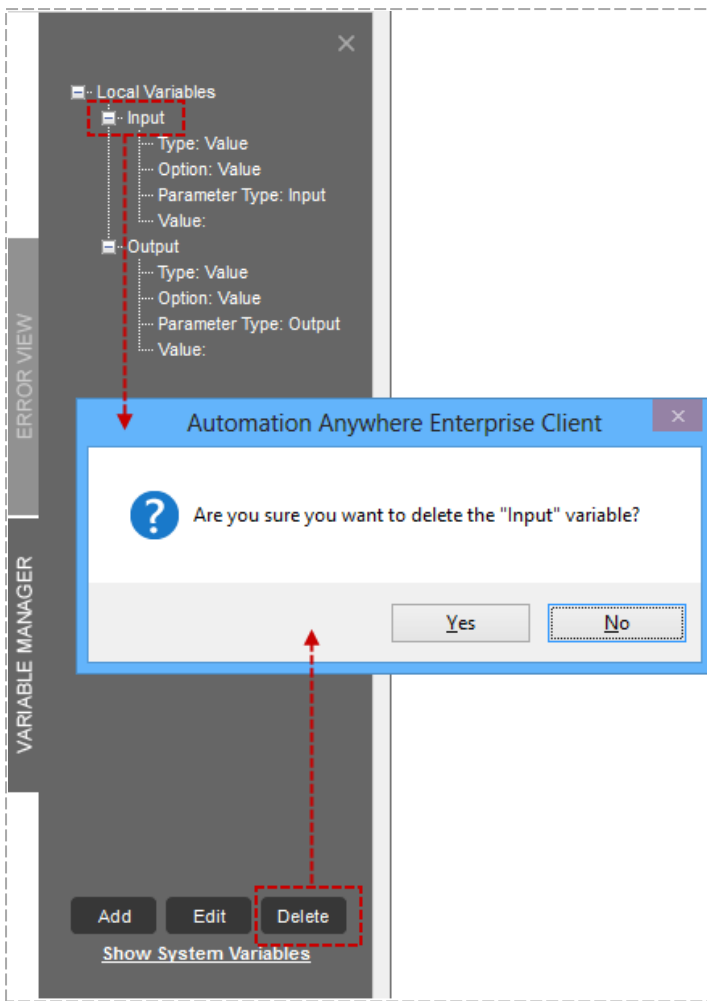
Deleting a Variable


To delete a variable, use the Variable Manager.

For instance if you 'Copy All' variables to another Logic, you might need to delete several variables that are redundant in the new navigational flow.

To delete a variable,

1. Select the variable you want to delete.
2. Click the Delete button or right-click on the variable and select Delete.
3. When the confirmation message is displayed, click yes.




 Note: You can delete variables one at a time.

System Variables

Automation Anywhere provides powerful pre-defined system variables that you can use to design Logic Blocks.

System Variables ?

| Caption | Name | Return Value | Description |
|---------|----------------|--------------|--|
| + | Date/Time | | (This category contains Date/Time variables.) |
| + | Error Handling | | (This category contains error handling related variables.) |
| + | System | | (This category contains System related variables.) |

 Note: Though similar, not to be confused with System Variables available in Task Editor.


System Variable types that can be used in the Logic Block include:


- **Date/Time:** System-related date and time variables.
- **Error Handling:** Error Handling related variables.
- **System:** Variables specific to a particular client machine.

Date/Time System Variables

| Caption | Name | Return Value | Description |
|---------|----------------|--------------|---|
| - | Date/Time | | (This category contains Date/Time variables.) |
| | Year | Integer | Returns Year. e.g. if date is 02/23/04, it returns 2004. |
| | Month | Integer | Returns System Month. e.g. if date is 02/23/04, it returns 2. |
| | Day | Integer | Returns System Day. e.g. if date is 02/23/04, it returns 23. |
| | Date | Date | Returns System Date in mm/dd/yyyy HH:mm:ss format. ... |
| | Hour | Integer | Returns System Hour. e.g. if time is 17:33:49, it returns 17. |
| | Minute | Integer | Returns System Minute. e.g. if time is 17:33:49, it returns 33. |
| | Second | Integer | Returns System Second. e.g. if time is 17:33:49, it returns 49. |
| | Milliseco | Integer | Returns System Millisecond. e.g. if time is 17:33:49:10, it returns 10. |
| + | Error Handling | | (This category contains error handling related variables.) |
| + | System | | (This category contains System related variables.) |

You can use the set of Date and Time system variables to insert or monitor the current date and time of a system as the navigational flow is implemented.

 Tip: You can change the date format for the System Date.

Click  to change the date format:

| Date | Date | Date | Returns System Date in mm/dd/yyyy HH:mm:ss format. |
|-----------------------|-------------|---------|--|
| Hour | Hour | Integer | |
| Minute | Minute | Integer | |
| Second | Second | Integer | |
| Milliseco | Millisecond | Integer | |
| Error Handling | | | |
| System | | | |

Select Date Format

Date format: mm/dd/yyyy HH:mm:ss

- dd/mm/yyyy hh:mm:ss AM/PM
- dd/mm/yyyy hh:mm AM/PM
- dd/mm/yyyy HH:mm:ss
- dd/mm/yyyy HH:mm
- m/dd/yyyy
- m/d/yy
- yy/mm/dd
- yyyy-mm-dd

Error Handling Variables

| Caption | Name | Return Value | Description |
|-----------------------|-------------------|--------------|--|
| Date/Time | | | (This category contains Date/Time variables.) |
| Error Handling | | | (This category contains error handling related variables.) |
| Error Line Number | Error Line Number | Integer | Returns Automation Anywhere task error line number. |
| Error Description | Error Description | String | Returns Automation Anywhere task error line description. |
| System | | | (This category contains System related variables.) |

Use the set of Error Handling system variables to return task error line number and description.

System Type System Variables

| Caption | Name | Return Value | Description |
|--------------------|-------------------|--------------|--|
| System | | | (This category contains System related variables.) |
| Machine | Machine | String | Returns Machine Name. |
| Clipboard | Clipboard | String | Returns Clipboard text data. |
| System | System (Name) | String | Name = { "Path", "PATHEXT", "USERDOMAIN", "PROCESSOR_ARCHITECTURE", "ProgramW6432", "PUBLIC", "APPDATA", "windir", "LOCALAPPDATA", "CommonProgramW6432", "USERDNSDOMAIN", "TMP", "USERPROFILE", "ProgramFiles", "PROCESSOR_LEVEL", "FP_NO_HOST_CHECK", "HOMEPATH", "COMPUTERNAME", "PROCESSOR_ARCHITEW6432", "USERNAME", "NUMBER_OF_PROCESSORS", "PROCESSOR_IDENTIFIER", "SystemRoot", "ComSpec", "LOGONSERVER", "TEMP", "ProgramFiles(x86)", "CommonProgramFiles", "__COMPAT_LAYER", "USERDOMAIN_ROAMINGPROFILE", "PROCESSOR_REVISION", "CommonProgramFiles(x86)", "ALLUSERSPROFILE", "SystemDrive", "PSModulePath", "OS", "ProgramData", "HOMEDRIVE" } |
| AAApplicationPath | Application Path | String | Returns Product Application Path. |
| AAInstallationPath | Installation Path | String | Returns Product Installation Path. |

You can use 'System' variables to include parameters in your automation task that are related to a particular computer. The variables return actual system settings and parameters, such as RAM, CPU/RAM usage, and total RAM.

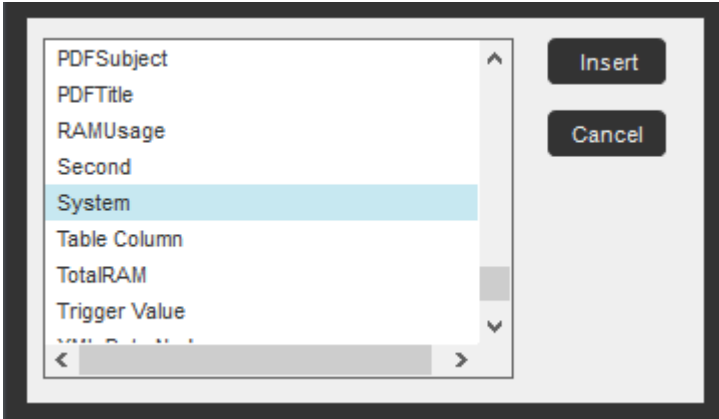
Common Use Case: These variables are useful when the performance of a system needs to be tracked during an activity; for instance load testing.

The following table provides names, return values, and descriptions for the system-related system variables.

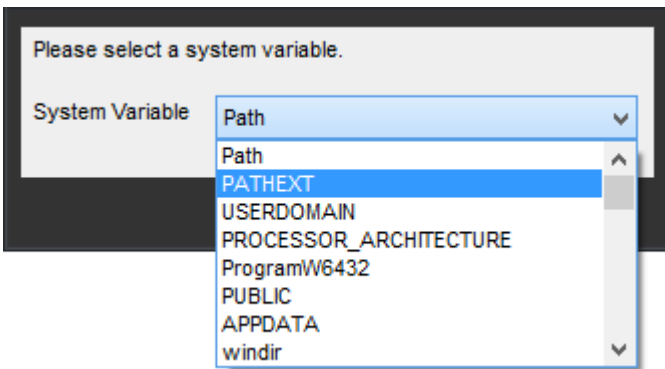
*When you select the System variable, a menu is displayed from which you can select the specific system variable (see steps below).

Steps to select System from Variables:

1. Click F2 and you will see Insert Variable window.



2. Select System and click Insert; a pop up window for System Variable Option appears.

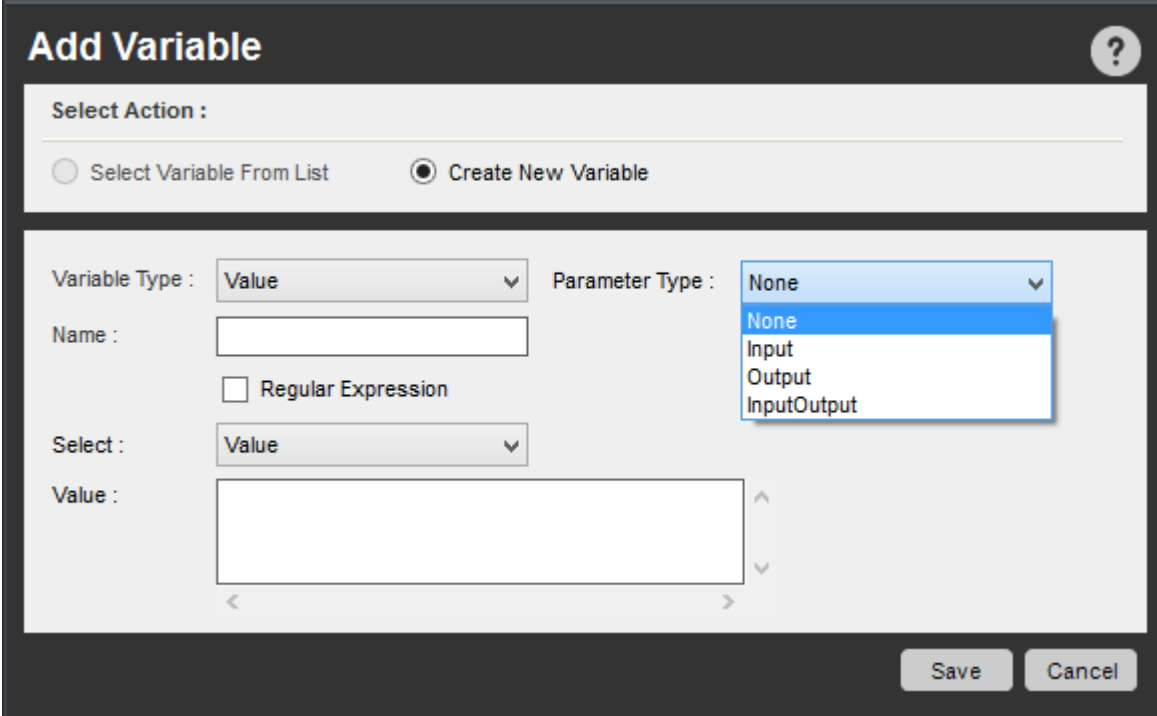


3. Click OK and insert the System Variable.

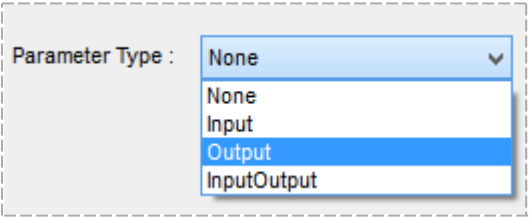
Variables - Parameter Types

Since MetaBot Logics can be used in other TaskBots / Logics (of the same MetaBot), you might want to pass on certain variable parameters to another TaskBot / MetaBot Logic when executing a Bot. For this, you can use variables and define their parameter types, which can then be added as Input or Output Parameters in an automation.

You can add and update parameter types from Variable Manager → Add/Edit :



Parameter Types



In MetaBot Logic, you can define **four** types of parameters for all variable types - None, Input, Output, and InputOutput

1. **None** - When you want to use the variable value only in the MetaBot Logic that it was created for, define the variable as **None**, You cannot pass the value of such variables to another TaskBot or MetaBot Logic. Its value can only be read during Logic execution.
2. **Input** - When you want the variable to accept values from TaskBot or MetaBot Logic of the same MetaBots, define the variable as **Input** Parameter Type. You can also assign any other value/variable as an Input Parameter in the automation.
3. **Output** - When you want to use the variable value as an output in the MetaBot Logic it was created for and pass on the value to other TaskBot or MetaBot Logic as an Output Parameter, define the variable as **Output** Parameter Type.
4. **InputOutput** - When you want to use the variable value as both input and output in the MetaBot Logic it was created for and pass on the value to other TaskBot or MetaBot Logic, define the variable as **InputOutput** Parameter Type. You can assign a value/variable as an Input Parameter and a variable as an Output Parameter in the automation.

Passing parameters from and to MetaBot Logic

As an Automation Expert, when you are creating automation that comprise a combination of TaskBots and MetaBot Logics, you would want to pass its parameter values from one to another for smooth functioning of your automation.

Automation Anywhere allows you to pass parameters from a Logic to other TaskBots, Logics, and DII APIs. The reverse is also possible - you can pass parameters from TaskBots, Logics, and DII APIs to Logics.

What is 'Passing of Parameters'?

When you create MetaBot Logic, you want to ensure that it can be optimally used in various TaskBots and MetaBot Logics. To achieve this, you first have to create variables with different parameter types - None, Input, Output, and InputOutput.

Each variable, based on its parameter type, is then used as an input parameter or output parameter or both in a TaskBot/ MetaBot Logic.


When it is used as an input parameter, you can add values to the variable or assign another variable as its value.

When it is used as an output parameter, you can only assign variables as its value. This will be read during automation execution.

This is passing of parameters from and to MetaBot Logic.

Refer [Using MetaBot Logic in TaskBots](#) to know how to use variables as parameters.


Refer [Variables - Parameter Types](#) for details on types of parameters.

 Note: If you have upgraded from Automation Anywhere Enterprise 10.x to the current version, refer the following sections on variable behavior while passing it as a parameter.

Passing parameters from Logic to TaskBot and from TaskBot to Logic

The table below shows the variable behavior when it is passed as a parameter from Logic to TaskBot and vice versa:


| Variable type in TaskBot (T1) / Logic (L1) | Variable type in TaskBot (T2) / Logic (L2) | Behavior |
|--|--|---|
| Value (V1) | Value (V2) | V1 overwrites V2 |
| | List (L2) | V2 is converted to 1x1 list and is assigned V1's value |
| | Array (A2) | V2 is converted to 1x1 Array and is assigned V1's value |
| Random (R1) | Value (V2) | V1 overwrites V2 |
| | List (L2) | V2 is converted to 1x1 list and is assigned V1's value |
| | Array(A2) | V2 is converted to 1x1 Array and is assigned V1's value |
| List (L1) | Value (V2) | First index of the list is assigned when it is used outside of the loop. In loop, the value is assigned with reference to counter. If it is outside of the range, then first index is considered. |
| | List (L2) | V1 overwrites V2 |
| | Array (A2) | V2 is converted to nx1 Array and is assigned V1's value |
| Array (A1) | Value (V2) | You have to input row and column of V1 and its value is assigned to V2 |
| | List (L2) | You have to input the column of V1 and its value is assigned to V2 |
| | Array (A2) | V1 overwrites V2 |

 Note: The target variable **value** is overwritten by source variable value. However target variable **type** remains unchanged.

Passing parameters from Logic to API DLLs and from API DLLs to Logic

The table below shows the variable behavior when it is passed as a parameter from Logic to API DLLs and vice versa:

| Variable type being passed from Logic | Variable type of DLL API | Behavior |
|---------------------------------------|--------------------------|---|
| Value (V1) | Value (V2) | V1 overwrites V2 |
| | List (L2) | Not supported. Read-only cell |
| | Array (A2) | Not supported. Read-only cell |
| Random (R1) | Value (V2) | V1 overwrites V2 |
| | List (L2) | Not supported. Read-only cell |
| | Array (A2) | Not supported. Read-only cell |
| List (L1) | Value (V2) | First index of the list is assigned when it is used outside of the loop. In loop, the value is assigned with reference to counter. If it is outside of the range, then first index is considered. |
| | List (L2) | V1 overwrites V2 |
| | Array (A2) | Not supported. |
| Array - A X B (A1) | Value (V2) | You have to input row and column of V1 and its value is assigned to V2 |
| | List (L2) | You have to input the column of V1 and its value is assigned to V2 |
| | Array (A2) | V1 overwrites V2 when V2 is A X B type. If V2 is one dimensional, you have to input the column. |

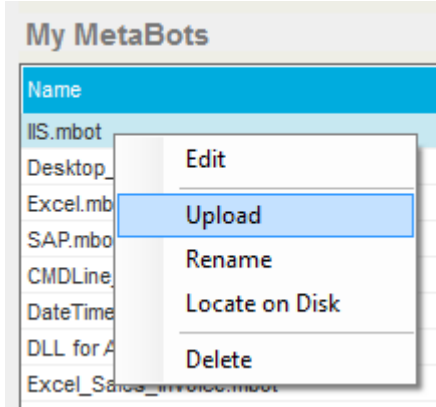
 Note: You can assign output variables to value type variables only.

Section: MetaBots in Enterprise Control Room

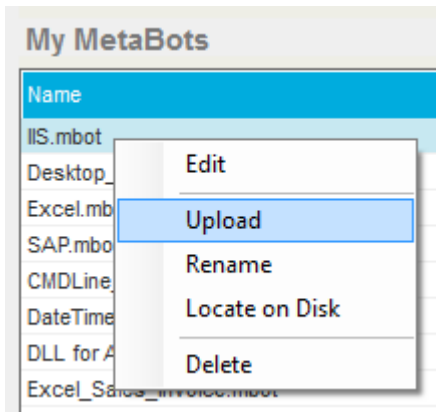
Uploading MetaBots to Control Room

MetaBots can be uploaded to the Control Room from where any number of Bot Creator Clients with MetaBot license can download and integrate them within automation tasks.

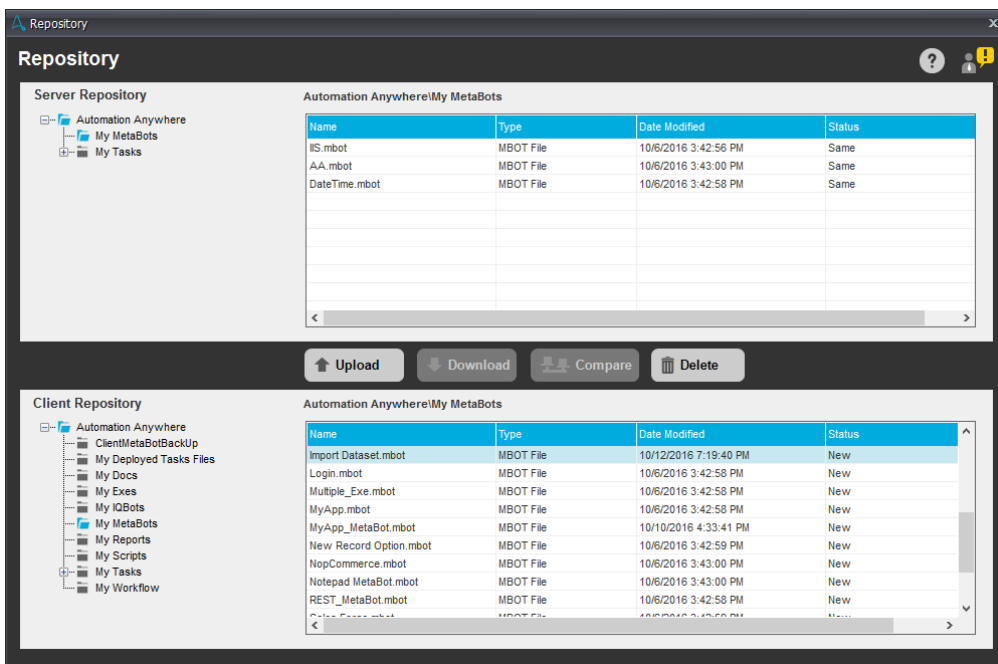
1. In the My MetaBots List, select (highlight) the MetaBot you want to upload.
2. Upload the Metabot using one of the following methods:
 - In the Actions button, select Upload from the drop-down list.



- Context click the MetaBot and select Upload.



- Go to Repository in Manager, select the MetaBot, and click on Upload.



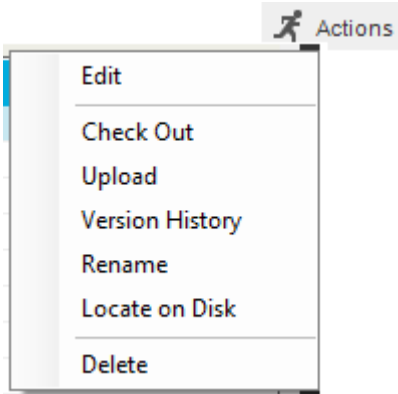


Tip: Ensure the MetaBot is closed in the MetaBots Client. Also, note that if the corresponding MetaBot **is the only MetaBot that is opened** in the MetaBots while uploading, you need to exit from the MetaBots Client.

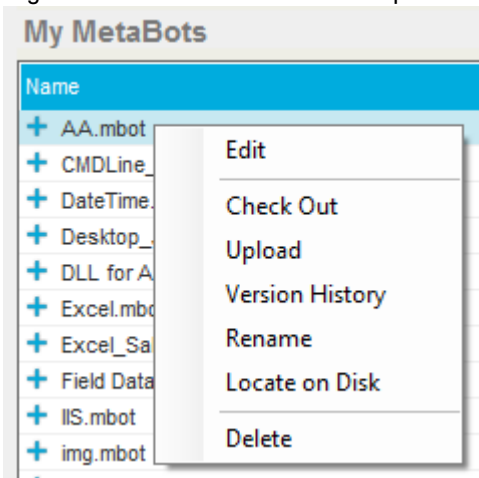
Uploading MetaBots when Version Control is Enabled

When Version Control is enabled, you can upload a MetaBot to the Control Room repository with comments, after using the 'Edit' option.

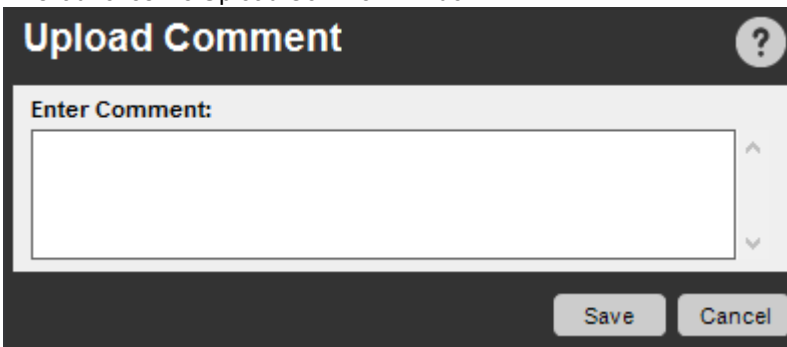
1. In the 'My MetaBots' List, select (highlight) the MetaBot you want to Edit and Upload.
2. Edit the Metabot using one of the following methods:
 - In the Actions button, select Upload from the drop-down list.



- Right click the MetaBot and select Upload.



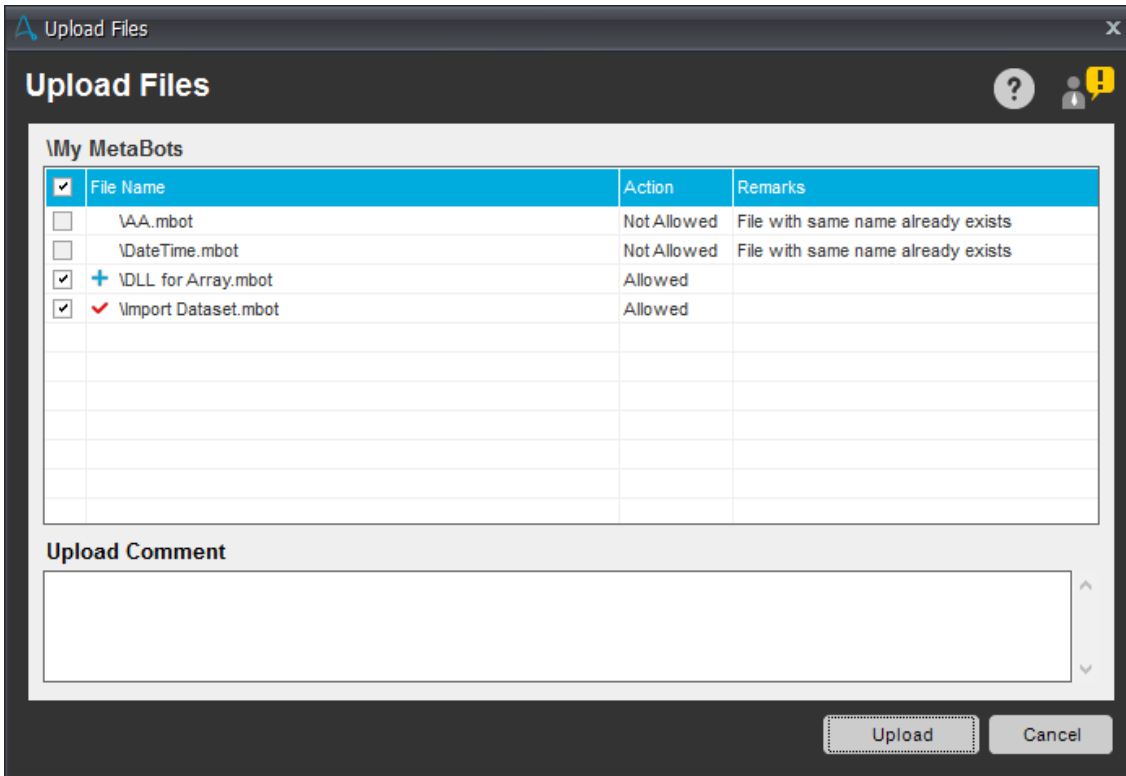
- This launches the Upload Comment window:





- Input your comments and click Save. This uploads the MetaBot to the Control Room.

Uploading Multiple MetaBots

1. Go to Manage > Repository.
2. Select the My MetaBots folder in the Client Repository list.
3. Select multiple MetaBots (use either ctrl or shift keys as required).
4. Click the Upload button.
5. In the Upload Files window, the MetaBots that are 'Allowed' to be uploaded are selected; if required, deselect the ones that you do not want to upload.



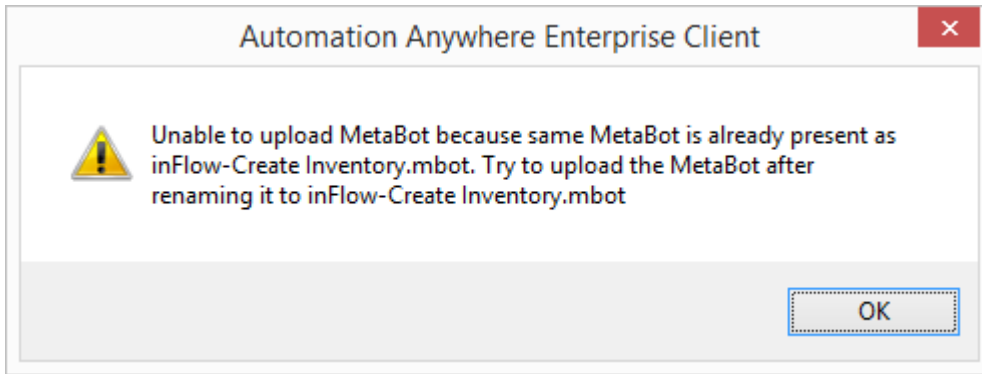
 **Tip:** No prefixed icon/sign denotes a successful upload. This ensures that the version history can be used as a reference point by the Client.

 **Note:** MetaBots marked 'Not Allowed' cannot be uploaded; the reason for which is depicted in the 'Remarks' column.

6. Add your 'Upload Comments' and click 'Upload'. These comments are applicable to all MetaBots that are uploaded.

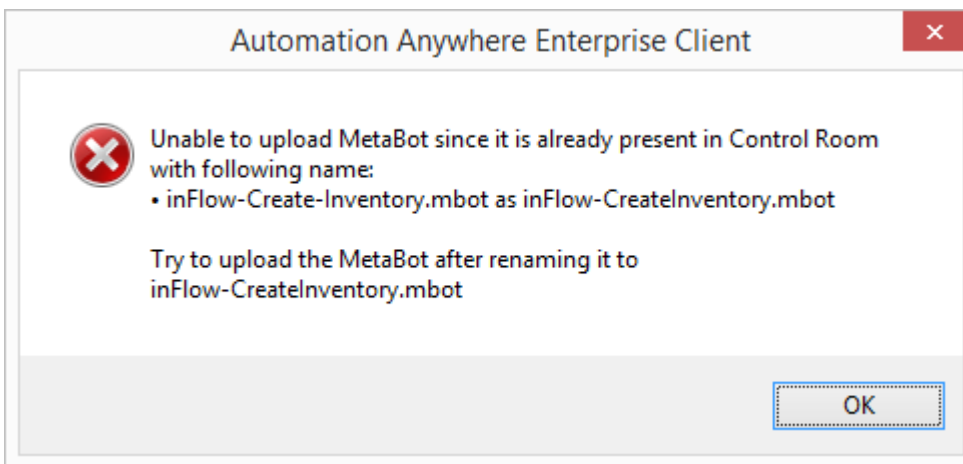
Uploading Renamed MetaBots

MetaBots that are already present in the Control Room repository and are renamed from a file system cannot be reuploaded to the Control Room repository. If you attempt to upload such a MetaBot from the **Client**, following message appears:

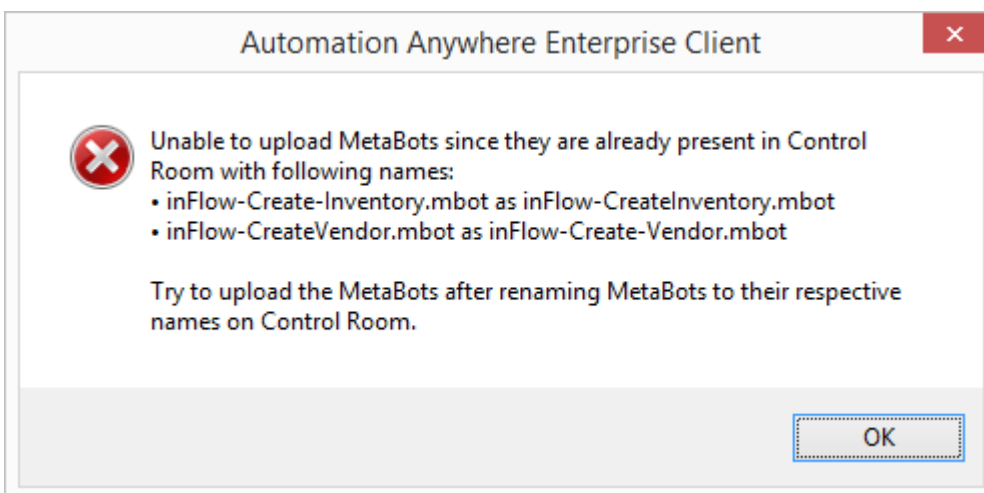


When you upload the renamed MetaBot(s) from the **Repository**, following messages appear:

- For a MetaBot:



- For multiple MetaBots:



You will have to revert to its original name in order to upload it.

Creating Roles and Assigning Permissions for MetaBots

You can define user roles and assign the necessary permissions for using MetaBots in the web based Control Room from the Administration → Roles page

Creating New Roles




To define new user roles, click

In the **Create Role** page, input the role name:

Select **View my bots** feature. This will enable the **Bots** panel below **Features** panel.


You can assign permissions based on the role you wish the user to play.

 **Note:** MetaBot users require a separate set of privileges to access the MetaBot files. Hence, the Control Room admin has to grant separate MetaBot Repository access rights at MetaBot file level, apart from the folder rights in **Which Bots and supporting files?** page.

Access Permissions for MetaBot Users

To grant privileges such as Execute, Upload, Download, and Delete, click the **MetaBots** tab beside the **TaskBots and Other Supporting Files** tab in the **Bots** panel. You can grant access privileges to a user at each MetaBot file level.

1. **Select All** - User can use the MetaBot in TaskBots, upload to Control Room, download to Client, and also delete the downloaded MetaBot.
2. **Execute** - User can use the Metabot in TaskBots but is not allowed to open the downloaded MetaBot in MetaBots Designer. This allows sharing of MetaBot as a Black-boxed bot. User cannot see/modify the content of a MetaBot.
3. **Upload** - User can upload MetaBots.
4. **Download** - User can download and modify MetaBots.
5. **Delete** - User can delete the downloaded MetaBots.

 Note: If you grant 'Download' permission to a user, the 'Execute' permission is automatically enabled so that a user can choose to **either** view/edit the content of a MetaBot that is downloaded **or** not all i.e. use it as a black-box.

You can choose to provide access rights based on the role that the user is allotted.

E.g. A BotCreator Client who has access to MetaBots can be granted all permissions whereas a BotRunner Client with MetaBots license can be granted only the 'Execute' permission.

You can verify whether the User has been created in the Roles list page. [Learn More](#)

Section: MetaBots in Enterprise Client

AI Sense - an overview

Automation Anywhere AI Sense enables intelligent automation of all the environments where object based automation is not available or is not reliable. Following type of applications are ideally suited for automation through AI Sense

Business applications which are exposed over Citrix (XenDesktop and XenApp)

Applications which are accessed over Remote Desktop Protocol (RDP)

Legacy applications such as Delphi etc.

AI Sense scans the application screen image and uses computer vision to identify all the UI elements (e.g. Labels and Text boxes). Post that, it automatically creates all the application UI objects from the image. All such automatically created objects are directly available for automation through MetaBot Logic. This saves a lot of time for user to manually create the objects on which automation is to be created.

Also it creates AI powered intelligent linking among the automatically created objects. This linking enables searching of one(child) object on the basis of other (parent) object. Intelligent linking ensures that automation continues to run reliably even when the absolute position or relative position of child and parents undergo modification.

MetaBot "Export Dataset" facilitates bulk data extraction from any application. This is available for standard technologies (e.g. HTML, .NET) and also for applications which are exposed as images, e.g. applications exposed over Citrix or RDP. For such applications, AI Sense ensures that a single scan of application is enough to extract all the objects visible on the screen. It greatly reduces the automation execution time by a factor of 5 to 10, depending on the number of fields (objects) being extracted.

Similarly, improved MetaBot "Import Dataset" facilitates bulk data entry into any application. For Citrix/RDP type of applications, it greatly reduces the automation execution time by a factor of 5 to 10, depending on the number of fields (objects) being entered.

For more information, follow the below mentioned articles:

- [Creating a MetaBot](#)
- [Adding and Recording a MetaBot](#)
- [Configuring MetaBot Screens](#)
- [Using Workbench to Create Logic](#)
- [Export Dataset command](#)
- [Import Dataset command](#)

Renaming MetaBots

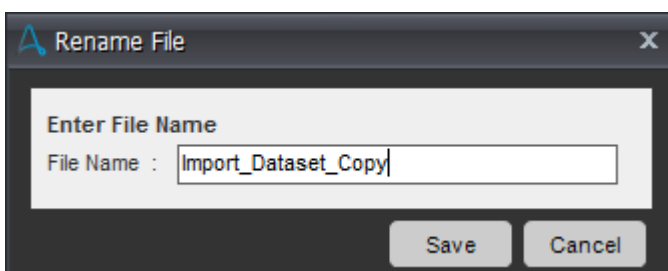
To rename a MetaBot, follow these steps:

1. In the 'My MetaBots' List that is available in the Automate tab of the Client, select (highlight) the MetaBot you want to rename.
2. Rename the MetaBot using one of the following methods:
 - In the File menu, select Rename.
 - In the Actions button, select Rename from the drop-down list.
 - Right click the MetaBot and select Rename.



Tip: Ensure the MetaBot is closed in MetaBot Designer. Also, note that if the corresponding MetaBot **is the only MetaBot that is opened** in the MetaBot Designer while renaming, you need to exit from MetaBot Designer.

3. In the Rename File window, specify a name for the MetaBot.



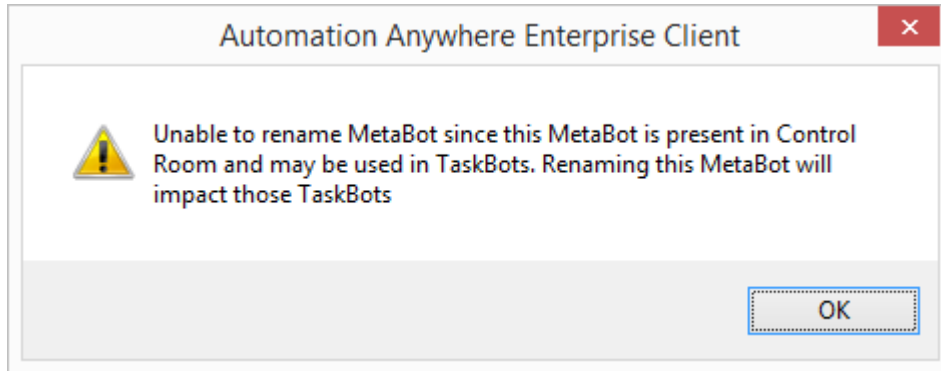
4. Click Save.


The renamed MetaBot is displayed in the My MetaBots List view.

Renaming Existing MetaBots

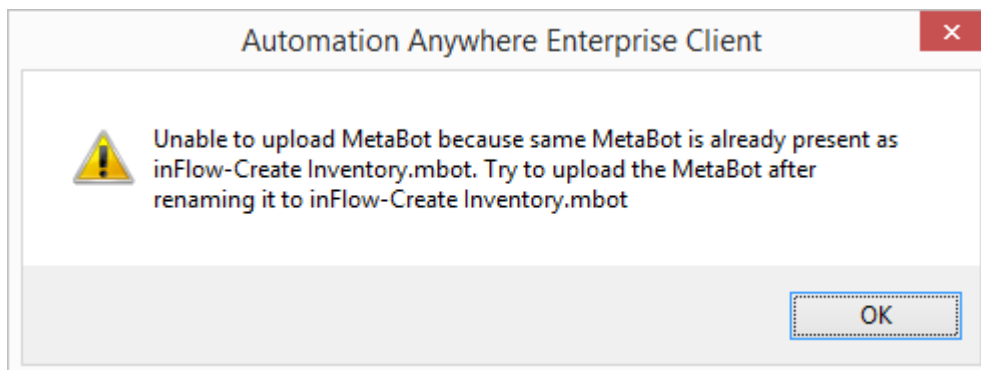
You can rename a MetaBot which is present only in your local machine and is not yet uploaded to Control Room. If the MetaBot is present in Control Room, system does not allow you to rename the MetaBot since it may be used in other TaskBots and renaming it may impact those TaskBots.

When you attempt to rename a MetaBot in the Client that already exists in the Control Room, following message appears:



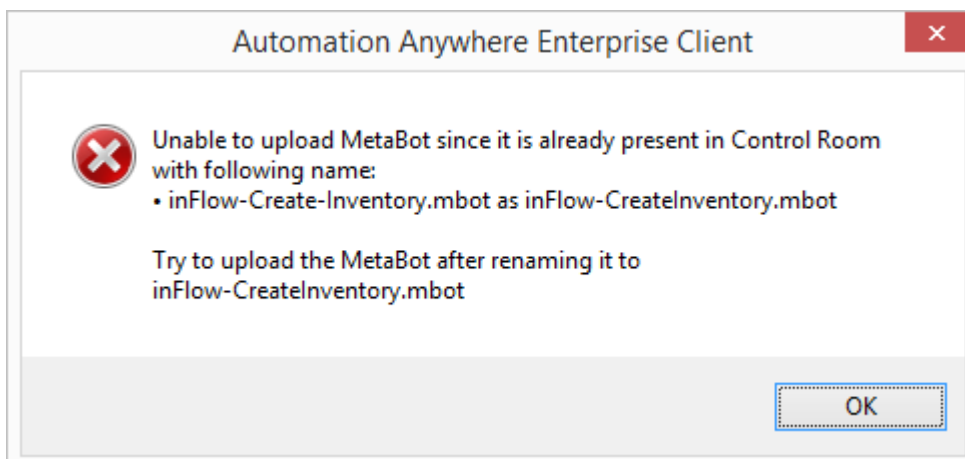
 Tip: It is recommended that you do not attempt to rename MetaBots from a file system.

MetaBots that are renamed from a file system cannot be uploaded again to the Control Room repository. If you attempt to upload such a MetaBot from the **Client**, following message appears:

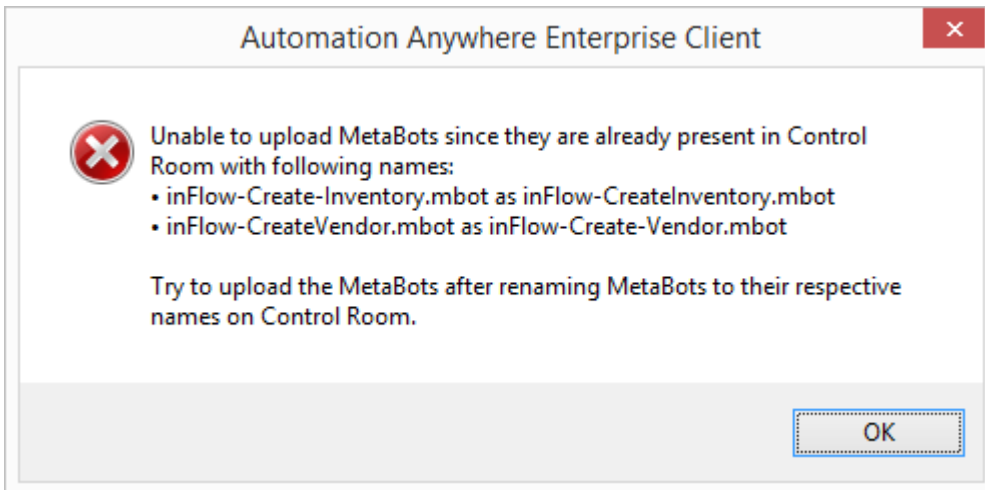


When you upload the renamed MetaBot(s) from the **Repository**, following messages appear:

- For a MetaBot:



- For multiple MetaBots:

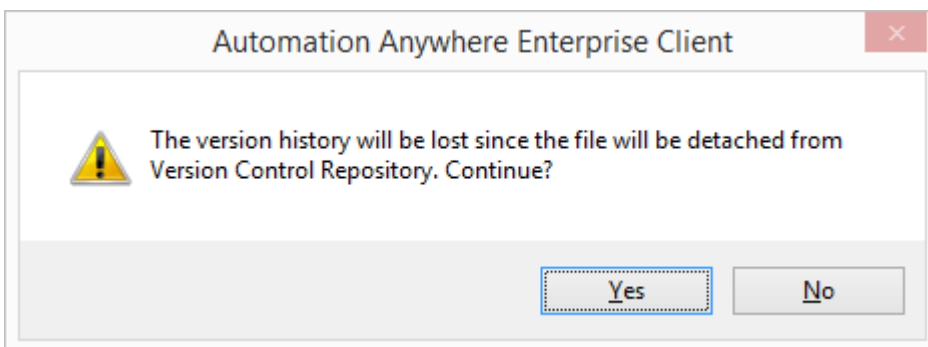


You will have to revert to its original name in order to upload it.

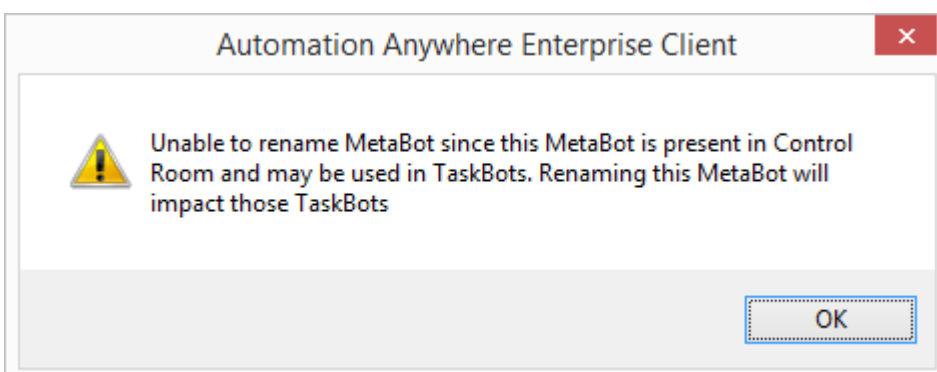
Renaming MetaBots if Version Control is enabled


You can rename a MetaBot in the local repository. However, note that if you rename a MetaBot and upload to the repository, it will overwrite the existing MetaBot with the previous name. The version history will be lost in such case.

- When renaming a file that has been uploaded to the server repository, you will be prompted:

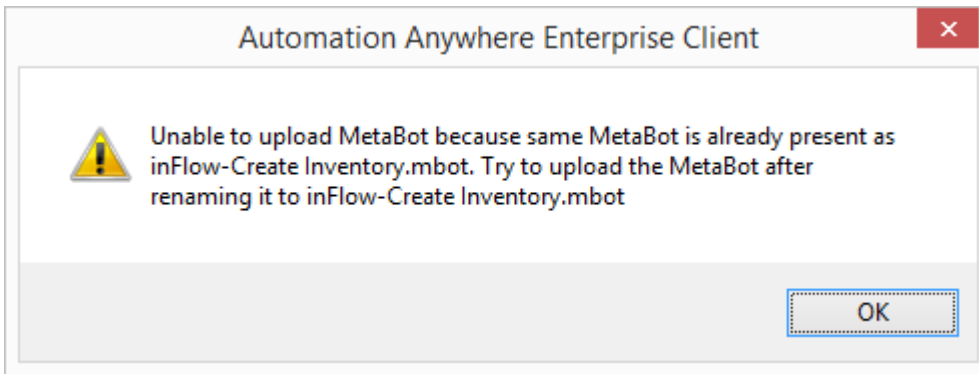


- If the file is checked out for edit, you will be prompted a message as only the local MetaBot version will be renamed :




 Tip: Undo Check out to rename the MetaBot.

- If you try to upload a MetaBot that was renamed from a file system, you are shown:




Deleting MetaBots

In some cases, you might want to delete an existing MetaBot.

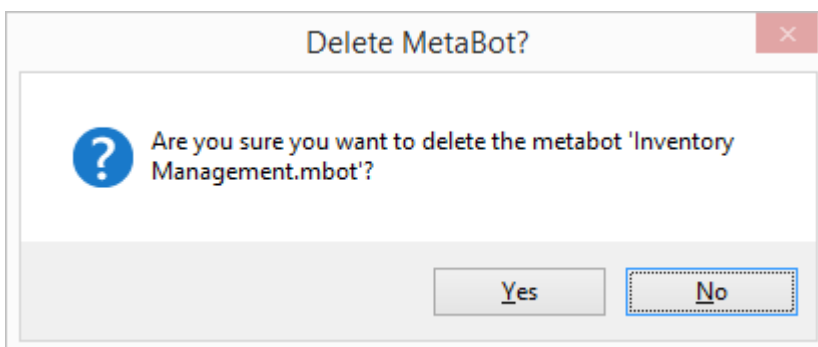
 Tip: It is recommended that you do not attempt to delete MetaBots from a file system.

To delete a MetaBot, follow these steps:


1. In the 'My MetaBots' List, select (highlight) the MetaBot you want to delete.
2. Delete the Metabot using one of the following methods:
 - In the Edit menu, select Delete.
 - In the Actions button, select Delete from the drop-down list.
 - Right click the MetaBot and select Delete.


 Tip: Ensure the MetaBot is closed in MetaBot Designer. Also, note that if the corresponding MetaBot **is the only MetaBot that is opened** in the MetaBot Designer while deleting, you need to exit from MetaBot Designer.

3. In the **Delete MetaBot?** window, click on Yes.



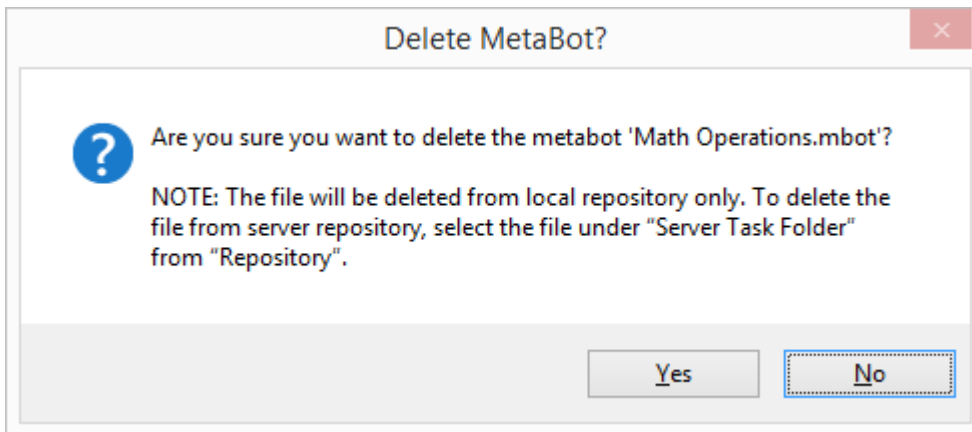
- The MetaBot is removed from the 'My MetaBots' List view.


 Note: After deleting a MetaBot, you cannot restore it in the Client. Before deleting a MetaBot, ensure you no longer have use for that MetaBot.

 Tip: As a best practice, ensure all MetaBots are copied to the Control Room repository as a way of backing up your MetaBots.

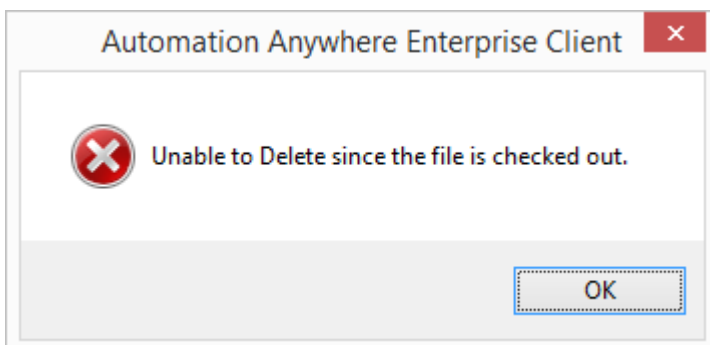
Deleting a MetaBot if Version Control is enabled

If you have version control enabled, while deleting the MetaBot from the Client, apart from confirmation, you will be notified that it will be deleted locally only.



 Note: To delete the MetaBot from the Control Room repository, you will have to select it from the relevant folder of the repository.

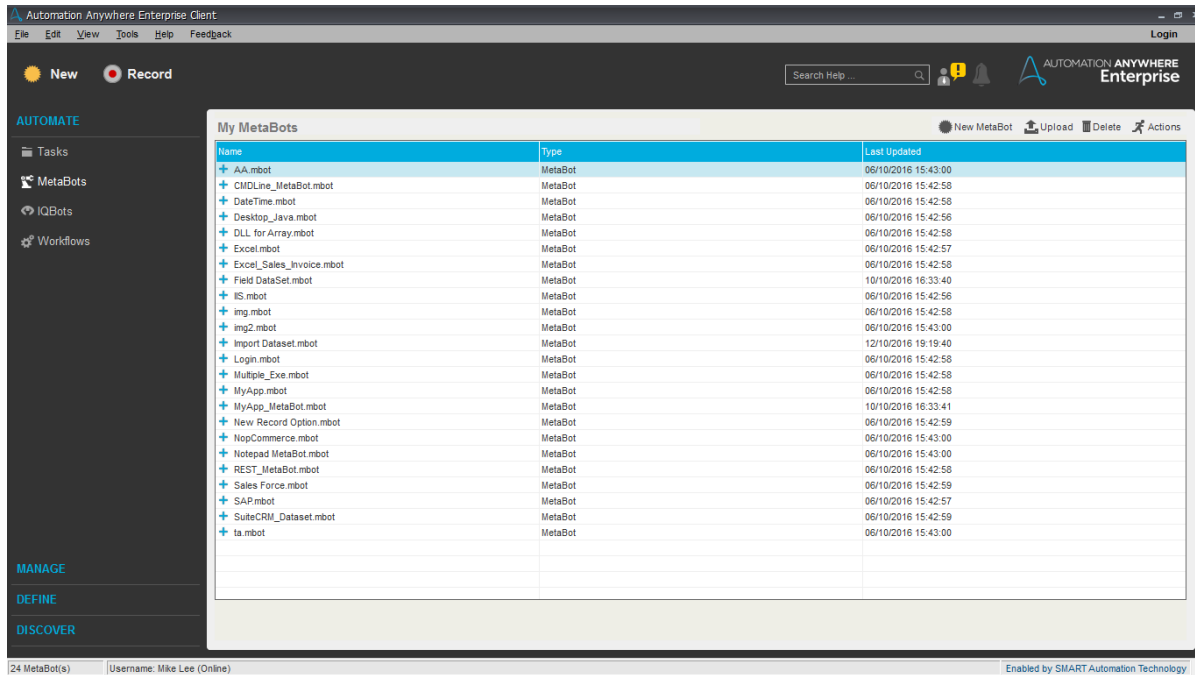
If the MetaBot is checked out, you will not be allowed to delete it.



Enabling Version Control

The MetaBot Designer has an integrated feature on Version Control that allows MetaBot users to manage various versions of MetaBots and enforce controlled edits.

Note: Automation Anywhere supports Subversion v1.8.13 and v1.8.14 with Visual SVN Server 3.3.x



(Image as seen in Enterprise Client - MetaBot Designer version 10.3)

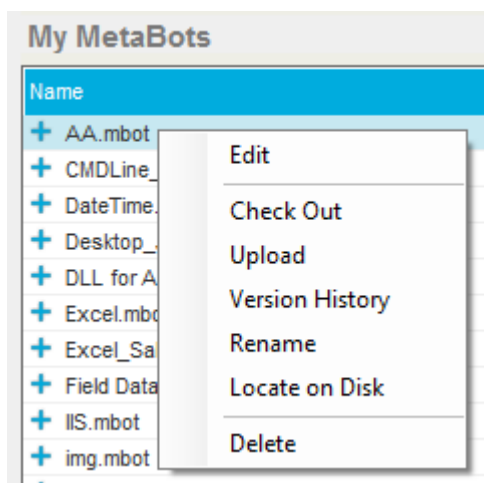
Control the versioning of MetaBots in Clients by enabling the feature from Automation Anywhere Enterprise Control Room.

[Learn More](#)

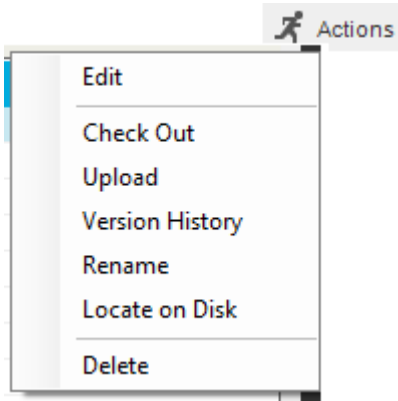
Using Version Control in MetaBot

To perform controlled edits to your MetaBots, use versioning to create new files, check out for edit, upload with comments and view the version history.

You can context click the MetaBot in the 'My MetaBots' list view of the Automation Anywhere Enterprise Client.



You can also perform similar operations using Actions.



Tip: Ensure the MetaBot is closed in the MetaBot Designer Client. Also, note that if the corresponding MetaBot **is the only MetaBot that is opened** in the MetaBot Designer while performing these operations, you need to exit from the MetaBot Designer Client.

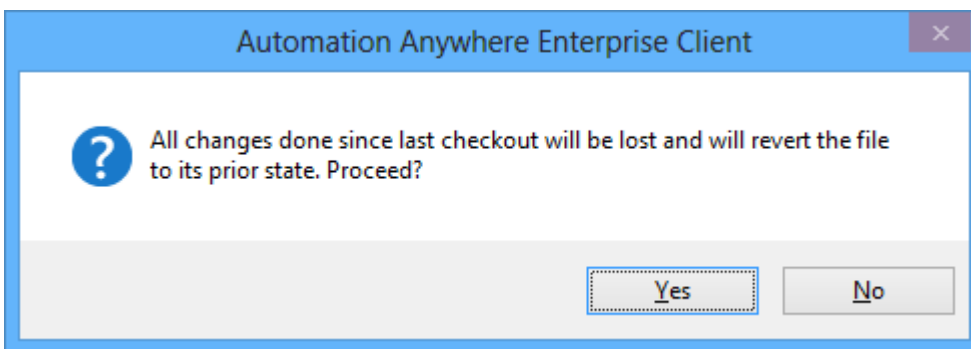
Perform the operations detailed below using either context click or 'Actions'.

1. **Edit:** You can edit a MetaBot from the 'My MetaBots list' provided it has been 'Uploaded' and 'Checked Out'. On editing, you are guided to the 'Assets' view of the selected MetaBot in the MetaBot Designer. Here, you can opt to add/record screens, dll's, and add/update Logic(s).

A new MetaBot created in the local repository (of the Client) has a plus sign (+) appended to the MetaBot name. Refer the article on '[Creating a MetaBot](#)' to learn how to create a new MetaBot.

2. **Check Out:** A MetaBot that already exists in the Control Room repository can be checked out for editing. A check mark (✓) indicates the file is checked out for editing.

The option toggles to 'Undo Checkout' once the MetaBot is checked out for editing. This enables you to undo the last update(s) to the checked out file. Select 'Yes' to confirm:



3. **Upload:** Post editing, you can upload a MetaBot to the Control Room repository with comments. No prefixed icon/sign denotes a successful upload. This ensures that the version history can be used as a reference point by the Client.



You can also opt to upload MetaBot(s) from the My MetaBots folder in the Repository. [Learn More](#)

4. **Version History:** You can view revisions to a MetaBot and if required, roll back any updates. It allows you to identify the user and relevant 'Action' that was performed during a specific 'Date and time' with relevant check in 'Comments'. [Learn More](#)
5. **Rename:** You can rename a selected MetaBot in the local repository provided it has not been checked out. Note that if you rename a MetaBot and upload to the repository, it will overwrite the existing MetaBot with the previous name. The version history will be lost in such case. [Learn More](#)

6. **Locate on Disk:** In some cases, you might want to locate the MetaBot that is associated with an automated task you've created.

Automation Anywhere MetaBot files have the file extension of: '.mbot'

To locate a MetaBot on your computer, follow these steps:

1. In the main Automation Anywhere window, select the MetaBot in the My MetaBots List for which you want to locate the .mbot file.
 2. Either click on the Edit menu or on the Actions button, and select 'Locate on Disk'. The MetaBot files are listed in the File Explorer.
 3. Locate the automation MetaBot file. The .mbot name matches the name you've assigned to the MetaBot.
7. **Delete:** You can delete a MetaBot from the local repository provided it has not been checked out. [Learn More](#)

Viewing Version History

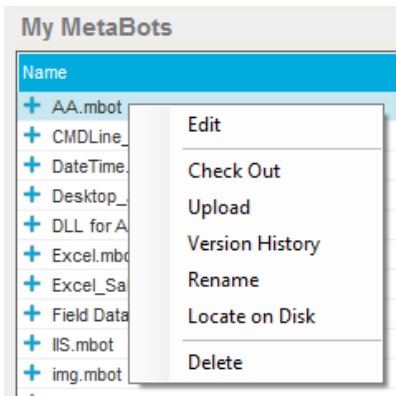
Use Version History to view the history of updates to the selected MetaBot. Also roll back updates, if required.

On selecting Version History, you will be able to view a list of all revisions created for the selected MetaBot.

Viewing history of file versions

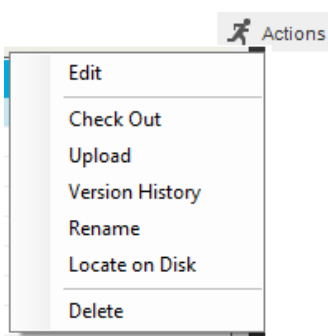
You can view 'Version History' of the selected MetaBot from:

1. The Context Menu:

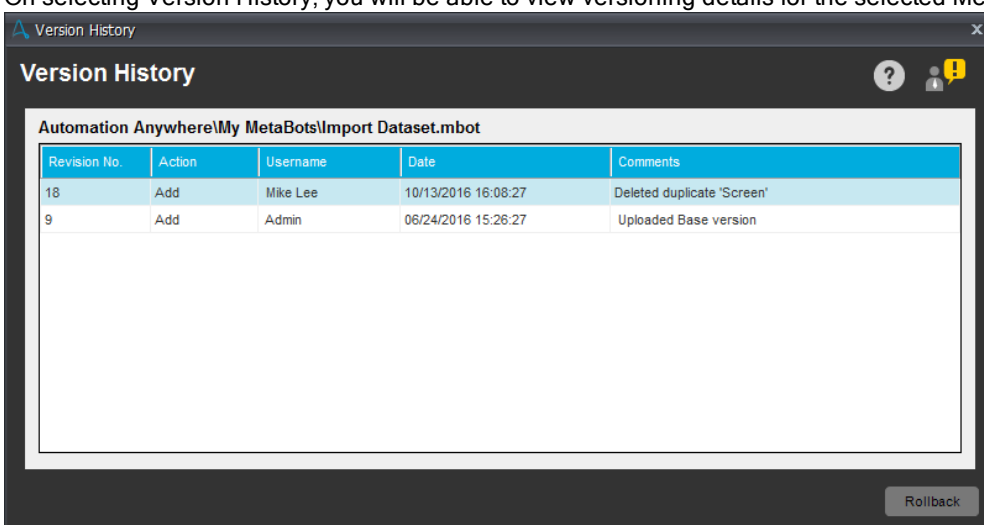


OR

2. Actions list:



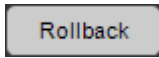
3. On selecting Version History, you will be able to view versioning details for the selected MetaBot:




The Version History is displayed in descending order of the time-stamp i.e. the latest revision at the top and the first revision at the bottom. It allows you to identify the user and relevant 'Action' that was performed during a specific 'Date and time' with relevant check in 'Comments'.

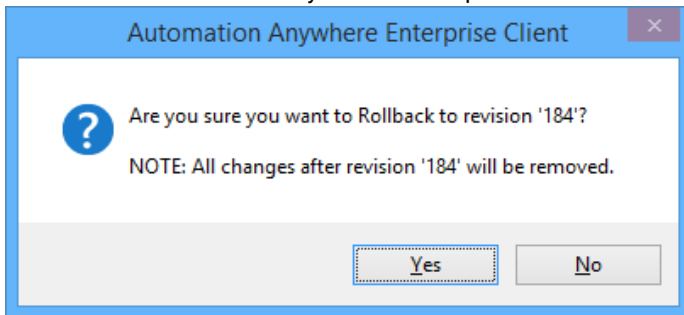
Rollback Updates

You can also rollback the updates to a specific revision. Use this to revert updates/changes in the selected MetaBot to the selected revision from the version history.

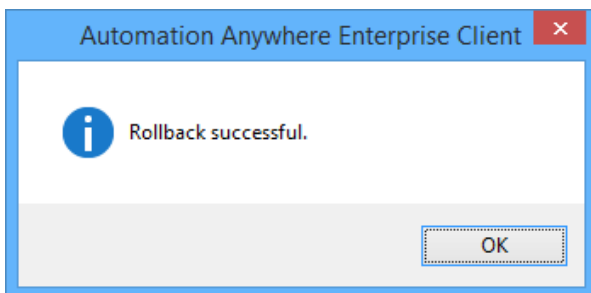



 Tip: Ensure MetaBot Designer client is closed before using Rollback.

Select the revision to which you want the updates to be rolled back and confirm:



On confirming, all the changes done since the selected revision to the latest revision will be rolled back and a success message is displayed:



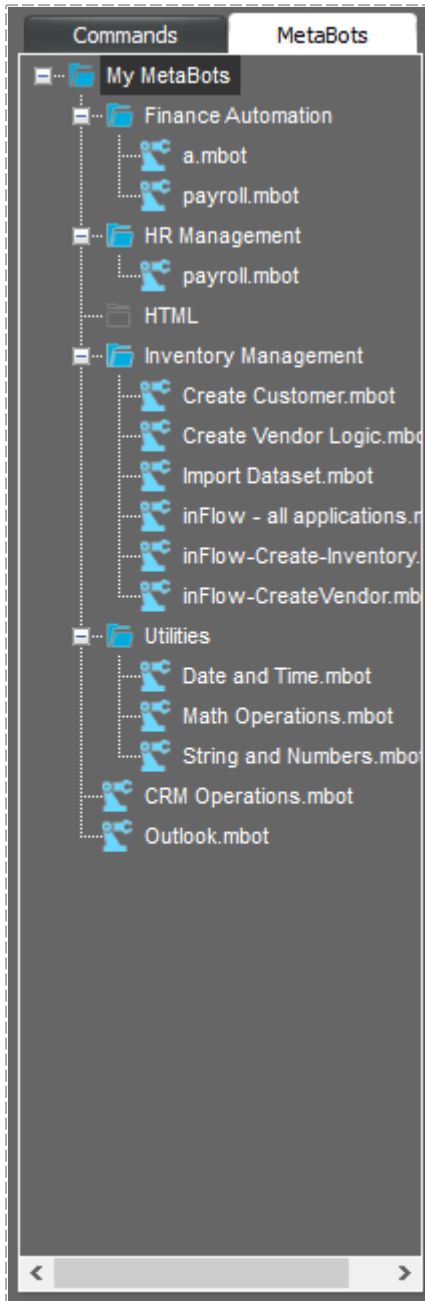
 Note: It is recommended you check in (upload) the file once it is rolled back to the selected revision so that the latest revision(s) are reflected in the updates.

Using MetaBot Logic in TaskBots and MetaBot Logics

You can integrate Logic in your TaskBots as well as other MetaBot Logics in the Automation Anywhere Workbench to create automation. Based on your work-flow, you can add either one or more Logic from a single MetaBot or multiple Logic from various MetaBots.

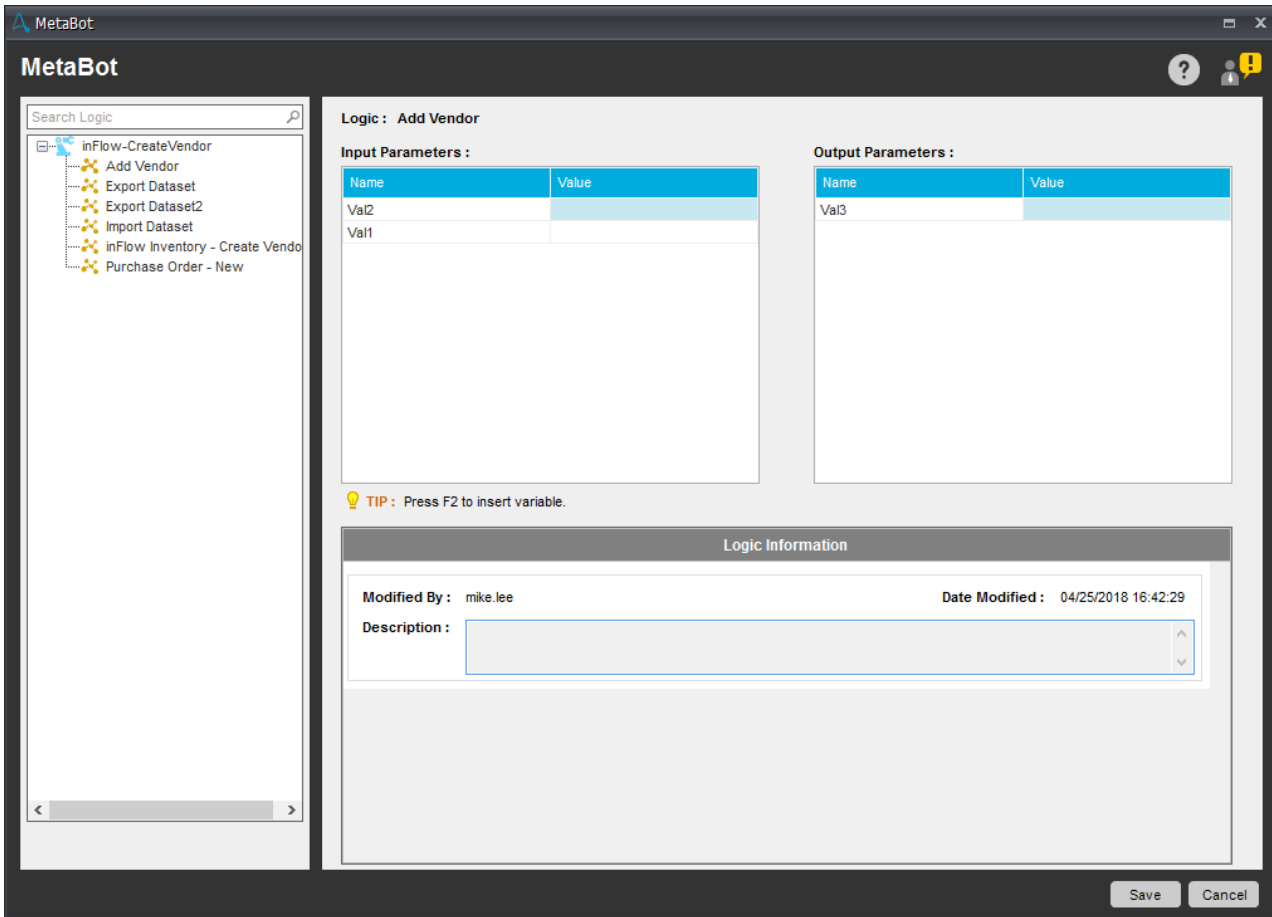
Adding MetaBot Logic to a TaskBot

1. To begin, click on the MetaBot section given at the top of the command panel in the Workbench




2. Double click/drag and drop a MetaBot from the list of MetaBots.

3. The **MetaBot** window is launched.




4. Select the desired logic from the list. If this logic needs certain values and/or variables as input and/or output, they get listed under the **Input** and/or **Output** Parameters pane. Otherwise, the parameter tables appear empty.

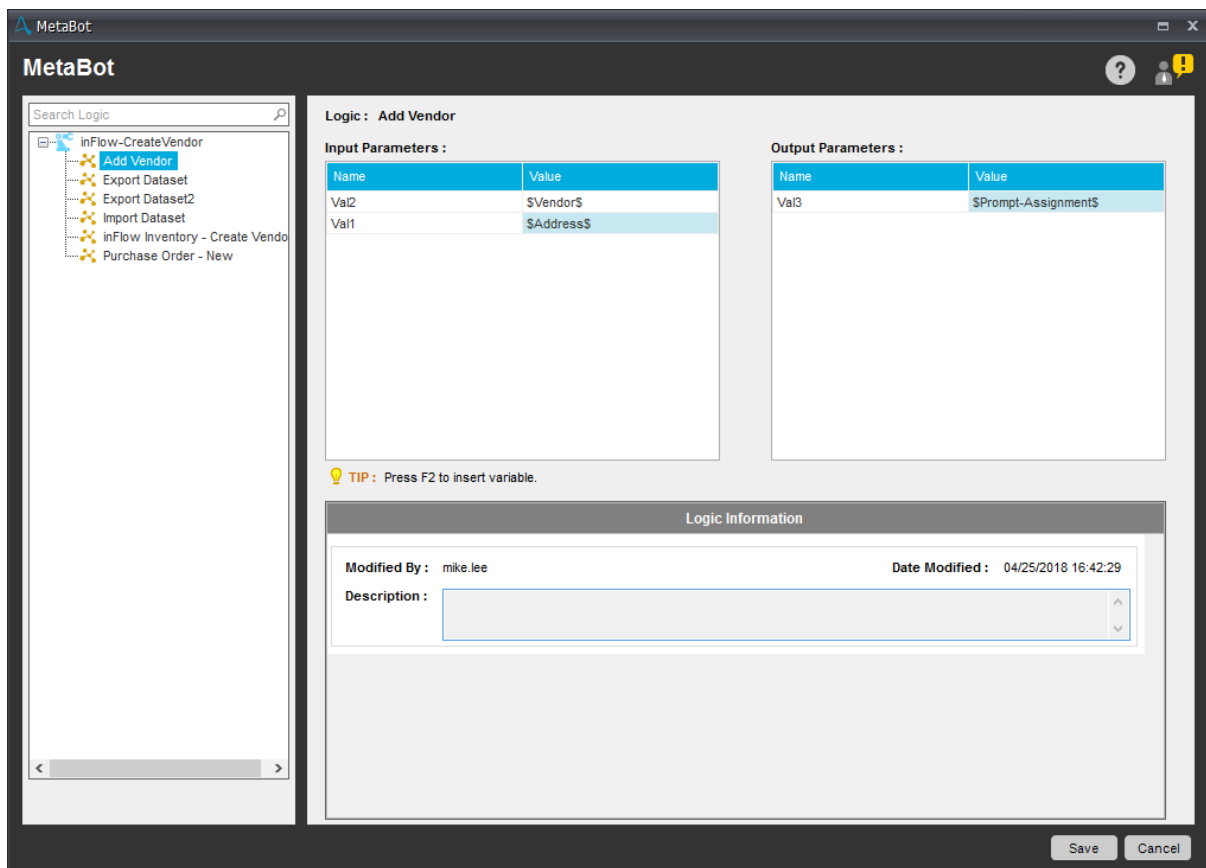
- If Input and InputOutput parameter type variables are available in the Logic, you can assign a value or variable required as an input in **Input Parameters** pane.
- Similarly, if Output and InputOutput parameter type variables are available in the Logic, you can assign another variable as an output in **Output Parameters** pane.
- Refer [Variables - Parameter Type](#) and [Passing Parameters in Automation](#) for details.

 Tip: Use the 'Search Logic' facility to navigate to the Logic you want.


5. If you have assigned variables in the logic, you will have to assign the required set of values to the Input and Output Parameters.


- You cannot leave these blank.
- You can also assign variables to the Input and/or Output Parameters from the list of variables available in the task.

 Tip: If the Input and/or Output parameters require use of credentials or sensitive data, you can opt to use 'Credential Variables'. Select the required variable listed under \$CredentialVariables\$ from Insert Variables. You cannot, however, pass these variables from one TaskBot/MetaBot Logic to another TaskBot/MetaBot Logic. Refer the article [Assigning Credential Variables](#) for details.




6. The **Description** field provides information about the Logic that is being configured. The name of the user who updates the Logic Information and the date and time it was modified is also displayed. Similarly, you can refer the **Input** and **Output** parameter Description below the Logic description. This is added when you are creating variables. This is highly useful as it provides contextualized help without any clicks, while using any the MetaBot Utilities/Logics in TakBots.

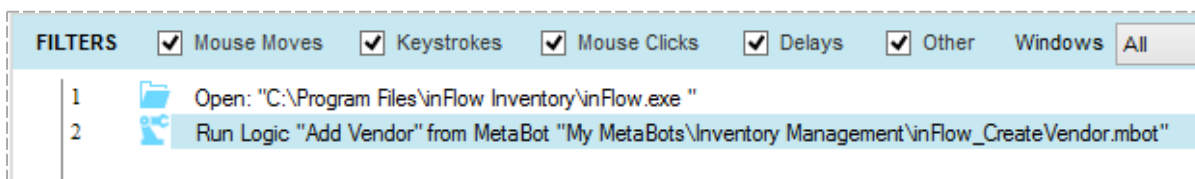
 **Tip:** If you want to view the description without scrolling, hover your mouse over the Description field and the entire text is shown as tool tip.

 **Note:** If your Logics and Input/Output parameters were created in 10.x versions, by default the description is not shown. However, you can update the Logic description and that is shown once it is edited and saved using the Properties option. Similarly, you can edit the variables to include the parameter description.

7. Once you are done, click **Save** to include the MetaBot Logic in your TaskBot.

 **Note:** During TaskBot execution, if your target application is open in admin mode, you will have to run the task in admin mode.

The Workbench depicts the event data for Logic:



Import DataSet Command

Inserting huge amounts of data from a single source can be done using the Import Dataset command in a MetaBot Logic from the Workbench. You can insert data into various fields from an external file such as an Excel spreadsheet using single command. It enables you to create logic that would otherwise involve numerous keystrokes and clicks.

Supported Technologies and Controls

Import Dataset command supports the following technologies:

1. HTML
2. JAVA
3. MSAA
4. .NET
5. Flex

Import Dataset command supports the following controls and actions:

1. **Static Text (label)** - GetProperty, Click, LeftClick, RightClick, and DoubleClick
2. **Text Box** - SetText, AppendText, Click, LeftClick, RightClick, and DoubleClick
3. **Radio Button** - Select, LeftClick, RightClick, and DoubleClick
4. **Check Box** - Check, Uncheck, Toggle, LeftClick, RightClick, and DoubleClick
5. **Combo Box** - SelectItemByText, SelectItemByIndex, LeftClick, RightClick, DoubleClick, and Expand
6. **List View** - SelectItemByText, SelectItemByIndex, LeftClick, RightClick, and DoubleClick
7. **Buttons** - LeftClick

Import Data to an application

You can configure the Import Dataset command from Workbench → Commands.

1. Create a Screen using required application. [Learn More](#)
2. Configure the Screen for required parameters. [Learn More](#)



Tip: If an object does not display or displays an unidentifiable text and/or number, its recommended you provide an alias to that object during Screen configuration. [Learn More](#)

3. In the Logic Editor, select Import Dataset from the Commands list.
4. Start filling in details using a combination of variables, text and controls. Unless a field is populated, you cannot move to the next.



Note: All the objects which are not selected for automation execution are disabled by default. To enter value in those, click on the check-box in **Write** column.

Import Dataset

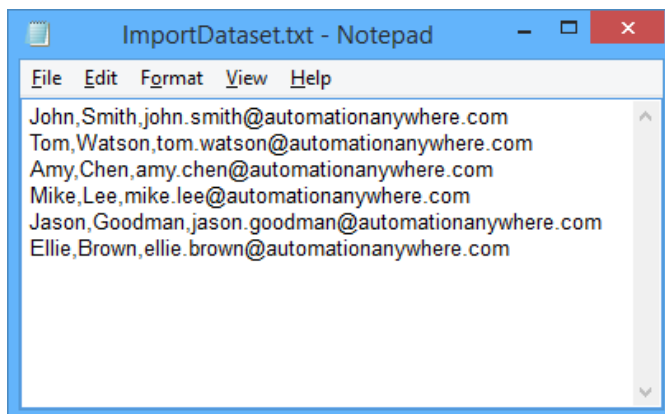
Select 'Write' to fill object value during automation execution [Show Selected](#)

| # | Entry Name | Type | Value | Write |
|----|---------------------|----------|---------------------|-------------------------------------|
| 1 | Vendor | TextBox | \$Excel Column(1)\$ | <input checked="" type="checkbox"/> |
| 2 | Contact Name | TextBox | \$Excel Column(2)\$ | <input checked="" type="checkbox"/> |
| 3 | Phone | TextBox | \$Excel Column(3)\$ | <input checked="" type="checkbox"/> |
| 4 | Website | TextBox | \$Excel Column(4)\$ | <input checked="" type="checkbox"/> |
| 5 | Fax | TextBox | \$Excel Column(5)\$ | <input checked="" type="checkbox"/> |
| 6 | NAMELESS | TextBox | | <input type="checkbox"/> |
| 7 | NAMELESS | TextBox | | <input type="checkbox"/> |
| 8 | NAMELESS | TextBox | | <input type="checkbox"/> |
| 9 | Credit | TextBox | \$0.00 | <input type="checkbox"/> |
| 10 | Balance | TextBox | \$0.00 | <input type="checkbox"/> |
| 11 | Currency | ComboBox | US Dollar (\$) | <input type="checkbox"/> |
| 12 | Currency | TextBox | US Dollar (\$) | <input type="checkbox"/> |
| 13 | Currency | ListView | | <input type="checkbox"/> |
| 14 | Carrier | ComboBox | | <input type="checkbox"/> |
| 15 | Carrier | TextBox | | <input type="checkbox"/> |
| 16 | Carrier | ListView | | <input type="checkbox"/> |
| 17 | Emergency Phone | TextBox | | <input type="checkbox"/> |
| 18 | NAMELESS | TextBox | | <input type="checkbox"/> |
| 19 | Email | TextBox | | <input type="checkbox"/> |
| 20 | Alternate Contact 3 | TextBox | | <input type="checkbox"/> |

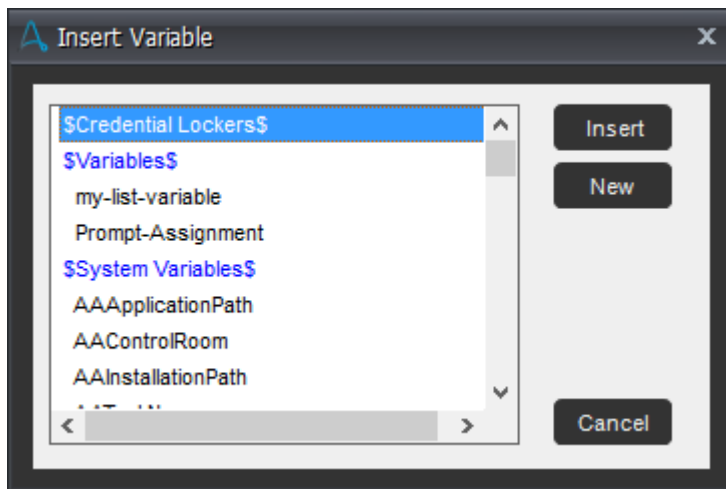
Click 'Unlock' icon to change the execution order

Press F2 to insert variable. Save Cancel

- You can use either the direct assignment of variable i.e. assign a Value or indirect assignment i.e. assign value from a text document as shown:



- For variables that make use of credentials or any sensitive data, use Credential Variables. Select the required variable listed under \$CredentialVariables\$ from Insert Variables. Refer the article [Assigning Credential Variables](#) for details.
- You can choose to create a variable, if not present when inserting one. In the Value field, press function key F2 to launch Insert Variable dialog. Click **New** to create a variable.



- You can also insert object values using the properties window. To configure additional properties such as delay, search criteria etc, double click anywhere on the corresponding object row and update the properties.

| # | Entry Name | Type | Value | Write |
|---|------------|---------|-------|-------------------------------------|
| 1 | Name | TextBox | John | <input checked="" type="checkbox"/> |
| 2 | Credi | Name | | <input checked="" type="checkbox"/> |
| 3 | Balan | | | <input checked="" type="checkbox"/> |

Select Action : SetText

Keystrokes
 Mask

Enter Text : John

Advanced Options

Wait for the object to exist: 15 seconds



Show Object Properties

| Name | Value |
|-----------|---------------------|
| Object ID | txtVendorName |
| Name | Name |
| Value | |
| Class | WindowsForms10.E... |

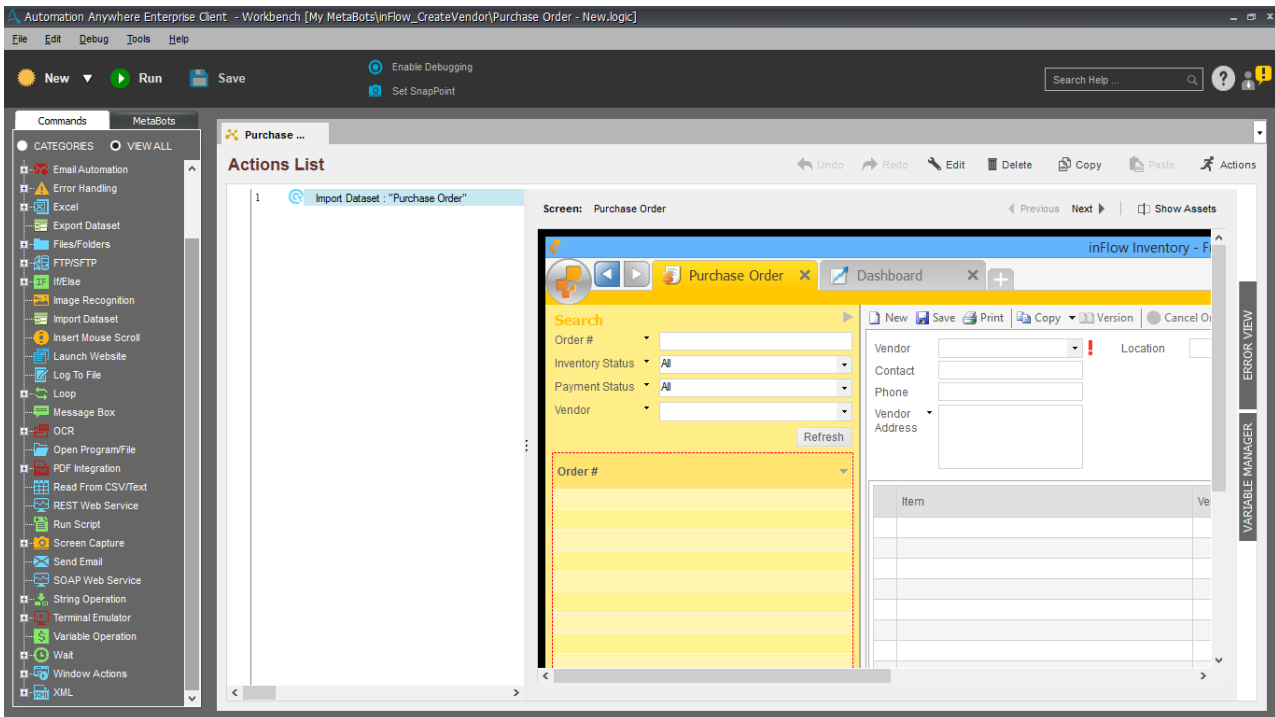
TIP : Press F2 to insert variable

Update


- i. **Select Action** that needs to be performed during Logic execution. This can be changed if required.
 - ii. Optionally **Insert Keystrokes** and/or **Mask** the keystrokes
 - iii. Type the required text in **Enter Text**
 - iv. In **Advanced Options** specify:
 - a. **Wait time** for the object to exist. This can be changed if required.
 - b. Optionally view the **Object Properties** which will be used to execute the command
 - v. Save or **Update** if in edit mode.
- By default, all fields are displayed. Once you select fields for inserting values, and you open in edit mode after saving only those are shown. To view all click **Show All**.

- You can also choose to change the sequence in which the object values will be filled when automation is executed. For this you must click . This is the default state. To lock the sequence, click . This ensures the sequence is not changed.
- The data that is visible can be filtered on **Type** and sorted on **Type** and **Entry** columns.

5. Your logic will reflect the Import Dataset command in the event data.



6. Save.

 **Tip:** The **Save** button is disabled until all object values that are selected for **Write** are filled.

It is a best practice to Calibrate Screens that are used for Import Dataset command if the application is subject to frequent updates. These would affect the MetaBot during execution.

 **Note:** You can fill the form in OCR mode, IR mode and mixed mode.

Export Dataset Command

Extracting huge amounts of data from a single source can be done using the Export Dataset command in a MetaBot Logic from the Workbench. You can read data from various fields from source and save it into an external file such as an Excel spreadsheet using single command. It enables you to create logic that would otherwise involve numerous keystrokes and clicks.

Supported Technologies and Controls

Export Dataset command supports the following technologies:

1. HTML
2. JAVA
3. MSA
4. .NET
5. Flex

Export Dataset command supports the following controls and actions:

1. **Static Text (label)** - GetProperty
2. **Text Box** -GetProperty
3. **Radio Button** - GetStatus and GetProperty
4. **Check Box** - GetStatus and GetProperty
5. **Combo Box** - GetProperty, GetTotalItems, GetSelectedIndex, and GetSelectedText
6. **List** - GetProperty, GetTotalItems, GetSelectedIndex, and GetSelectedText
7. **Buttons** - LeftClick

Export Data from an application

You can configure the Export Dataset command from Workbench → Commands.

1. Create a Screen using required application. [Learn More](#)
2. Configure the Screen for required parameters. [Learn More](#)



Tip: If an object does not display or displays an unidentifiable text and/or number, its recommended you provide an alias to that object during Screen configuration. [Learn More](#)

3. In the Logic Editor, select Export Dataset from the Commands list.
4. Start filling in details using a combination of variables, text and controls. Unless a field is populated, you cannot move to the next.



Note: All the objects which are not selected for automation execution are disabled by default in the Value column.

To enable editing of Value parameters, select **Read**.

Export Dataset

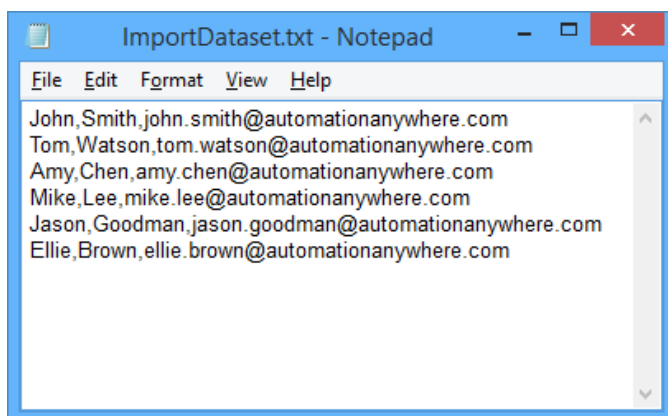
Select 'Read' to extract object value during automation execution [Show Selected](#)

| # | Entry Name | Type | Value | Read |
|----|---------------------|------------|----------------------|-------------------------------------|
| 1 | Order # | StaticText | \$Order-no\$ | <input checked="" type="checkbox"/> |
| 2 | Inventory Status | StaticText | \$Inventory-Status\$ | <input checked="" type="checkbox"/> |
| 3 | Payment Status | StaticText | \$Payment-Status\$ | <input checked="" type="checkbox"/> |
| 4 | Vendor | StaticText | \$Vendor-Name\$ | <input checked="" type="checkbox"/> |
| 5 | Vendor Address | StaticText | \$Vendor-Address\$ | <input checked="" type="checkbox"/> |
| 6 | Date | StaticText | \$Date\$ | <input checked="" type="checkbox"/> |
| 7 | Due Date | StaticText | \$Due-Date\$ | <input checked="" type="checkbox"/> |
| 8 | Total | TextBox | \$Total\$ | <input checked="" type="checkbox"/> |
| 9 | Balance | TextBox | \$Balance\$ | <input checked="" type="checkbox"/> |
| 10 | Sub-Total | TextBox | \$Sub-Total\$ | <input checked="" type="checkbox"/> |
| 11 | Phone | TextBox | \$Phone-no\$ | <input checked="" type="checkbox"/> |
| 12 | Contact | TextBox | \$Contact-Name\$ | <input checked="" type="checkbox"/> |
| 13 | Search | StaticText | | <input type="checkbox"/> |
| 14 | Order # | TextBox | | <input type="checkbox"/> |
| 15 | All | TextBox | | <input type="checkbox"/> |
| 16 | All | TextBox | | <input type="checkbox"/> |
| 17 | NAMELESS | TextBox | | <input type="checkbox"/> |
| 18 | Unfulfilled, Unpaid | TextBox | | <input type="checkbox"/> |
| 19 | NAMELESS | TextBox | | <input type="checkbox"/> |
| 20 | Order # | StaticText | | <input type="checkbox"/> |

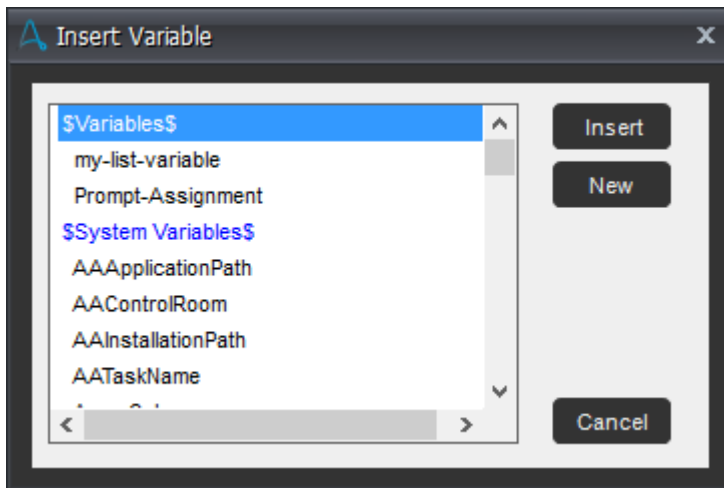
Click 'Unlock' icon to change the execution order

Press F2 to insert variable. Save Cancel

- You can use either the direct assignment of variable i.e. assign a Value or indirect assignment i.e. assign value from a text document as shown:



- You can choose to create a variable, if not present when inserting one. In the Value field, press function key F2 to launch Insert Variable dialog. Click **New** to create a variable.



- Value, Random, Array, and List type variables are supported in Export Dataset





Tip: Define an Array type variable to export 'Values' that can be filled multiple times for a single 'Entry Name'. [Learn More](#)

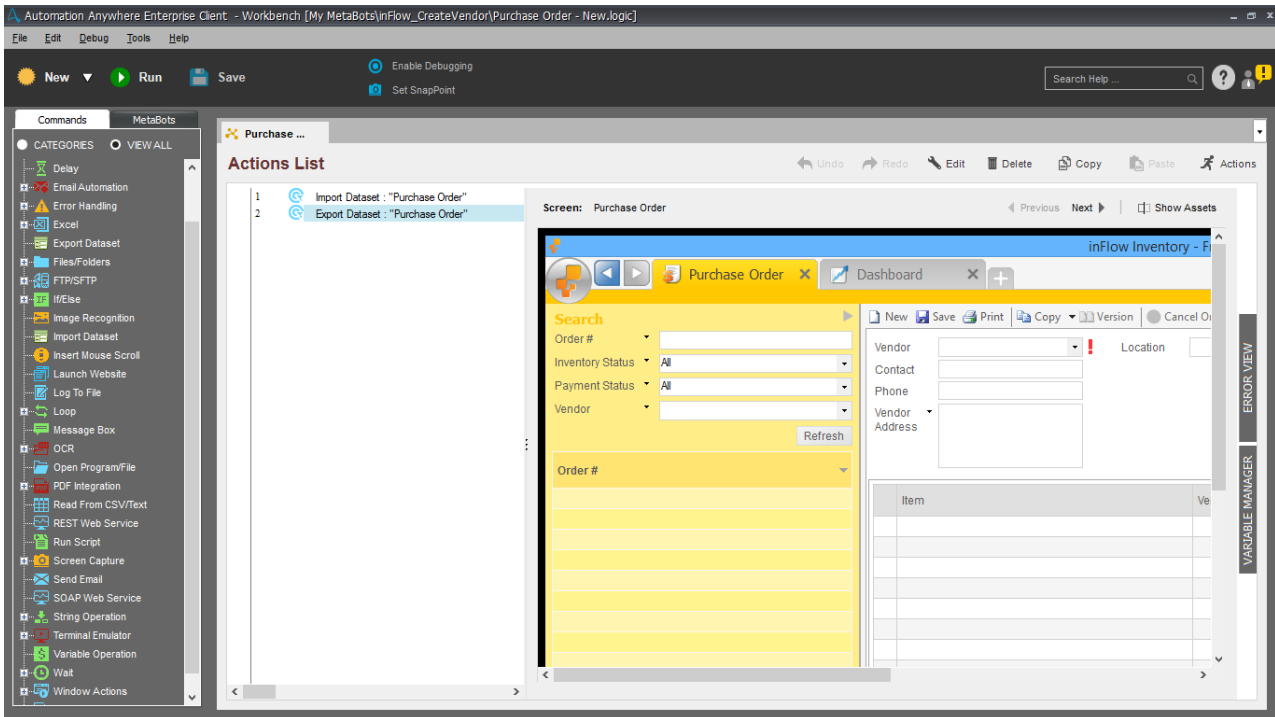
- You can also insert object values using the properties window. Double click the Value field to launch

| # | Entry Name | Type | Value | Read | | | | | | | | | | |
|---|---------------------|-------------|----------|-------------------------------------|------|-------|-----------|---------|------|---------|-------|--|-------|---------------------|
| 1 | Contact | StaticText | \$Name\$ | <input checked="" type="checkbox"/> | | | | | | | | | | |
| 2 | Contact | | | <input checked="" type="checkbox"/> | | | | | | | | | | |
| 3 | Select Action : | GetProperty | | <input checked="" type="checkbox"/> | | | | | | | | | | |
| 4 | Property Name : | Name | | <input checked="" type="checkbox"/> | | | | | | | | | | |
| 5 | Select Variable : | Name | | <input checked="" type="checkbox"/> | | | | | | | | | | |
| Advanced Options Wait for the object to exist: 15 seconds <input checked="" type="checkbox"/> Show Object Properties <table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>Object ID</td> <td>label13</td> </tr> <tr> <td>Name</td> <td>Contact</td> </tr> <tr> <td>Value</td> <td></td> </tr> <tr> <td>Class</td> <td>WindowsForms10.S...</td> </tr> </tbody> </table> | | | | | Name | Value | Object ID | label13 | Name | Contact | Value | | Class | WindowsForms10.S... |
| Name | Value | | | | | | | | | | | | | |
| Object ID | label13 | | | | | | | | | | | | | |
| Name | Contact | | | | | | | | | | | | | |
| Value | | | | | | | | | | | | | | |
| Class | WindowsForms10.S... | | | | | | | | | | | | | |
| TIP: Press F2 to insert variable | | | | Update | | | | | | | | | | |


- Select **Action** that needs to be performed during Logic execution. This can be changed if required.
- Select the **Property Name**
- Select relevant variable
- In **Advanced Options** specify:
 - Wait time** for the object to exist. This can be changed if required.
 - Optionally view the **Object Properties** which will be used to execute the command
- Save or **Update** if in edit mode.

- By default, all fields are displayed. Once you select fields for inserting values, and you open in edit mode after saving only those are shown. To view all click **Show All**.
- You can also choose to change the sequence in which the object values will be filled when automation is executed. For this you must click . This is the default state. To lock the sequence, click . This ensures the sequence is not changed.
- The data that is visible can be filtered on **Type** and sorted on **Type** and **Entry** columns.

5. Your logic will reflect the Export Dataset command in the event data.



6. Save.

 **Tip:** The **Save** button is disabled until all object values that are selected for Play are filled.

It is a best practice to Calibrate Screens that are used for Export Dataset command if the application is subject to frequent updates. These would affect the MetaBot during execution.