MetaBot Designer

Version 3.6

User Manual

Table of Contents

1. Prerequisites and System Requirements

This topic describes the prerequisites and system requirements for installing MetaBot Designer plug-in to your machine.

Prerequisites

To use MetaBot Designer, you first need to install Automation Anywhere Enterprise Client that can be accessed by a Control Room User who has the required privilege to access MetaBot Designer.

System Requirements

- Operating System
  - Microsoft Windows 7, 8, 8.1
  - Microsoft Windows Server 2008 R2, 2012
- CPU
  - Minimum: 2 GHZ, Multi Core Processor
  - Recommended: 3.5 GHZ+ with 4 Core and above
- RAM
  - Minimum: 4 GB RAM
  - Recommended: 8 GB RAM and above for smooth functioning of product.
- Hard Disk space - to install all Automation Anywhere Products namely:
  - Automation Anywhere Enterprise: Approx. 250 MB
  - MetaBot Designer: Approx. 36 MB
  - Total: Approx. 290 MB

    Note: You might have to upgrade to a higher configuration post installation depending upon product usage. For instance, generation of log files, Logic creation, and so on might require more disk space later.
- Microsoft .NET Framework 4.6
- Browser Support: Internet Explorer 8 and later
- Technology Support: Windows, HTML, .Net, WPF, Flex, and Java 1.6 onward (Desktop and Web).

Section 1: MetaBot Designer - Getting Started

2. MetaBot Designer - An Overview

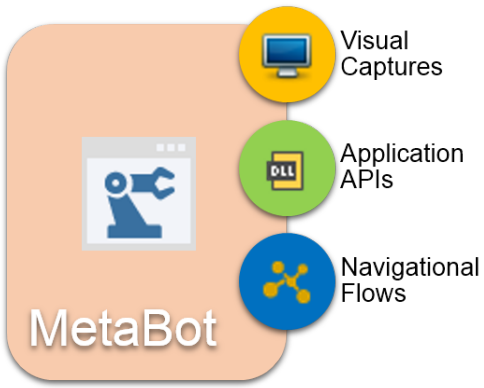
A MetaBot is an automation blueprint of an application that can be re-used to optimize processes. They help you standardize your processes end to end.

MetaBots enable you to create automation building blocks that capture information in the form of visuals or application APIs. These can thereon be integrated as commands and further leveraged by varied processes to deliver value at multiple levels.

MetaBots can be constructed using:

- **Visual captures (Screen) -** These are GUI components (screens) of an application.
  - In MetaBot Designer, a visual capture is referred as Screen.
- **Application APIs (DLL)-** 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 (Logic) -** 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 can be re-used; 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 eliminate common navigational errors in complex automation tasks.
- MetaBots automate without requiring access to live application.
- MetaBots can be easily calibrated to newer versions of applications to ensure compatibility.

Refer Using MetaBot Designer in Automation Anywhere Enterprise and Downloading MetaBots in Enterprise Client for details.

The MetaBot Designer

The MetaBot Designer helps you to:

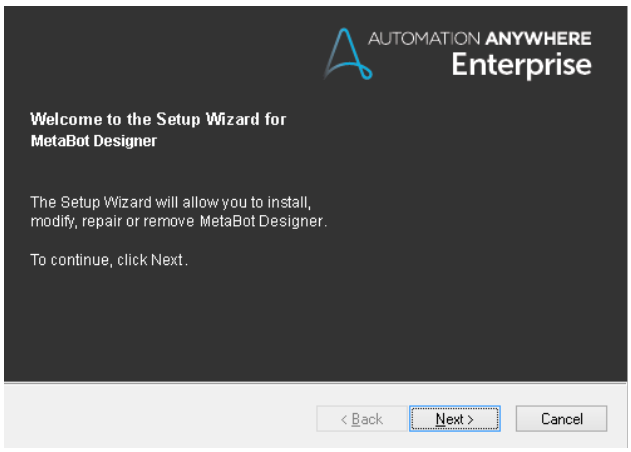
- Create MetaBots.
- Add Assets - Screens and DLLs to your MetaBots.
- Manage your MetaBot Assets.
- Configure and Calibrate the Screens.
- Create custom objects on the Screens.
- Create Logic to represent a navigational flow using Screens and DLLs.
- Upload the MetaBots to the Control Room for use by other Enterprise Client users.
- Edit MetaBots created by other users.
- Export and Import MetaBots for use in different Control Room setups.

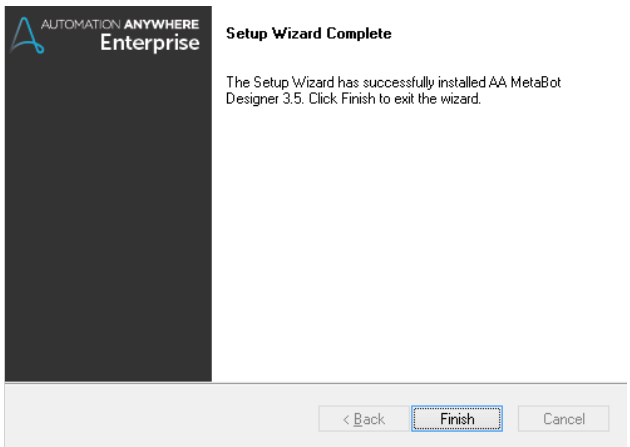
Installing the MetaBot Designer

System Requirements

- Pre-requisites: Automation Anywhere Enterprise 10.0
- Operating System: Windows 7, Windows 8, Windows 8.1
- Browser: Internet Explorer 8 and later
- Technology: .Net, WPF, Java 1.6 and above (Desktop and Web), HTML, Windows, Flex
- Hardware requirements: Same as Automation Anywhere Enterprise 10.0

Note: The MetaBot Designer is **an add-on** to Automation Anywhere Enterprise (AAE). With an AAE edition already installed on your machine, run the MetaBot Designer setup.

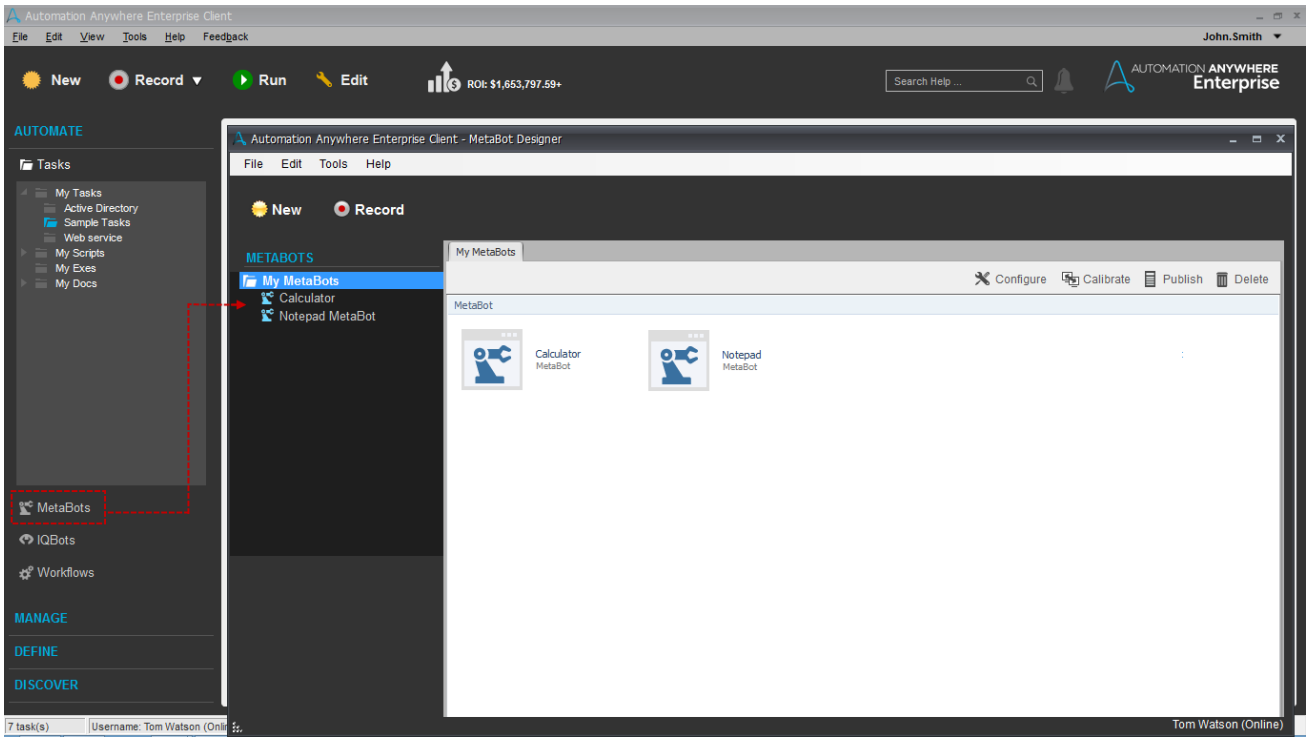




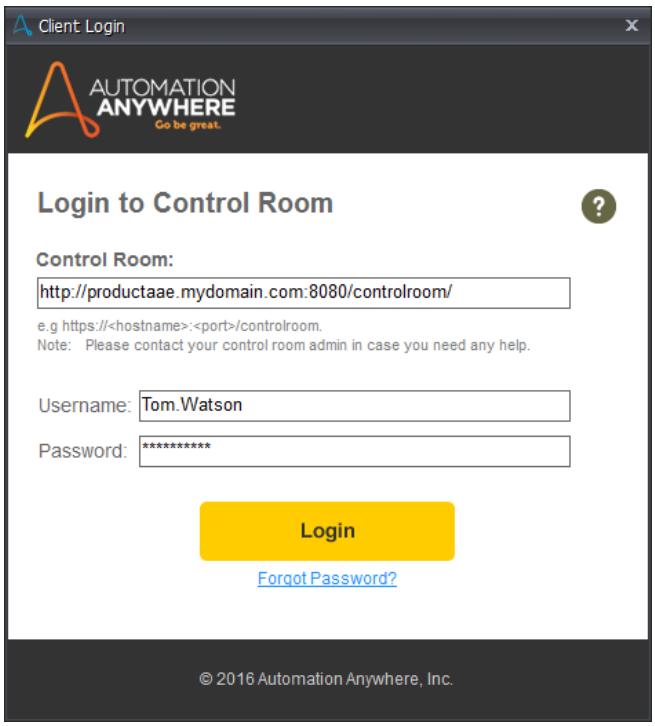
Launching the MetaBot Designer

After successful installation of MetaBot Designer setup, the feature is added to the product under the Automate Tab.

Click on 'MetaBots' to launch the MetaBot Designer.

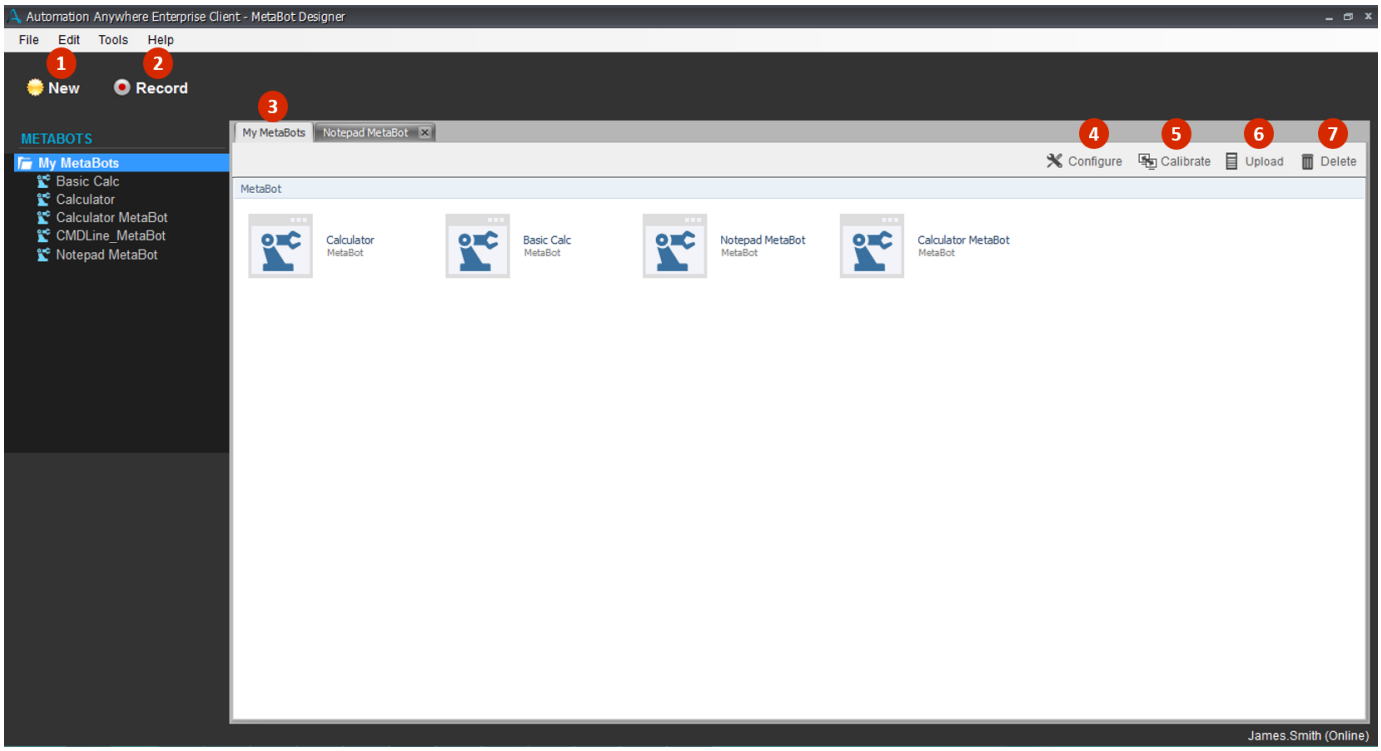


Log in as a MetaBot Client to Control Room



Note: The MetaBot Designer will automatically log you in if you are setup as a valid user in Control Room and are already signed in the AAE Client at the time of launch.

Understanding MetaBot Designer



Learn More

You can access the features illustrated below in MetaBot Designer:

1. **New -** Use this to create a New MetaBot. The MetaBot thus created does not include any Assets or Logic; its empty. You can start with either - 'Record Screen', 'Add Screen' or even 'Add DLL' based on the purpose for which the MetaBot needs to be created.

   Tip: The MetaBots are saved to the application path in the 'My MetaBots' folder. You can save these to a location you desire by changing the settings in Tools > Options > System Settings.

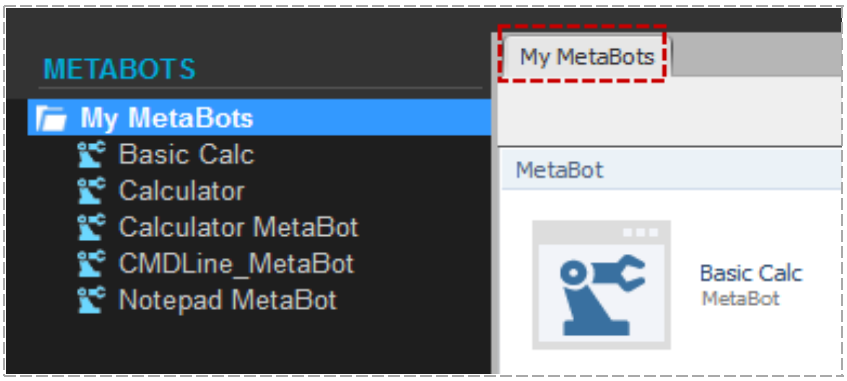
   Learn More

2. **Record -** Record is an alternate method of creating a MetaBot. Here, at-least one MetaBot will be present as its recorded.

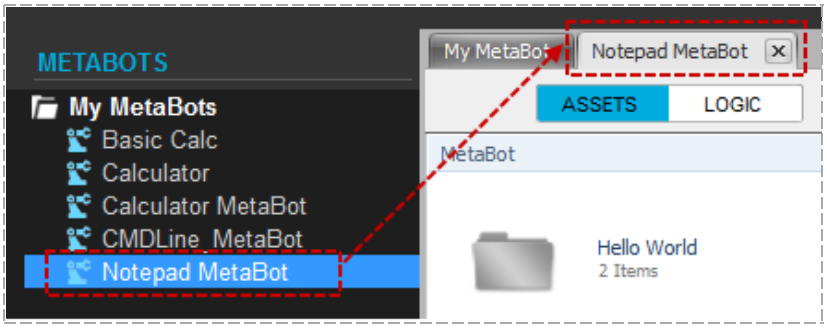
   

   Learn More

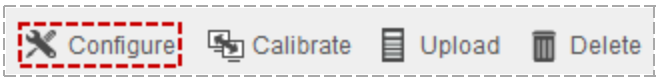
3. **My MetaBots -** This tab is shown by default; it displays all MetaBots. This tab is always open, by default.



Note: Each selected MetaBot opens in its own tab:



4. **Configure -** Use this to edit your application properties for the recorded/added screen.

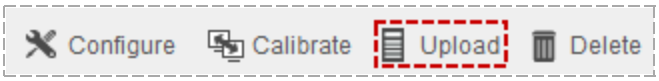


Learn More

5. **Calibrate -** Use this to to instantly compare an existing screen with a newer screen to identify changes if any.

Learn More

6. **Upload -** Use this to upload the MetaBots to the Control Room for download by other MetaBot users.



Learn More

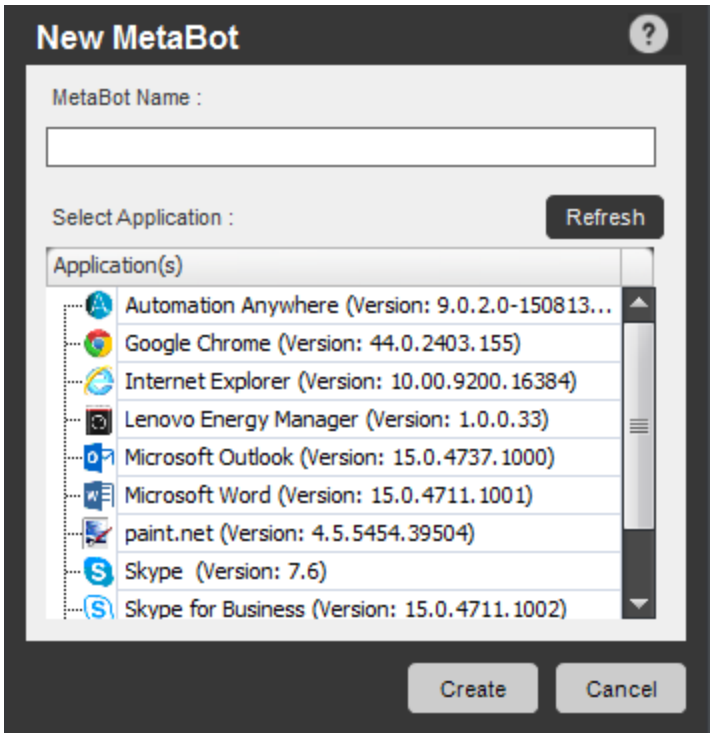
7. **Delete -** Use this to delete MetaBots, Screens and DIIs.
8. **Export/Import -** Use this to export and import MetaBots for use in different Control Room setups.
   Learn More

3. Creating a MetaBot

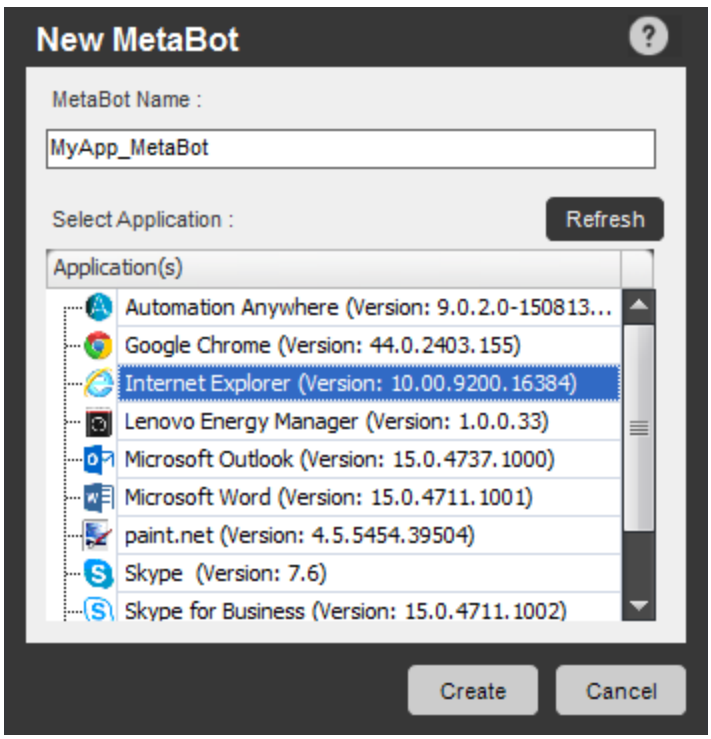
Let's create your first MetaBot. It will allow you to login to your Automation Anywhere account.

1. Open the MetaBot Designer and click on New.

2. A smaller window, called New MetaBot, opens.

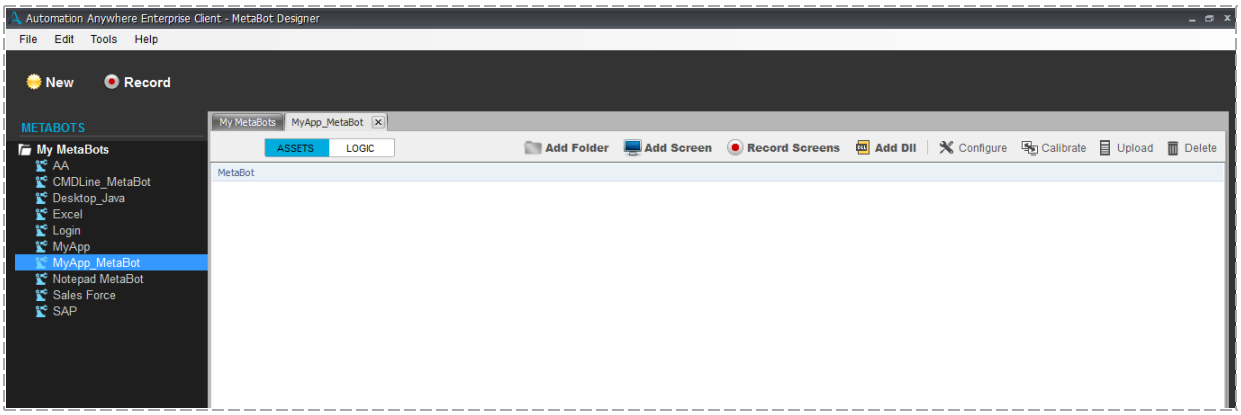


3. The New MetaBot window lists all the currently running applications. From this list, select an application (Internet Explorer in this case) for which you want to create a MetaBot. Type in a name - "MyApp_MetaBot" for instance and click on Create.



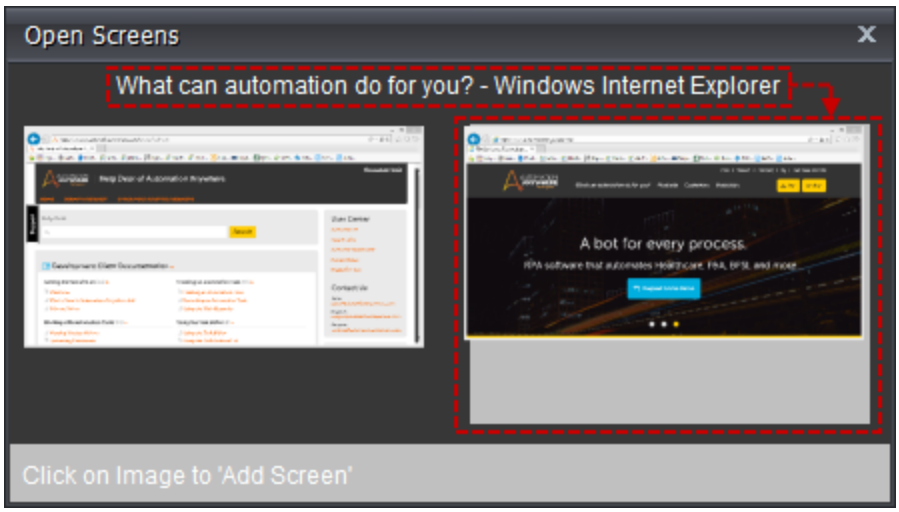
Tip: If you forgot to launch the application before starting to create a MetaBot for it, just open it now and click on Refresh on the New MetaBot window. Your application will appear for selection in the list.

4. This creates and opens a new but empty “MyApp_MetaBot”. It cannot do anything for you yet.



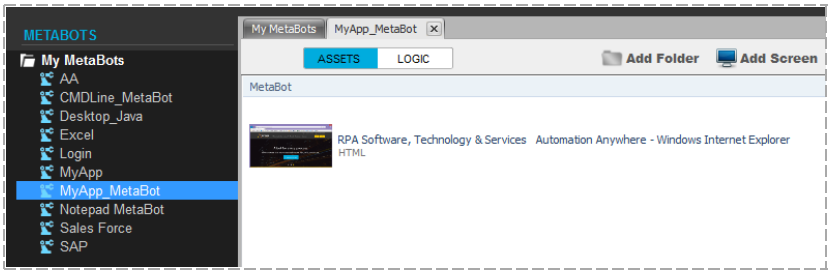
5. To Login to “MyApp” your Automation Anywhere account, we need its screen asset. Click on 'Add Screen'.

6. On the center of your screen, a small window called 'Open Screens' is displayed. This window will list all currently open instances of the target application for you to select.

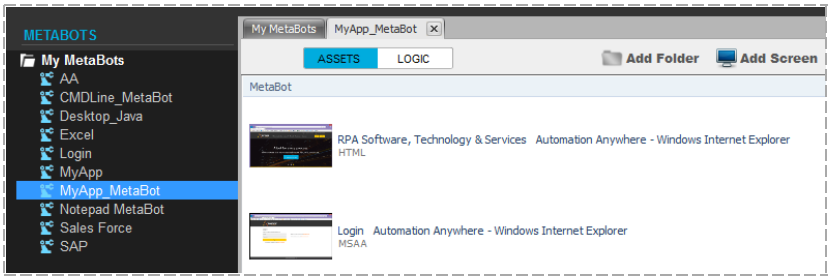


Note: If you have two instances of IE open - as is the case here, you will see two separate screens displayed in 'Open Screens'.

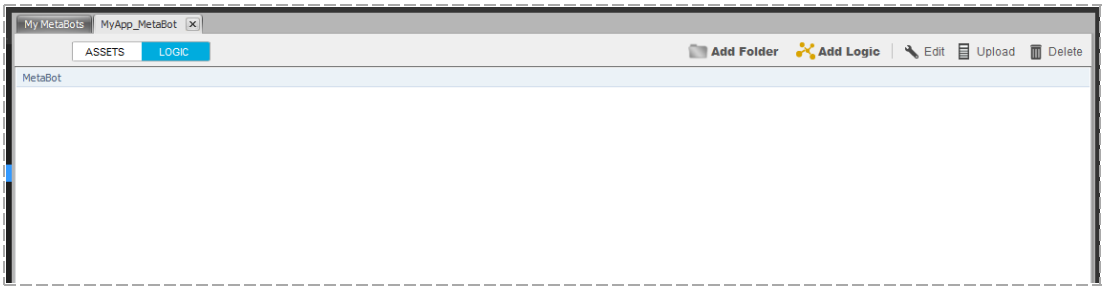
7. Click on the Windows Internet Explorer screen instance “What can automation do for you?...” to proceed.
8. The selected screen gets added to your MetaBot.



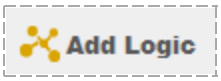
9. The next logical step would be adding more screens to your Assets library. Use either the 'Add Screen' or 'Record Screen' option. Learn More.
10. Let's add another screen to Assets. This screen will be used to add credentials.



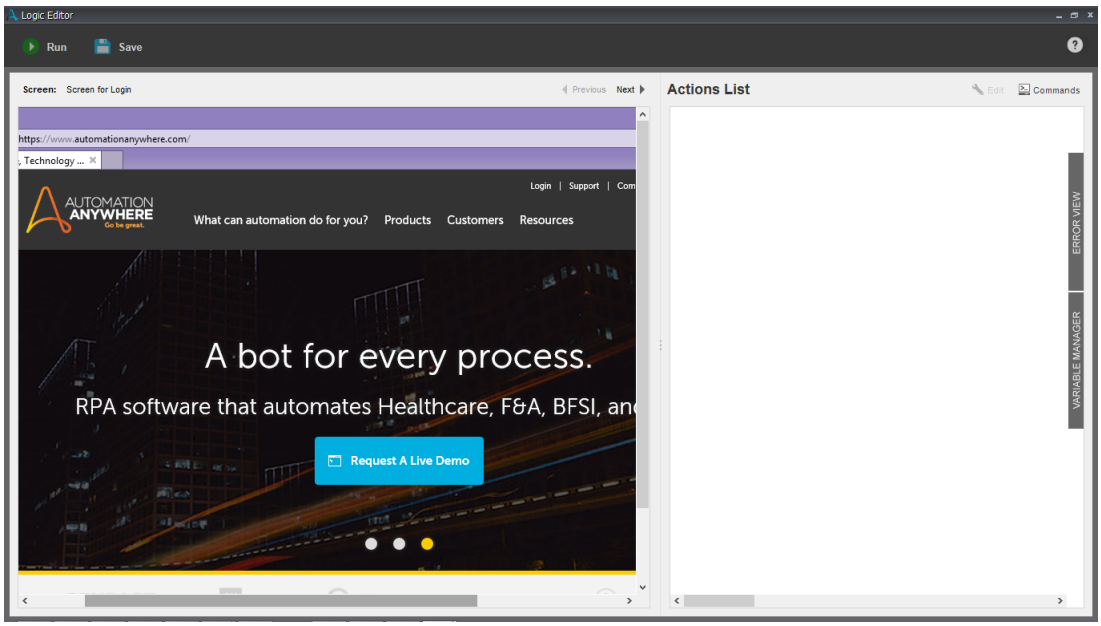
11. Let's use these screens to create a logic to login to your account. Click on Logic tab.



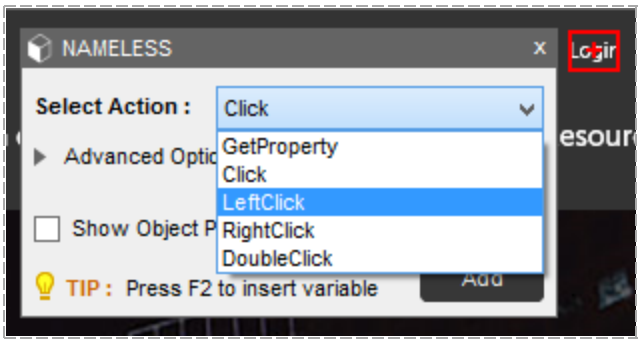
12. Click on Add Logic.



13. This opens a large window called Logic Editor, 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.

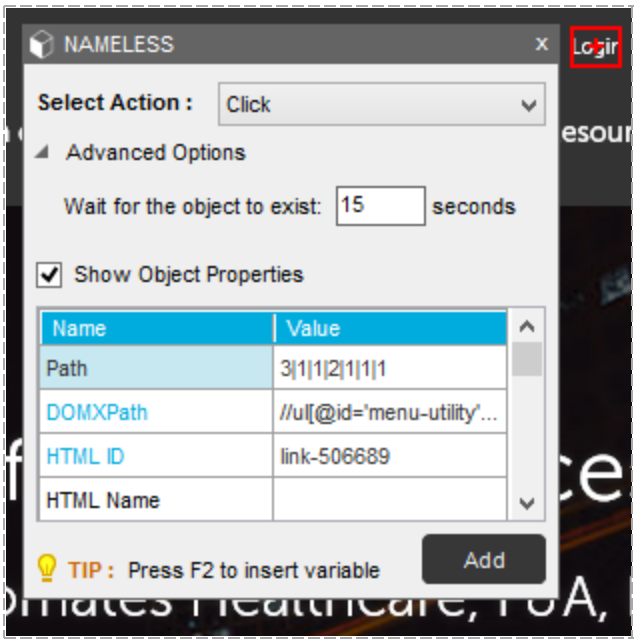


14. Click on 'Login' in the screen that was captured first. This will open a 'Properties' window near the selected object that will allow you to configure 'Actions'.

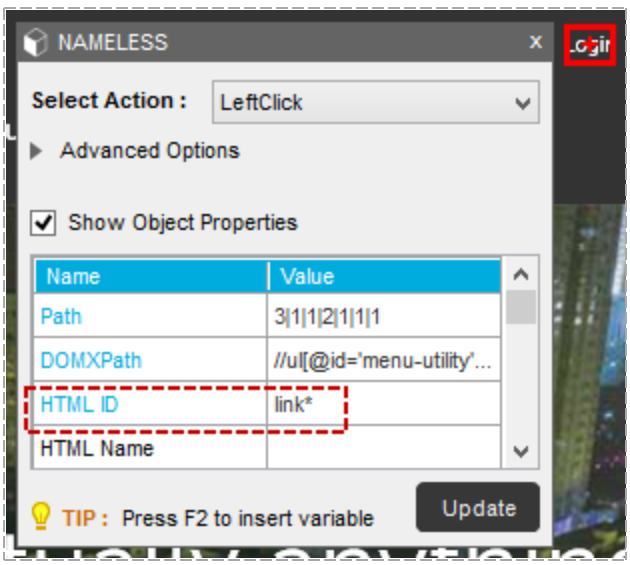


15. Select either 'Click' or 'LeftClick' from the drop-down.

16. In Advanced Options input the Wait time for the object to load. Enable 'Show Object Properties' to view the properties that will be used to search the object during play time.



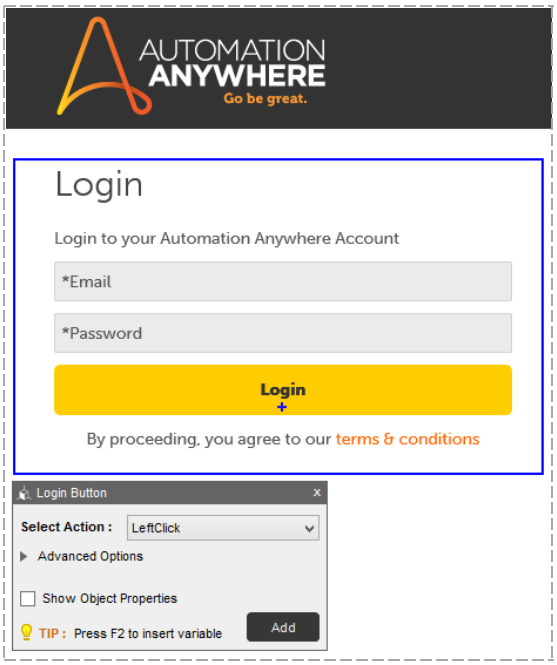
If the selected object has dynamic properties in its search criteria, it is recommended to use '*' in place of the actual value. For instance, in the above, change the value of property 'HTML ID' as shown:



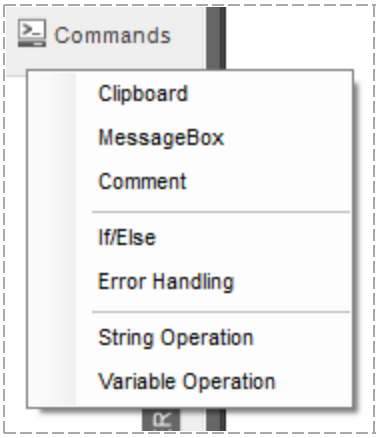
To know more refer Configuring MetaBot Screens.

Tip: If no name is displayed for the selected control, you can give an easily identifiable name to it. Learn More.

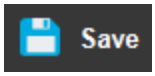
17. Subsequently, add other screens by clicking 'Next'. For instance, the above action will navigate you to the login screen wherein you have to input your credentials:



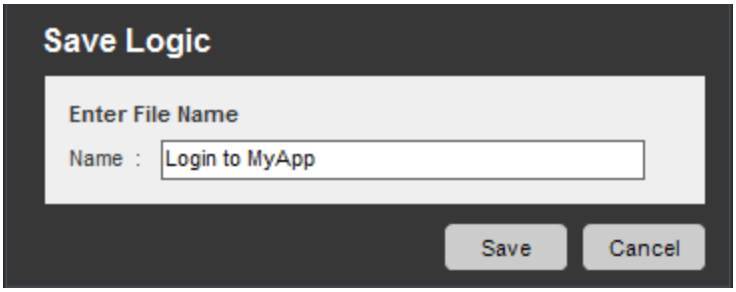
18. If your Logic requires that you add commands, select ones that are most appropriate for your logic. Learn More



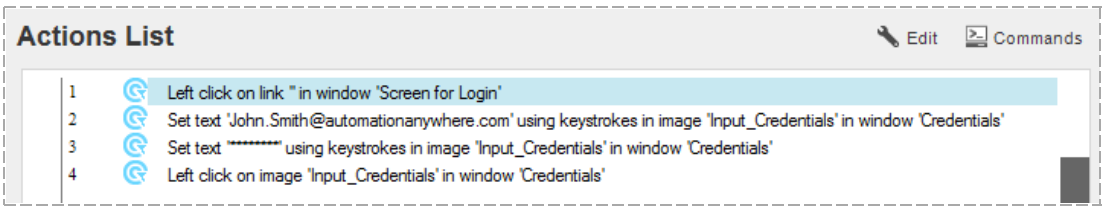
19. If needed, you can also add DLLs. Learn More
20. Once done, click on Save.



21. You will be prompted to enter a name for this new logic that you just created. Name this logic as “Login to MyApp”. Click on Save to finish.



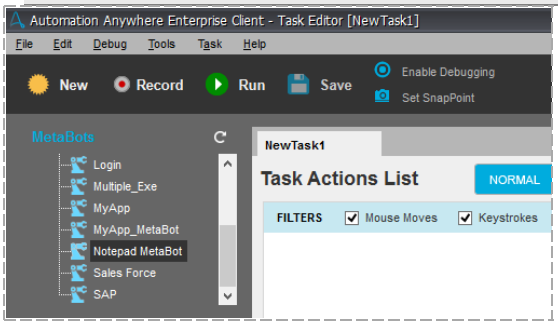
22. The saved actions will appear in the Logic Editor under the Actions List.



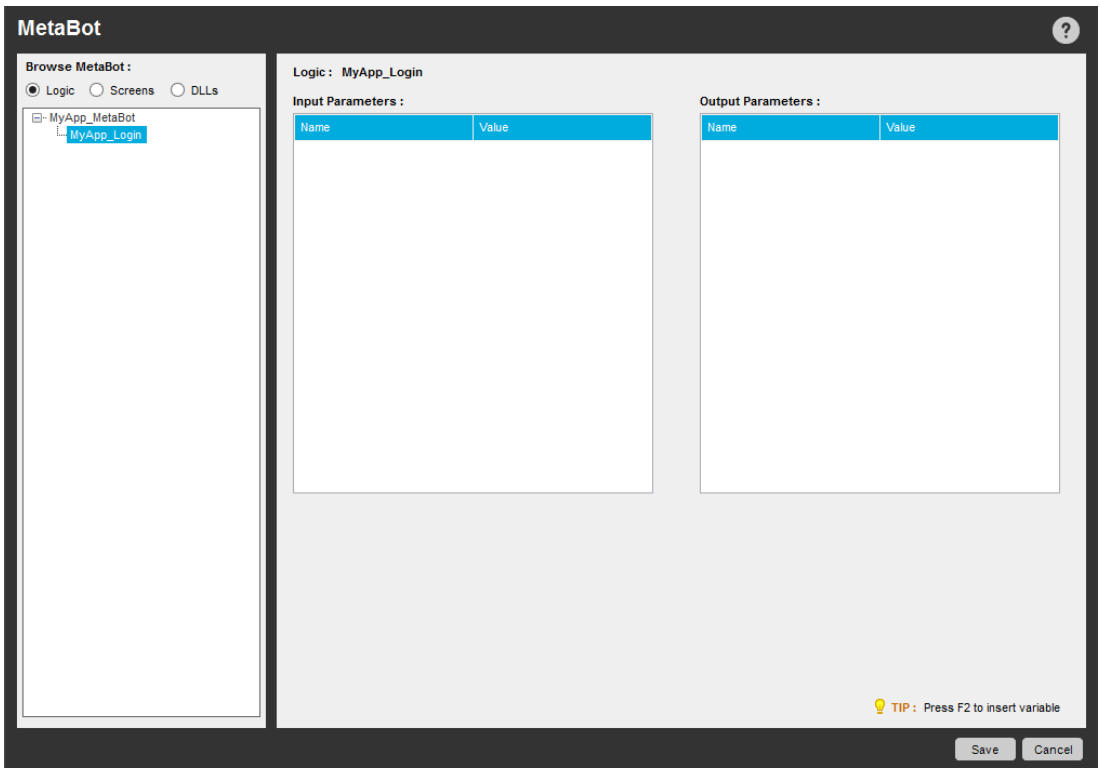
23. Let's run this logic to see if it works as expected. Click on Run.

24. Yay! You have successfully created a MetaBot with a logic to login to MyApp - your Automation Anywhere account.
25. 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 Task Editor to create a new automation task.
26. Select the 'MyApp_Metabot' from the list of MetaBots on the top left and drag it onto the task action list.

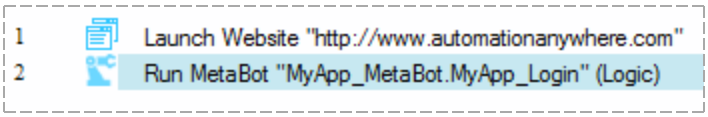
Note: These are available after the MetaBots have been Uploaded to the Control Room and later Downloaded to the Client.



27. This invokes the MetaBot UI where you can browse and select from the list of available Logic, Screens or DLLs. Select the MyApp logic and click on Save.



28. The logic gets added to your task.



Tip: If you specify any input/output parameters while creating a logic in the Logic editor, you can see them here. We didn't use any input/output parameters while creating "MyApp_Login" Logic. So you will find the parameter selection disabled. Learn More

29. Save the task and run.

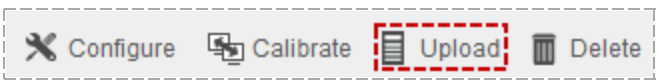
Tip: You can launch the website first and then use the "MyApp_Login" logic.

30. Hooray! You have successfully used a MetaBot to login to your Automation Anywhere account.

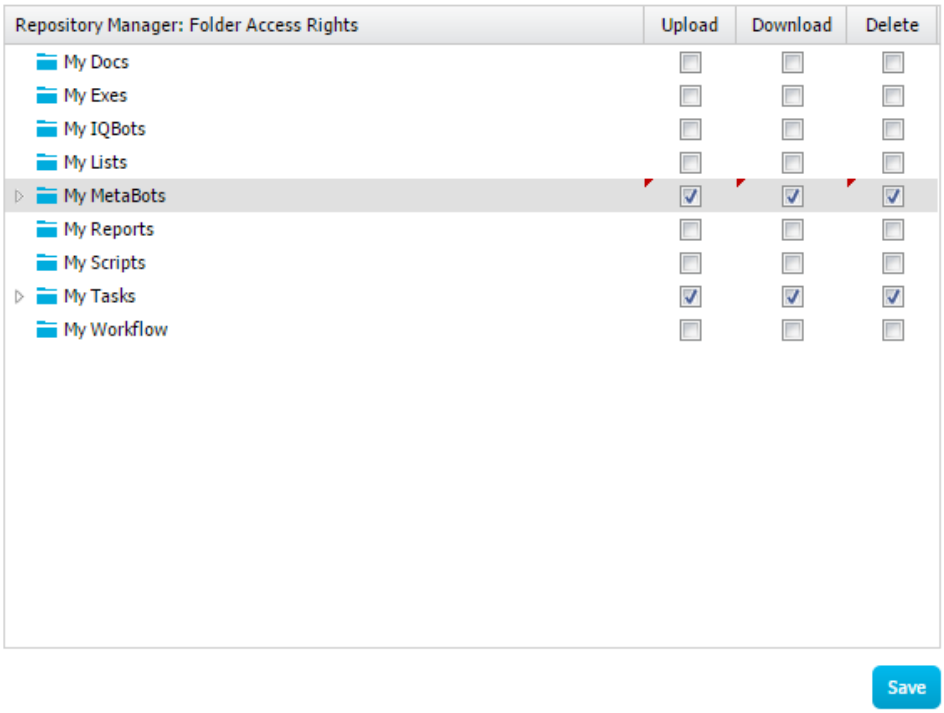
Uploading MetaBots

MetaBots, once created, can be uploaded from MetaBot Designer to Control Room from where any number of AAE development clients can download and use them.Learn More

- In MetaBot Designer, you can 'Upload' MetaBots to the Control Room.*

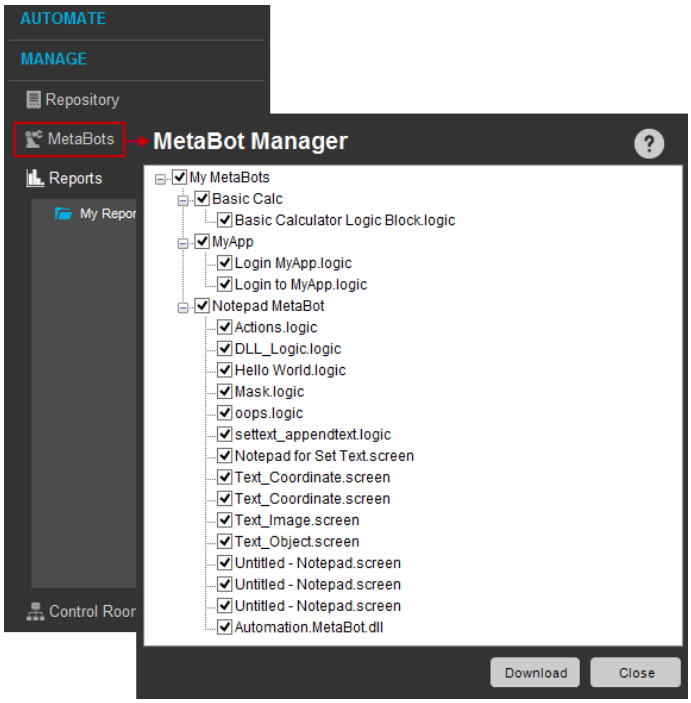


- The Control Room administrator can provide appropriate rights to an Enterprise Client user to upload and/or download the MetaBots.

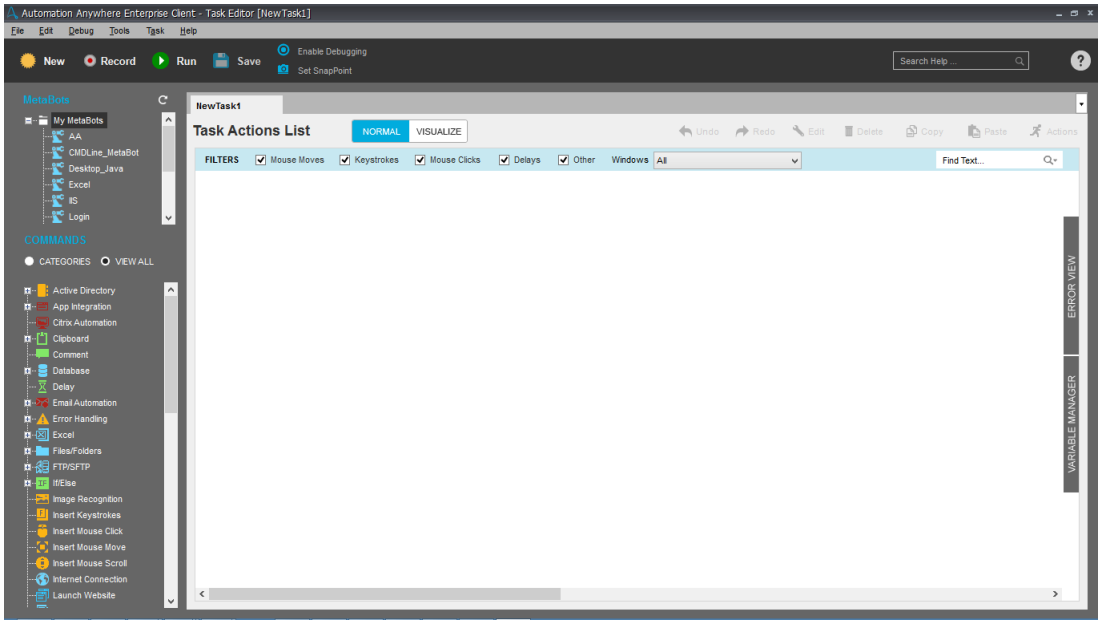


- In the AAE development client, you can access all uploaded MetaBots and 'Download' the MetaBots that you need to include in your automation task.*

  ** The Upload and Download features require logging in to the Control Room*

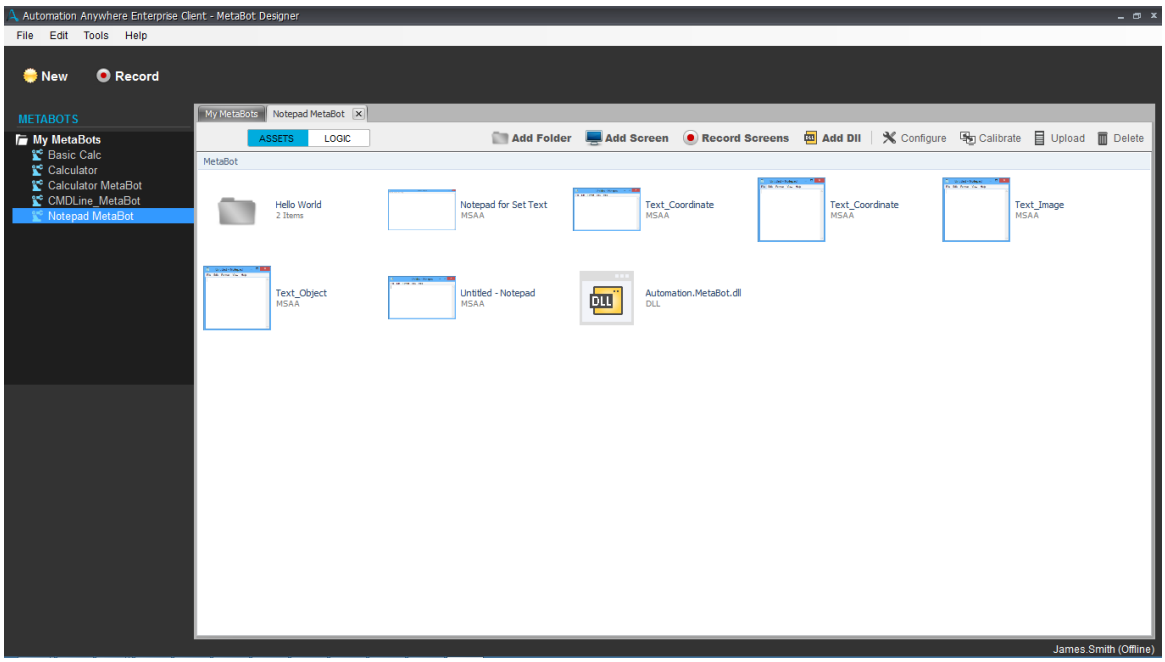


- Use the downloaded MetaBots in your automation tasks just as you would use a command.



- Click the refresh icon at the top in the MetaBots section to view the latest dowloaded 'My MetaBots'.

4. Understanding MetaBot Designer

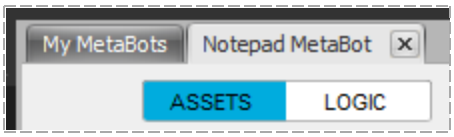


Using the MetaBot Designer, you can:

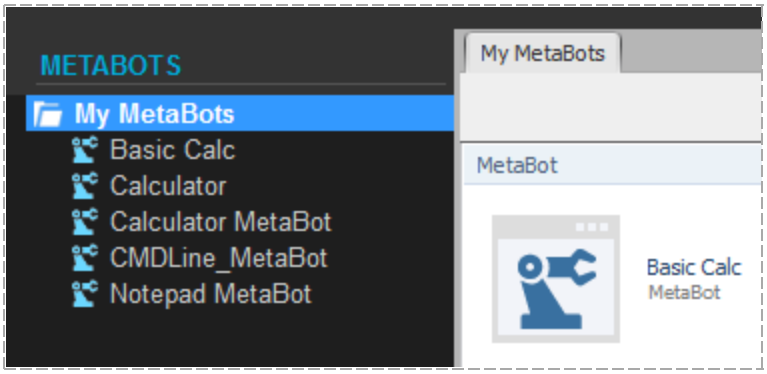
- Specify folders in an existing MetaBot,
- Add DLL's to create low level operations of an application by circumventing GUI
- Add and record new screens to a MetaBot;
- Create simple independent yet functional logic blocks.
- Also, create MetaBots and upload them to Control Room to create a repository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



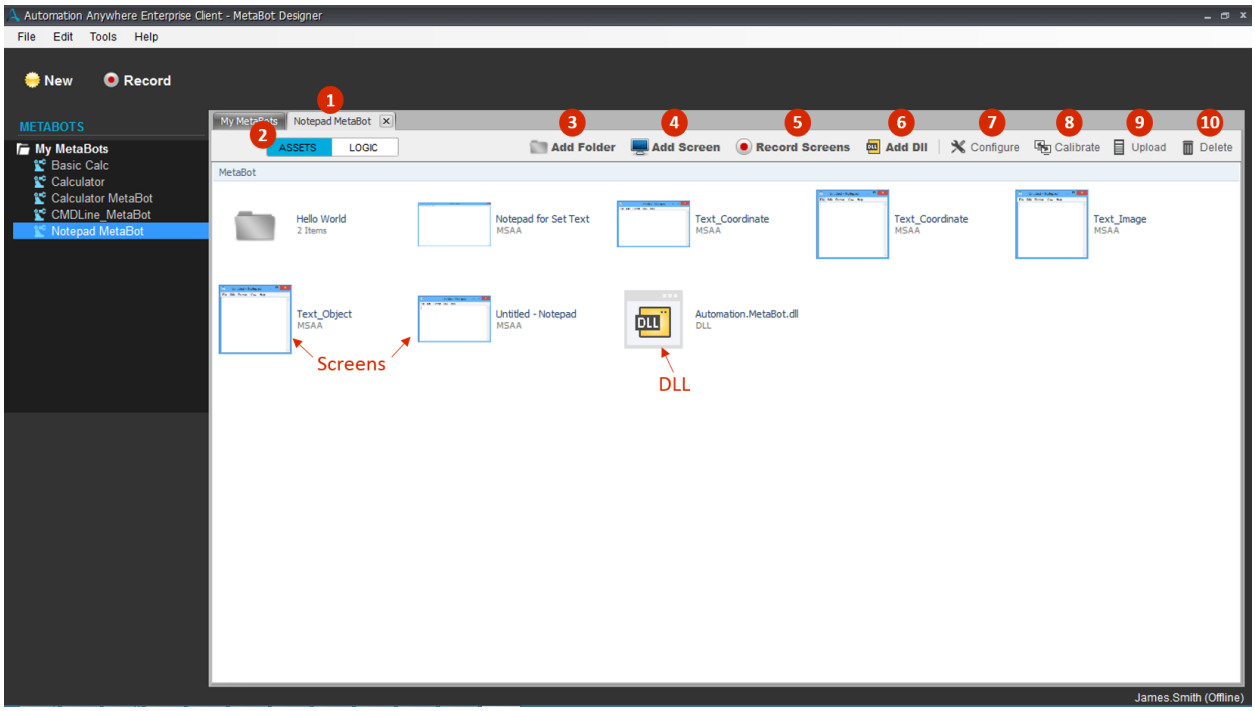
The MetaBots that you create; whether Screen or Dll, are listed in the left panel of the application. You can select one to configure/calibrate in the work space provided.



Note: Select 'My MetaBots' in the left pane to view all MetaBots at one go.

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'.

   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.

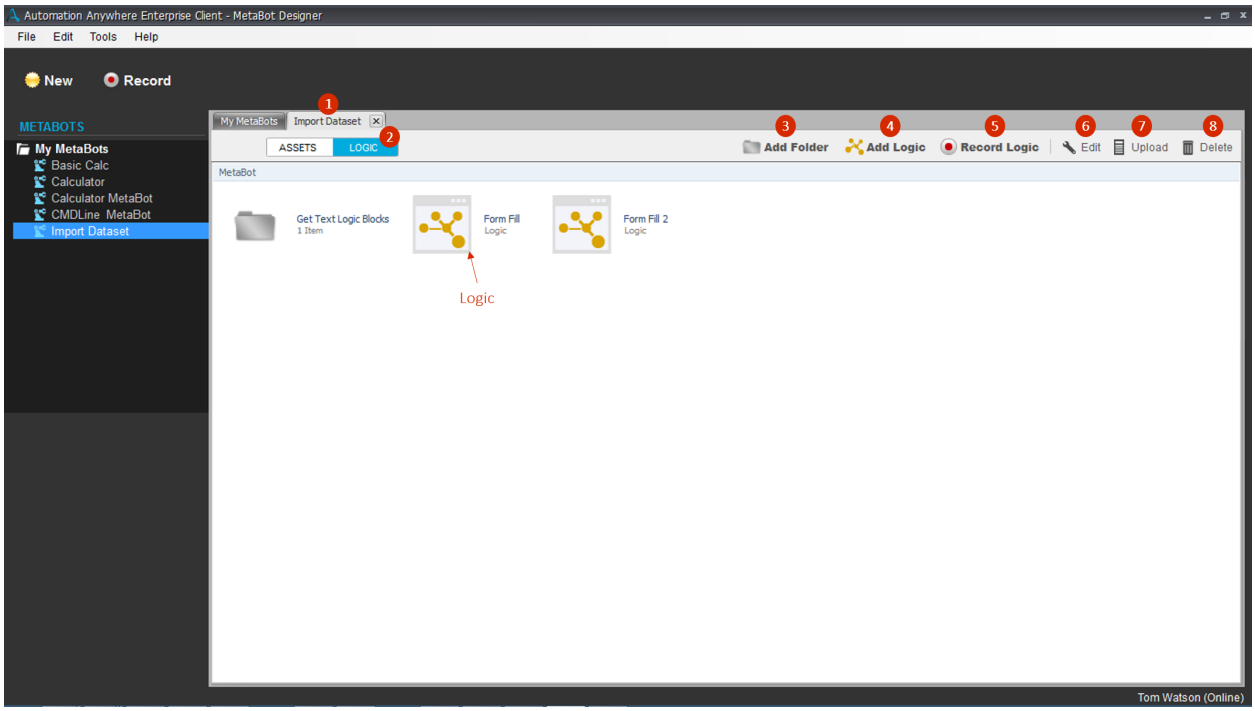
   Learn More

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.

5. 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 work-flow.

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.

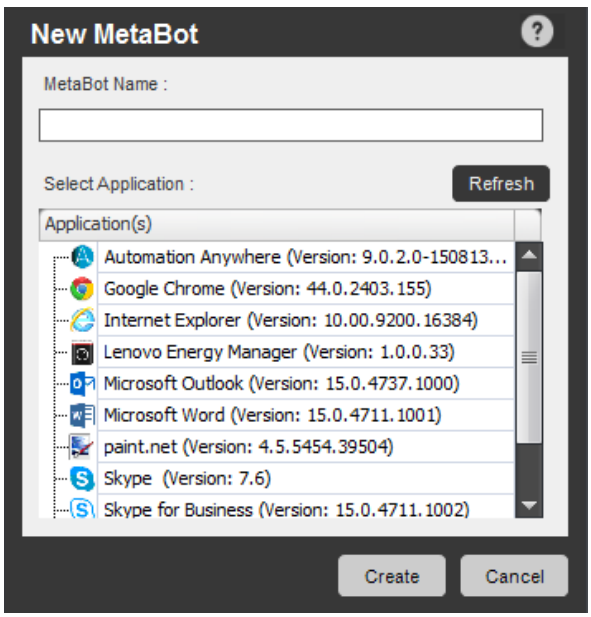
Step by Step Guide to Creating a MetaBot

Creating a New MetaBot using 'New'

1. Click New MetaBot

2. Select an application that you wish to create a MetaBot for. 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.



3. Click Create to save the MetaBot.

- The MetaBots are saved in the My Documents folder. You change the location to save your Metabots in the AAE Menu Tools > Options > System Settings.

Creating a new 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.

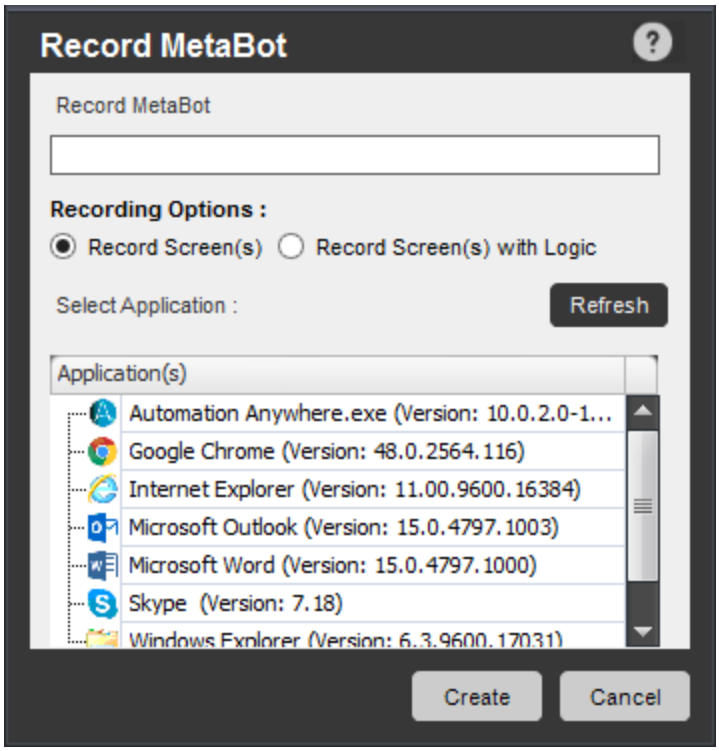
1. Record Screen(s)

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

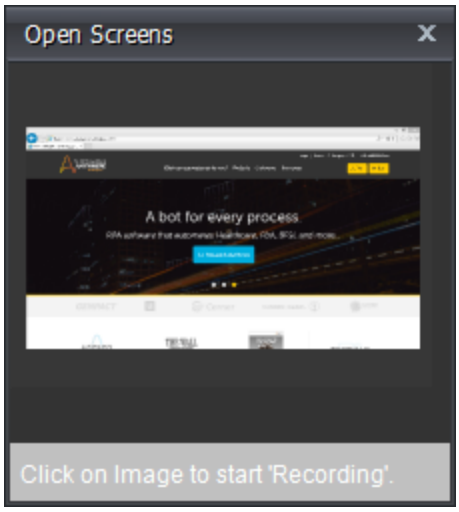
1. Click record

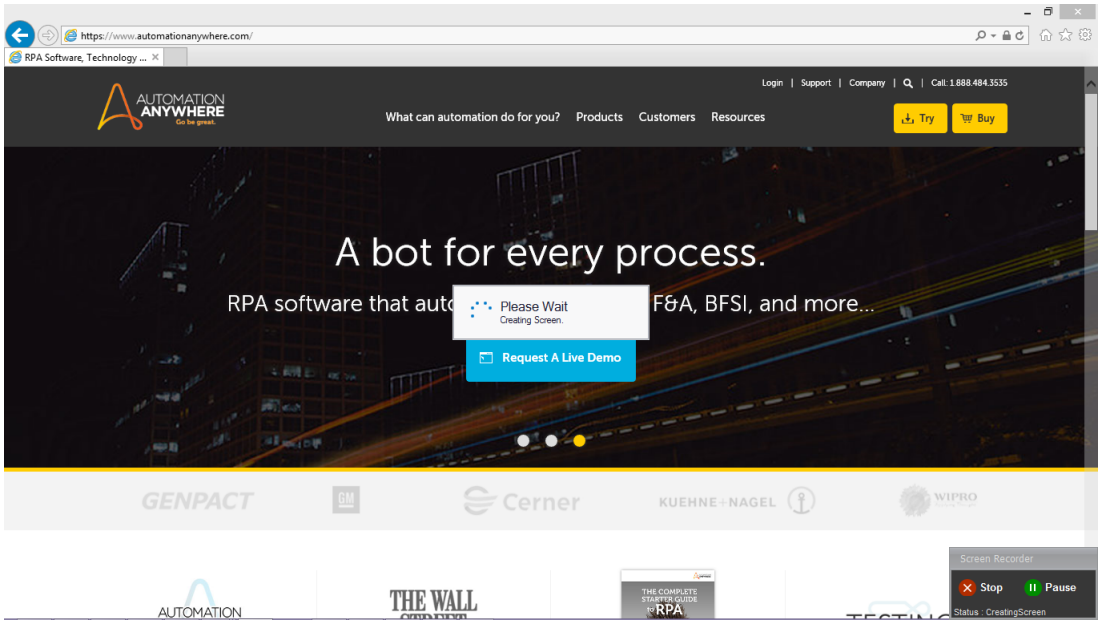
2. Specify the application from which you wish to capture GUI properties.
3. Select 'Record Screen(s) option in the Record MetaBot window.



4. Click on the image in Open Screens.

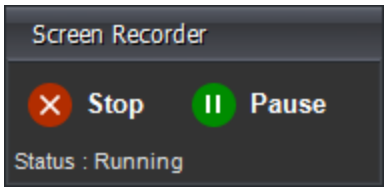


- MetaBot Designer begins recording the selected screen of the application:

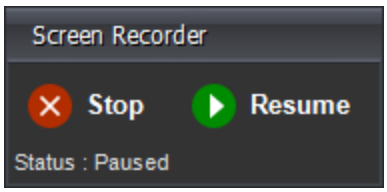


> Note: Each click will record the screen.

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



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

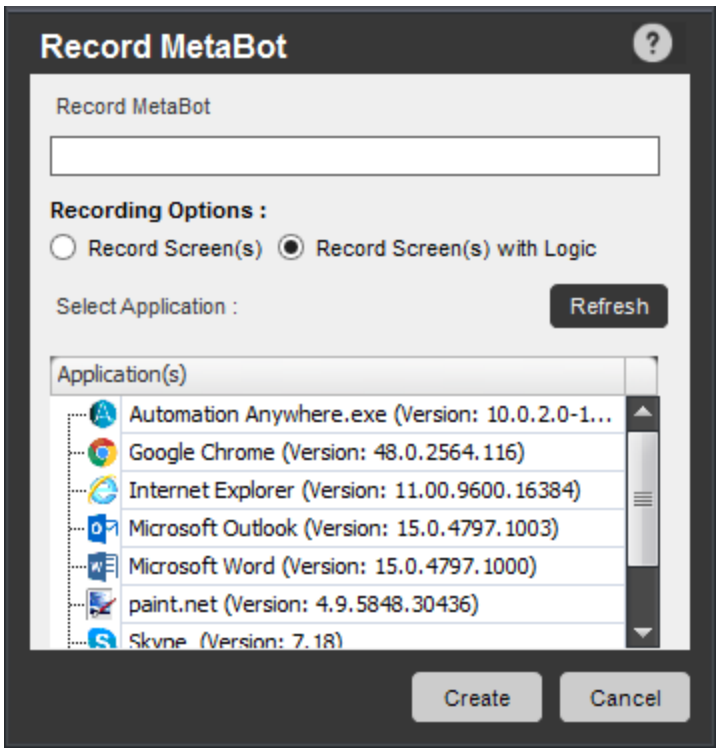


Note: If an application has multiple exe's, you are required to create a separate MetaBot for each.

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

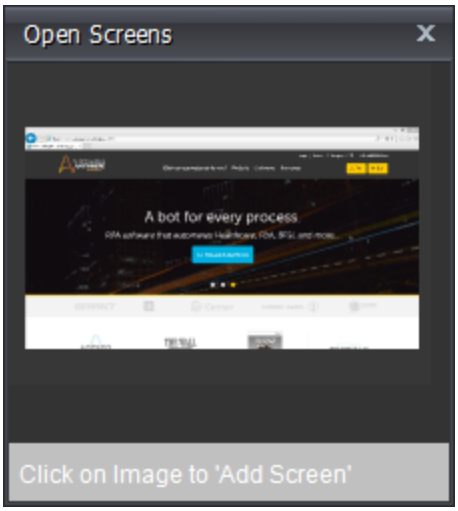
Adding Screens to a MetaBot using 'Add Screen'

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

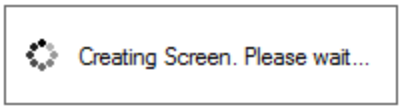
To add a Screen,

1. Click 'Add Screen'

2. Click on Open Screens window to begin capturing:



3. Wait for the Screen to capture:



4. The screen is added to the current MetaBot.

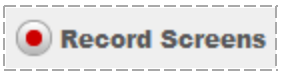
Recording Screens in a MetaBot using 'Record Screens'

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

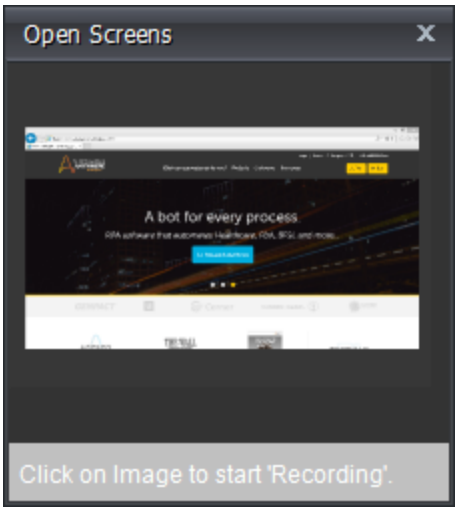
Use Record Screen to record all the Screens/UI elements (Menu item / Popup / context menu etc.) while you interact with the application in a workflow mode. These UI elements cannot be captured using Add Screen.

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.

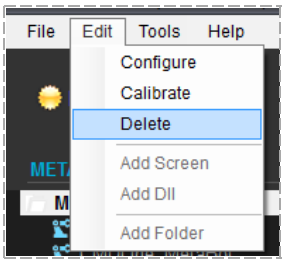
i.e. carry out all functions that you would when creating a new MetaBot.

Deleting MetaBots

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

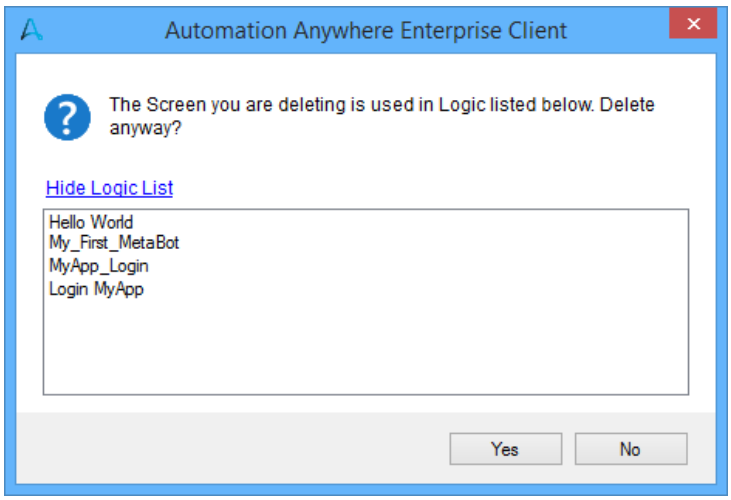
Delete in one of the following ways:

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':



6. Configuring MetaBot Screens

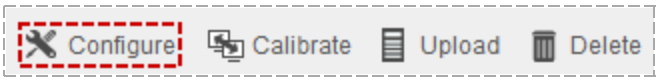
Essentially, you need to create MetaBots in such a manner that they cover all possible run time scenarios. This will ensure that the tasks in which they participate run without any glitches. Thus, simply capturing a screen may not help in most cases. This is where the Configure feature comes into picture.

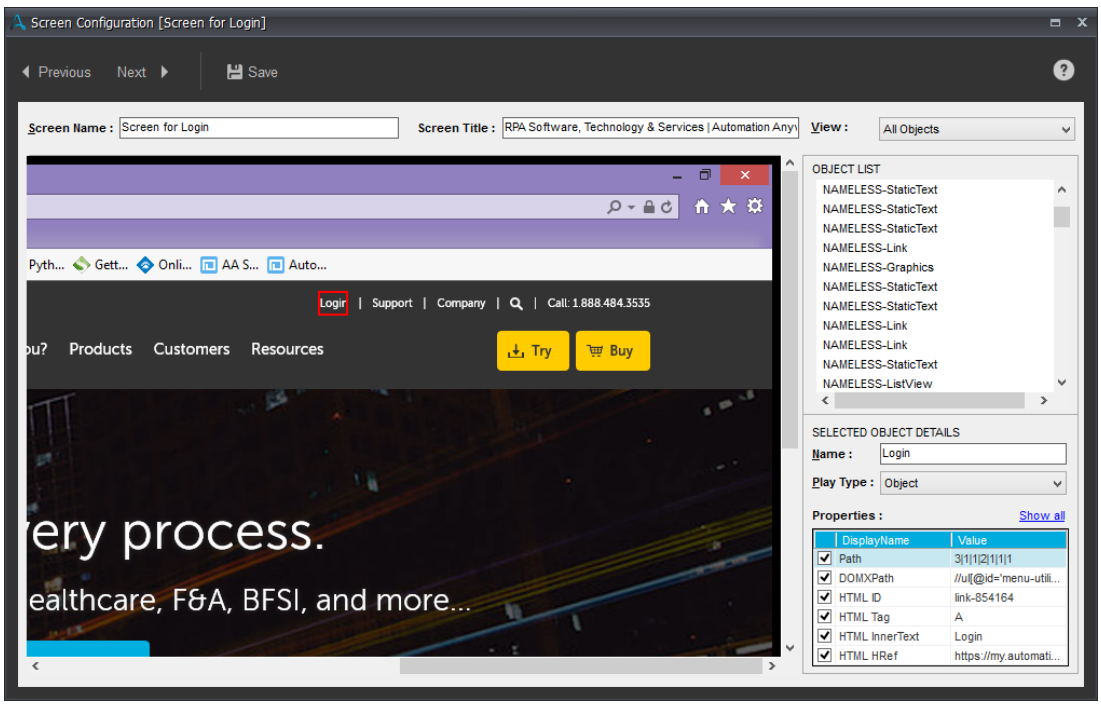
Use 'Configure' to edit object properties for the recorded/added screen. Here you can provide aliases such as a 'Screen Name' and a 'Screen Title'. You can also select an object to define its properties such as Name, Path, Value, ID, Class, Index, States etc and the 'Play Mode' to be used when running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

Guide to Configuring a Screen

1. Select a screen.
2. Click Configure to launch the Screen Configuration editor





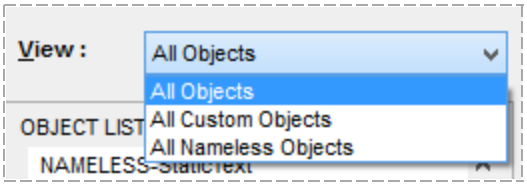
3. Highlight the control which you wish to use during run-time.
   - The object properties are populated in the right panel of the screen.
4. Optionally, provide a Screen Name.

> Note: Screen name is the name that we provide to the added screen, while screen title is the one that appears on the screen window that is configured.

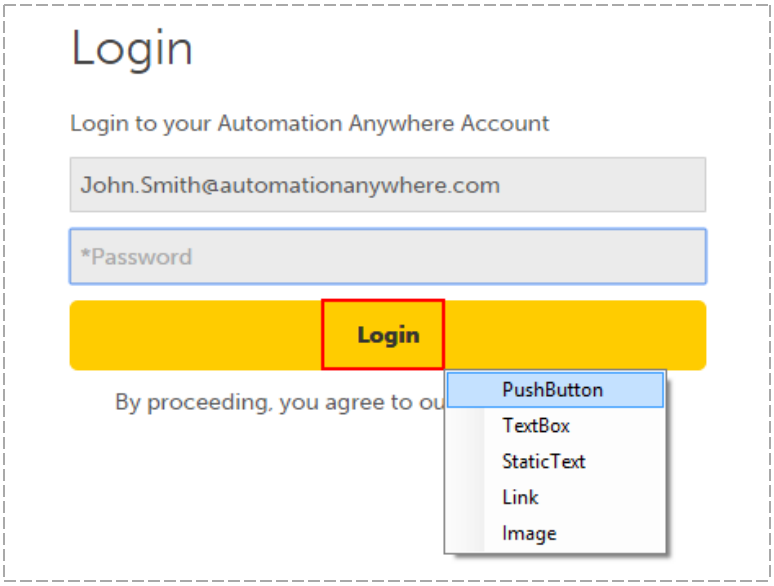
   - You can rename the Screen if you feel that it would be easier for a user to identify with something else.
   - Also, it is recommended that you provide a generic screen title to ensure compatibility with all situations

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

5. To customize the view type of object that has been captured, select any one from the list:

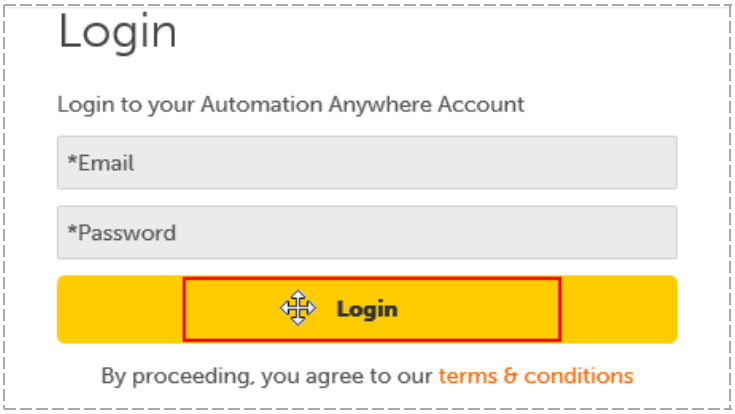


- Custom Objects are the ones that you define especially when the object properties are not captured while recording/capturing a screen. Simply drag and select the area; choose whether to treat the object as Push Button, Text Box, Static Text, Link or Image.

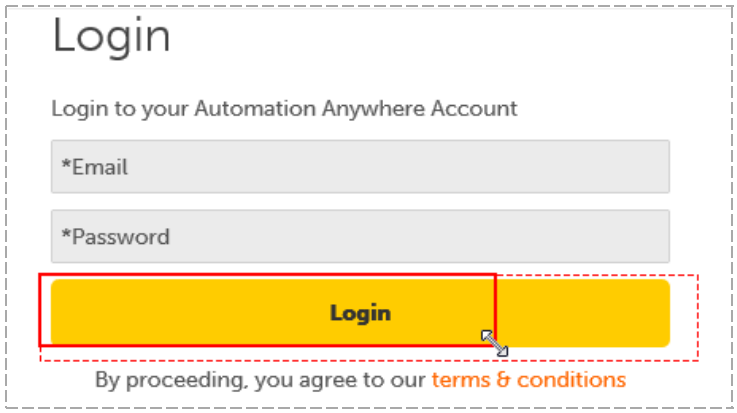


When custom objects move or are re-sized, you will have to update relevant Screen(s) to reflect the same. You can achieve this by moving and/or re-sizing those during configuration.

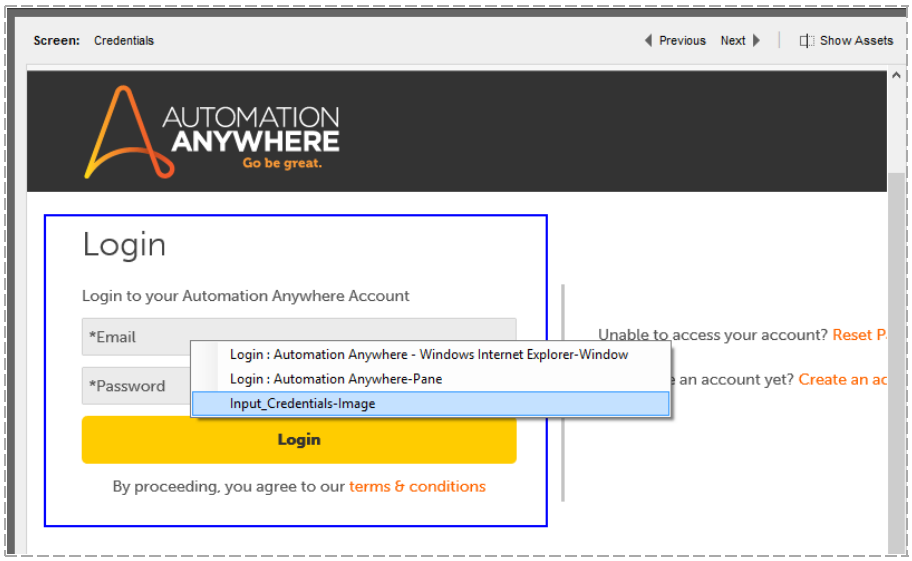
- To move the customized object to a different location, click and drag it to the target location.



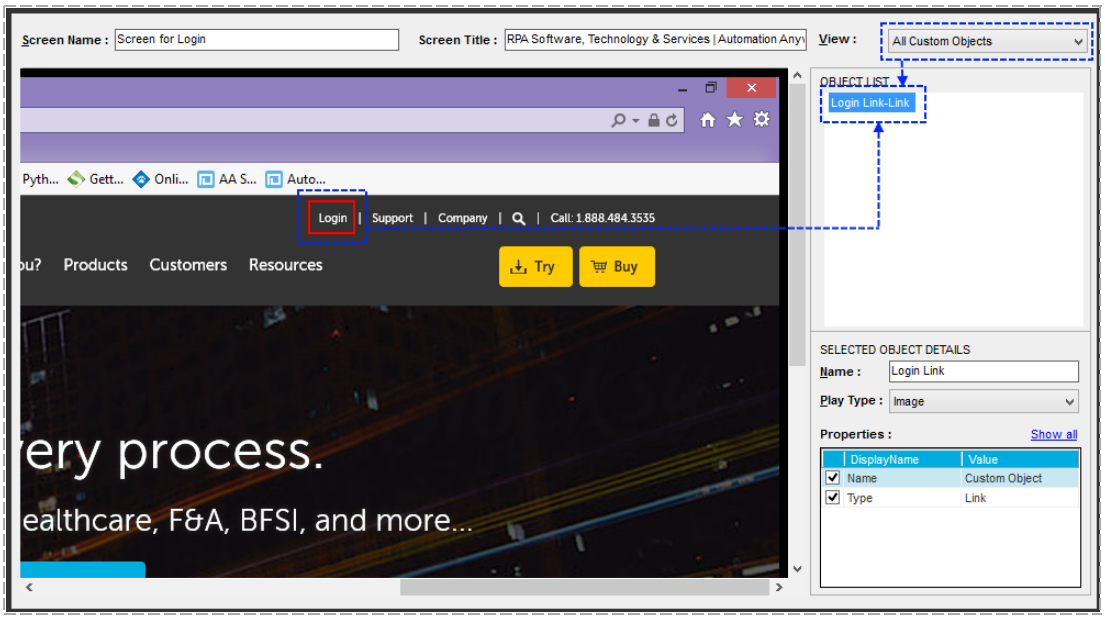
- To re-size a customized object, hold down the click button and drag the pointer as required.



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



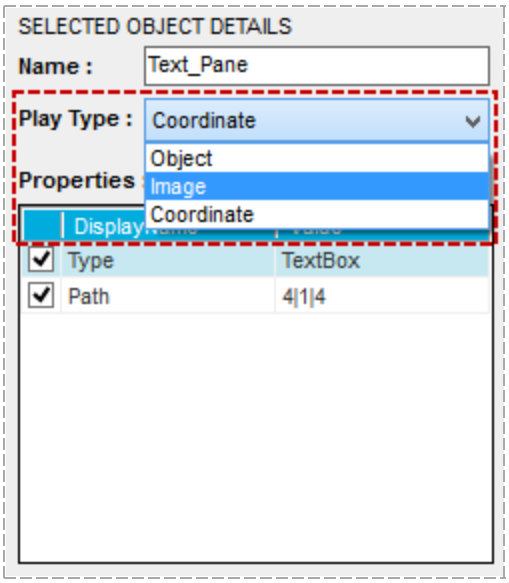
6. The Object List is populated as per the View selected.



7. 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.

8. Select a Play Type - whether Object, Image or Coordinate.



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 object are dynamic in nature; in the sense that they keep shifting their positions in the target application.
- **Image -** Use this as the play type for objects selected on the basis of its image properties. This could be useful when the selected object might shift in the target application.
- **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.

9. Select Properties type that you wish to use during play time. You can choose to view select properties or view them all. Some properties are selected by default.

   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.

10. 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 Calibration.

Previous and Next

7. Calibrating MetaBot Screens

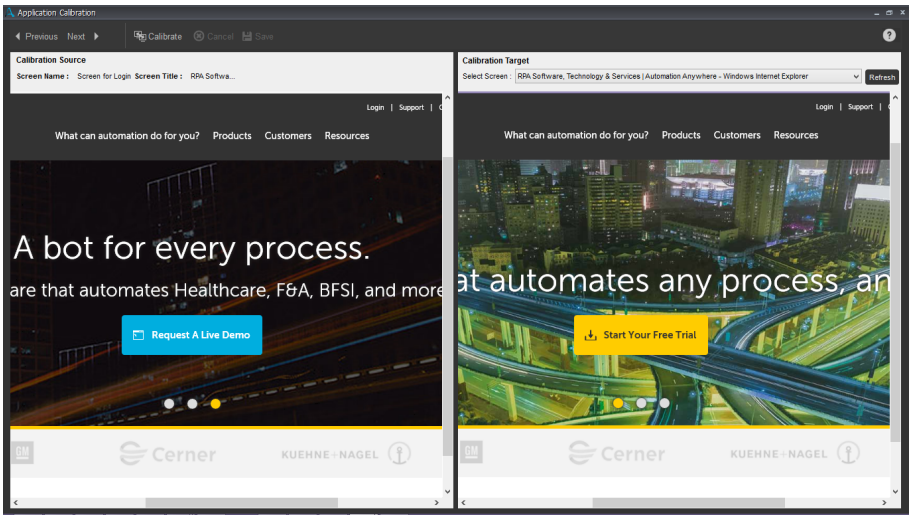
An application can undergo continual change during its life-cycle with improvements and newer features. MetaBot Designer's Calibration feature allows you instantly compare an existing screen with newer screen to identify those changes.

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.

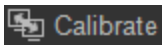
Guide to Calibrating a Screen

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

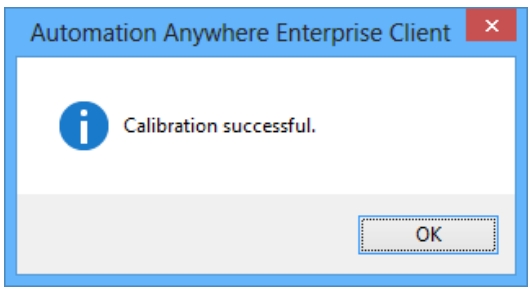
- The calibration screen is divided into two panels: The one on the left - Calibration Source indicates your current screen (which needs to be calibrated) and one on the right Calibration Target (usually the latest screen) .



3. In the Calibration Source panel, the screen that has been selected from the Metabot is displayed.
4. In the 'Screen to be Calibrated' panel select the control that requires calibration and hit 'Calibrate'.

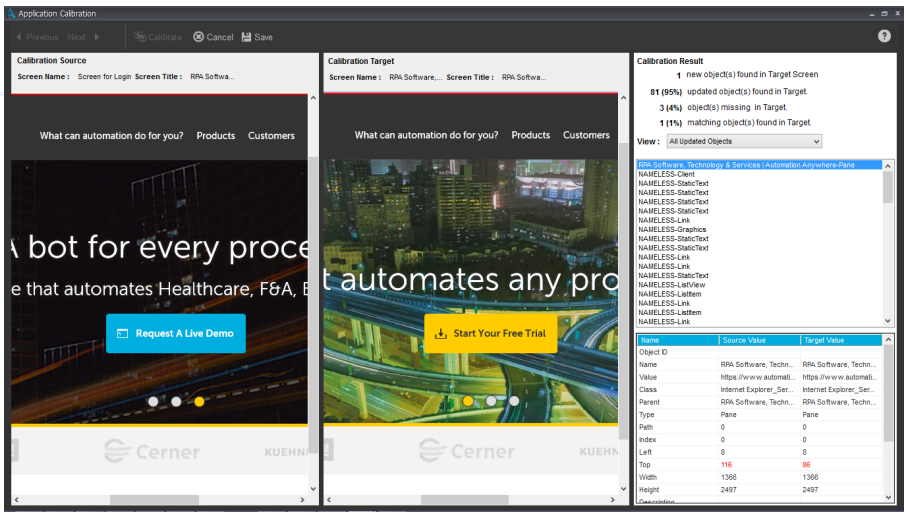


- You know the Calibration is successful when you encounter:

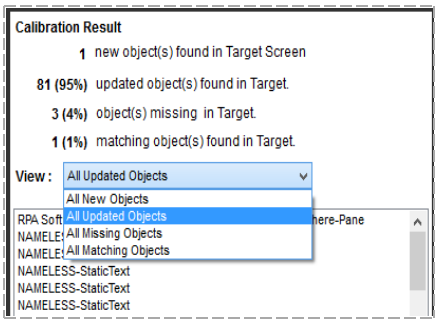


Note: You cannot calibrate screen that pop-up or do not have a title.

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



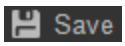
Here you can choose to view calibration in four 'Views':



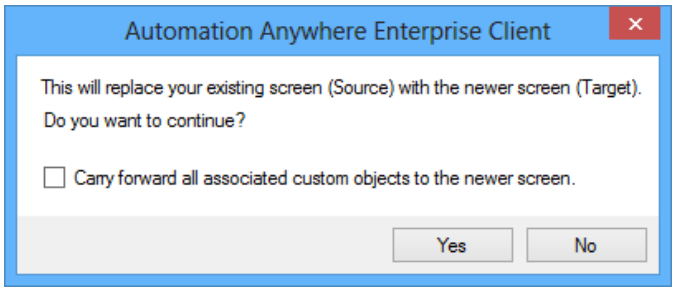
> Note: MetaBot Designer provides you quick summary (in numbers and percentages) of the newly added objects, updated ones, missing from targeted screen and matching objects that were found.

- 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.



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

8. 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".

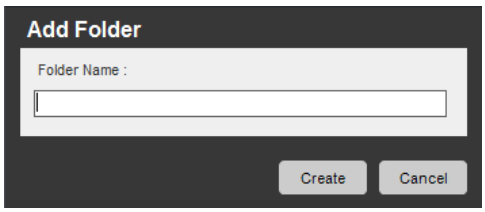
MetaBots can have any number of folders. You can also perform certain operations on a folder i.e. configure, calibrate, upload and delete.

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

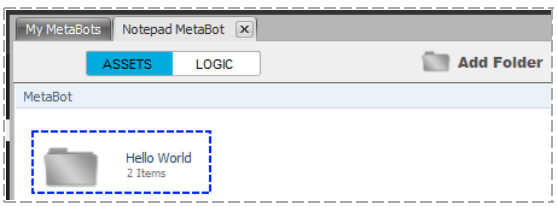
Guide to Creating a Folder

1. In the MetaBot Designer, click on Add folder

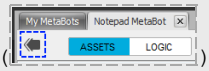
2. In the 'Add folder' window specify the folder name. Ensure that you use a name that reflects the type of folder created. Save by clicking 'Create'



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



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

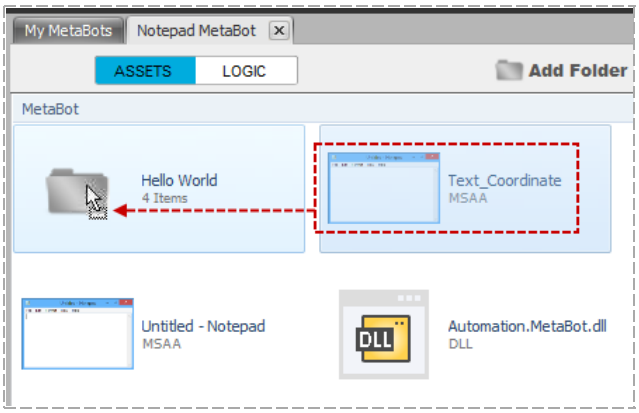
Note: Use the back button (

)if you wish to return to the main/previous window.

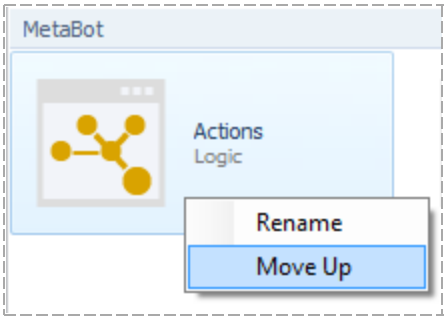
Moving Assets and Logic within Folders

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.

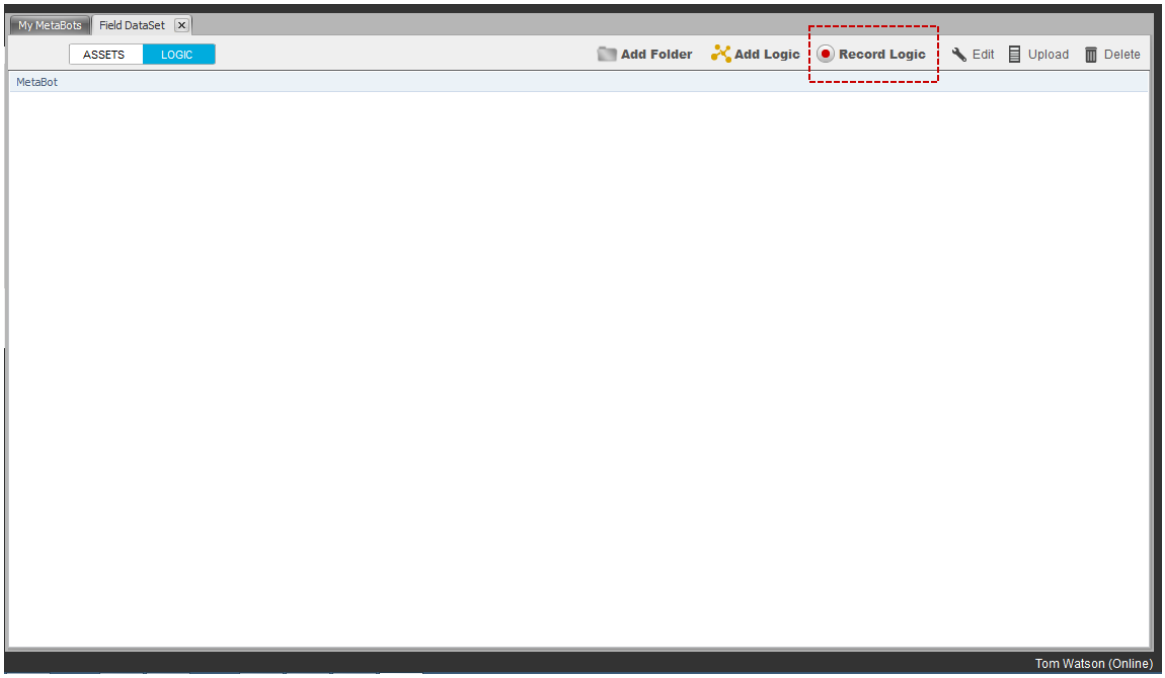
8. Recording Logic

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.

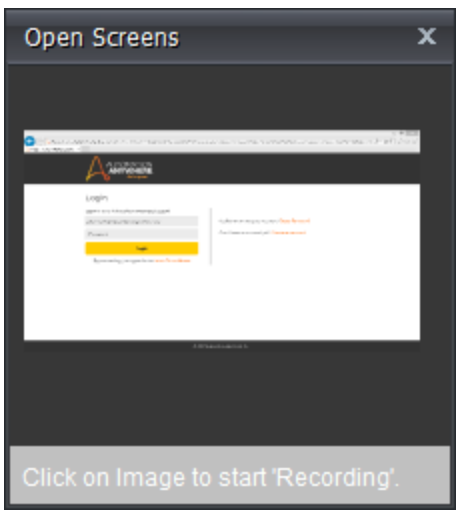
This topic describes the second option.



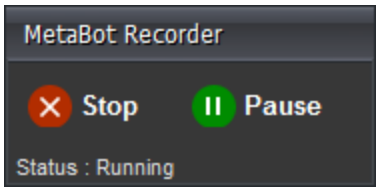
How to Record Logic

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.

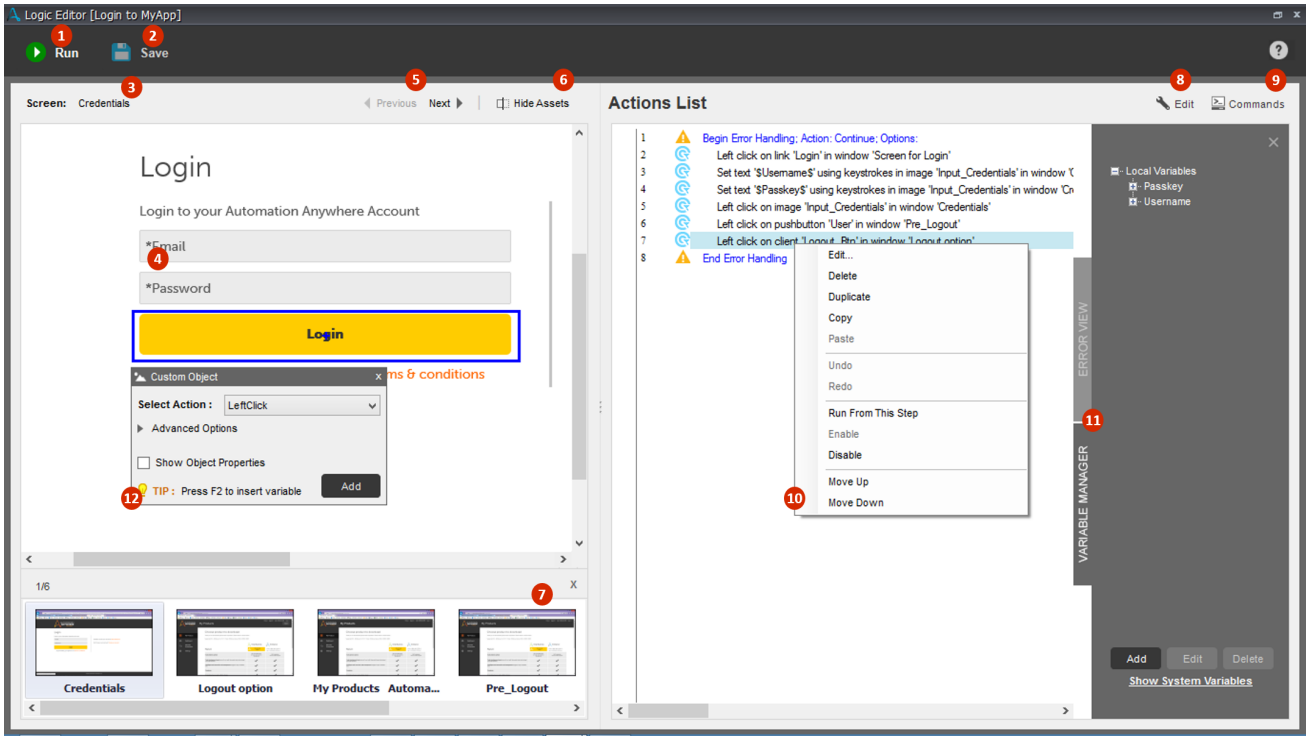
> Note: The MetaBot Recorder window displays the status of the current action being performed.

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.

9. Using the Logic Editor

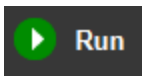
Use Logic Editor to create simple manageable independent navigational flow - Logic that can be integrated into automation tasks as and when required. Logic is a pre-configured use case of an application that leverages Assets (Screens and Dlls).



The Logic Editor

Use the following components to create a Logic.

1. **Run -** Use this to run the Logic Block in the Logic Editor view.

2. **Save -** Save a new or an edited Logic Block

3. **Screen -** Displays name of the selected screen provided during 'Configuration'.

4. **Captured Object -** This space displays the captured Assets (Screen or Dll).

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

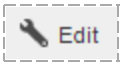
   

6. **Show/Hide Assets -** A toggle control used to display or hide captured Assets(Screens and Dlls).

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

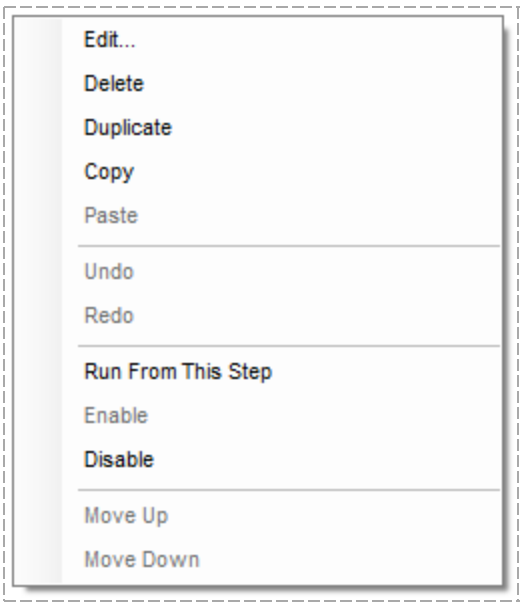
8. **Edit button -** Use to edit an existing Command or Action.



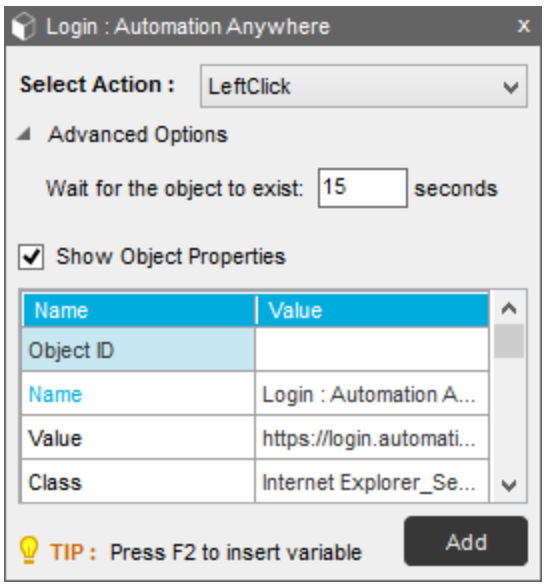
Note: 'Run', 'Save' and 'Edit' are enabled when an Action/Command is configured.

9. **Commands button -** Use to add Commands to the Logic Block.

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



11. **Variable Manager/Error View -** Use the 'Variable Manager' to manage variables and the 'Error View' for viewing and fixing task errors.
12. **Properties Window -** Use the floating 'Properties Window' to add and/or update actions.



Building Logic

A Logic is a combination of 'Commands' and 'Actions'. These, you can add with the help of Commands button.

1. Adding Commands

Combine any of the Commands to build your Logic:

a. **Clipboard -** Use this to perform certain activities with Windows clipboard.
Learn More

b. **Message Box -** Use this to insert a message box in the Logic.
Learn More

c. **Comment -** Use this to provide additional information about the Logic in the form of comments.
Learn More

d. **If/Else -** Use this to add conditional logic and actions to the Logic Block.
Learn More

e. **Error Handling -** Use this to handle errors while running a Logic, to aid in debugging.
Learn More

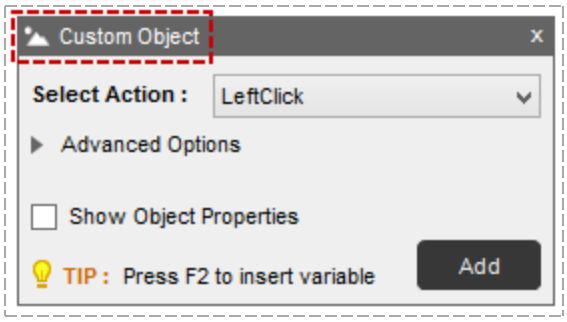
f. **String Operation -** Use this to manipulate a text string or extract part of a string and store it in a variable.
Learn More

g. **Variable Operation -** Use this to assign variables to the Logic.
Learn More

2. Adding and Updating Actions

To Add and/or Update various 'Actions' to build your Logic, select relevant action in the floating Properties window that appears when you click on an object.

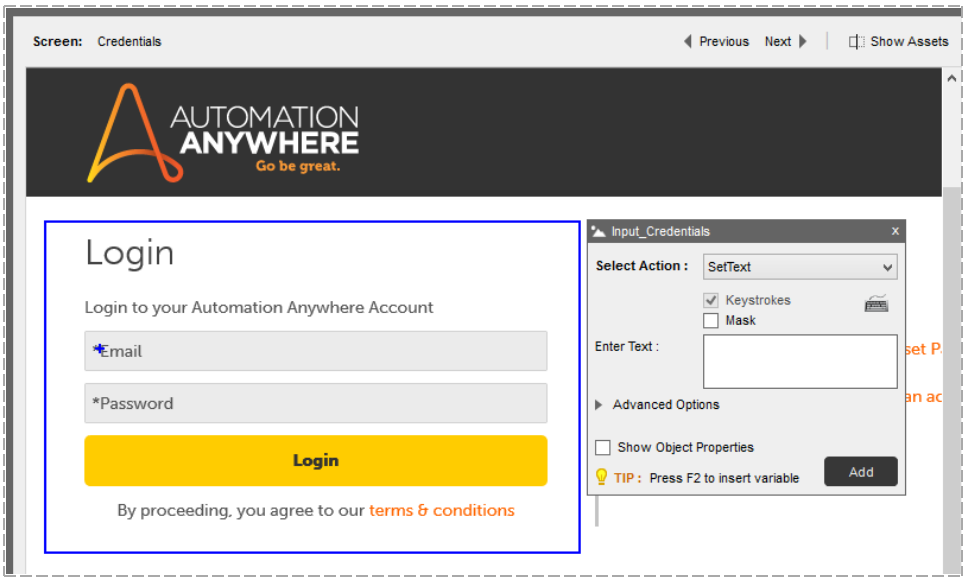
Tip: You can identify the 'Object Type' and it's 'Play Type' based upon the icon that appears in the title bar of the Properties window.



Actions depend upon the Object and Control Type selected for 'Screen' and Class and its API selected for 'DLLs'.

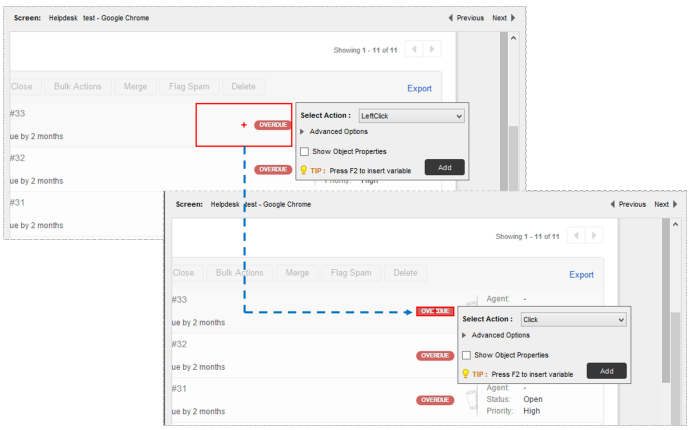
Also, the properties and relevant actions are controlled by the 'Play Mode' that has been selected while configuring the screen.

- To Add an Action, click on the captured Screen/Dll.

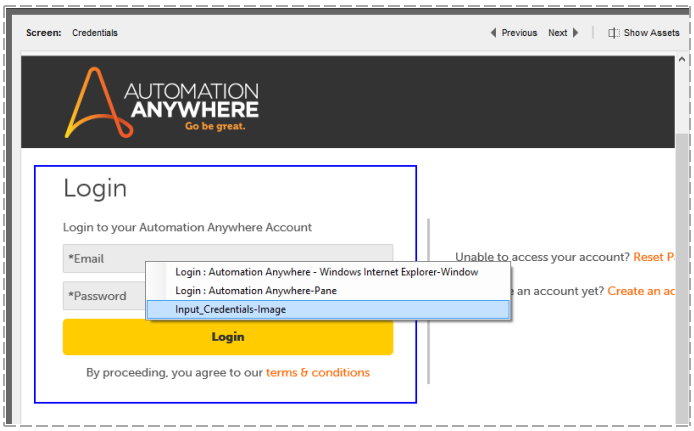


> Note: Custom objects are highlighted in blue outline.

- Drag the crosshair to select the precise location where you want to perform an action. Refer screenshot below:



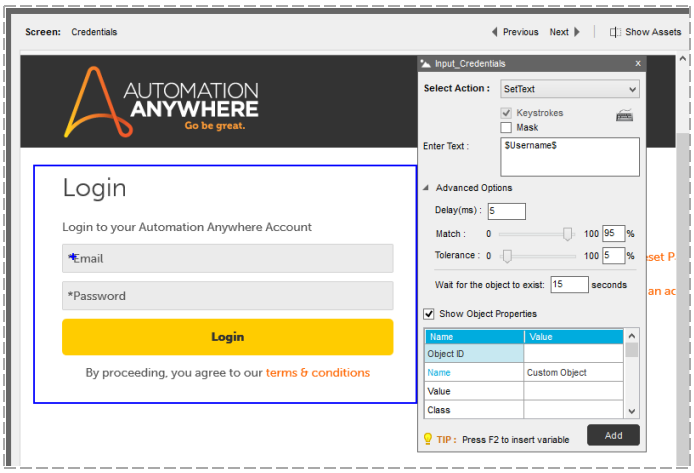
- To view a list of objects that are neighboring to the selected object, right click. This helps in selecting the precise object control based on its parent object. Use this option when you are unable to select a particular control on the screen.



- To add 'Delay' and/or 'Wait for Object to exist option' click on the 'Advanced Options' tab.

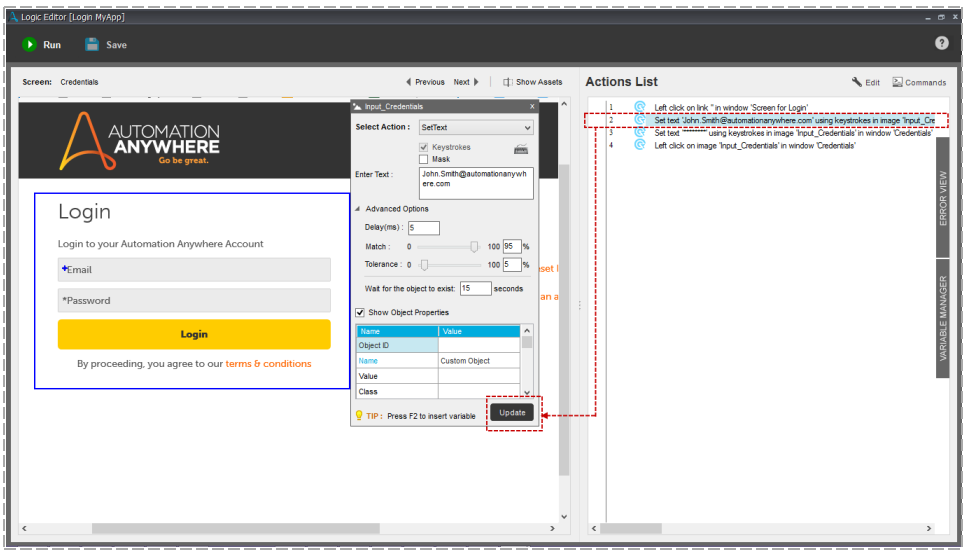
> Note: 'Delay' option is available on selection of the 'Keystrokes' action as it allows you to accommodate the time required between firing of two separate keystrokes.

- Also, to view the properties of the captured object, select the option 'Show Object Properties'

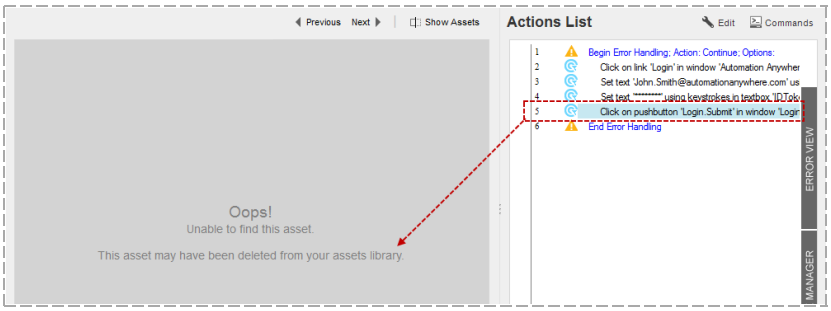


> Note: The default search criteria are displayed in blue font and can be variablized. To change the default search criteria for an object, use 'Configure'. Learn More

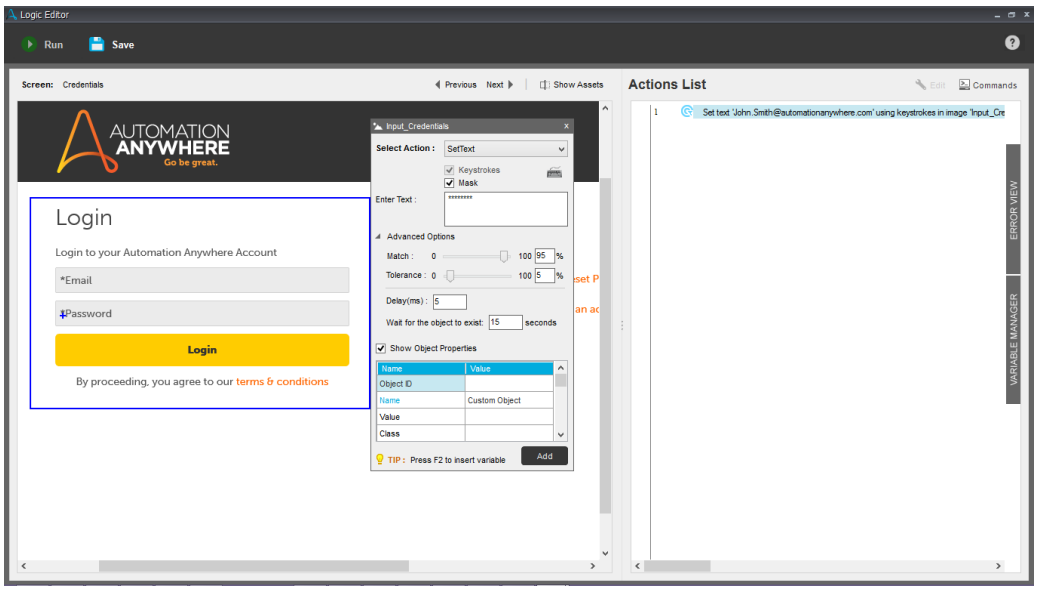
- To 'Update' an Action, double click on the event data in the Actions List or click the 'Edit' button.



Note: If the Asset that you navigate to is deleted, you are provided appropriate message.



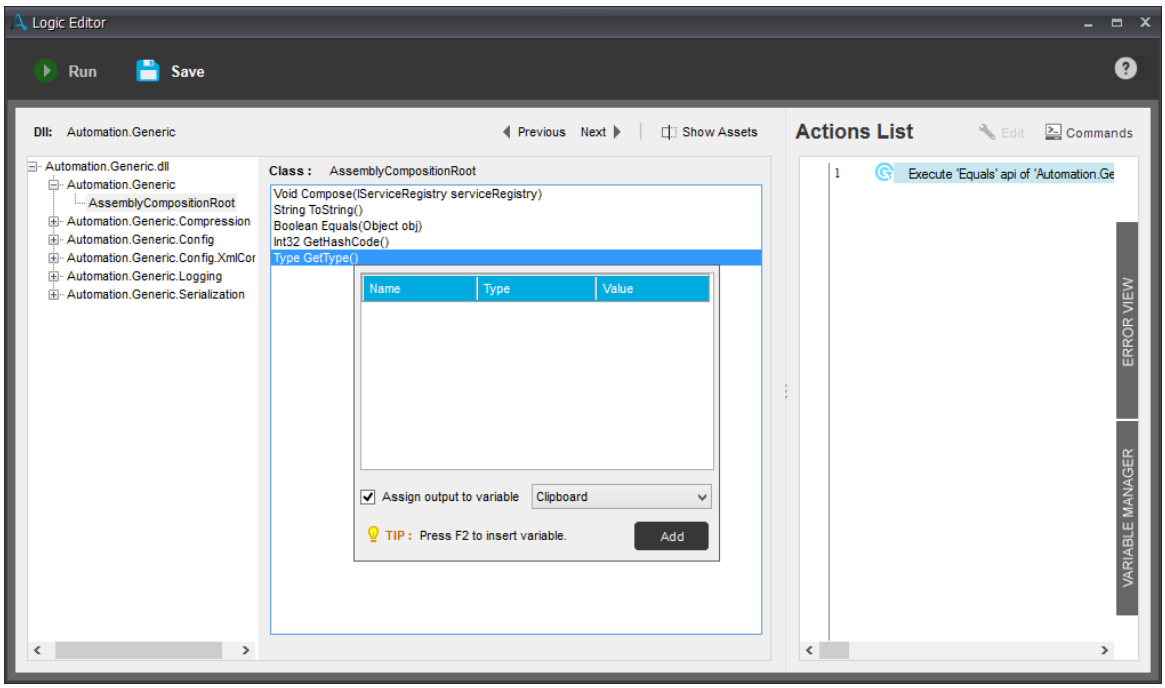
A. Screen Actions



Actions that are allowed on **HTML Controls**:

1. Click/Double Click/Right Click/Left Click
2. SetText/AppendText - Use these when the selected object require firing keystrokes.
   Learn More
3. GetProperty - Use this action when you want to search the objects based on their properties during play time.
   Learn More
4. GetVisibility - Use this action to build a logic based on a screen area's visibility during play time.
   Learn More
5. Wait
6. Drag and Drop
7. GetTotalItems
8. GetSelectedIndex/Text
9. SelectItembyText/Index
10. GetChildrenName/Value

B. DLL Actions



You can use a DLL in your Logic in the Logic Editor just like you would use a Screen.

1. Select a Dll from the Assets carousel (available when you click 'View Assets').
2. This will list one or more DLLs in the left pane.
3. Expanding your desired DLL will reveal its namespaces and related classes.
4. Select a class from the list. The supported APIs for the selected class are displayed on the right pane.
5. From the list of APIs select the one for which you need to input a value.

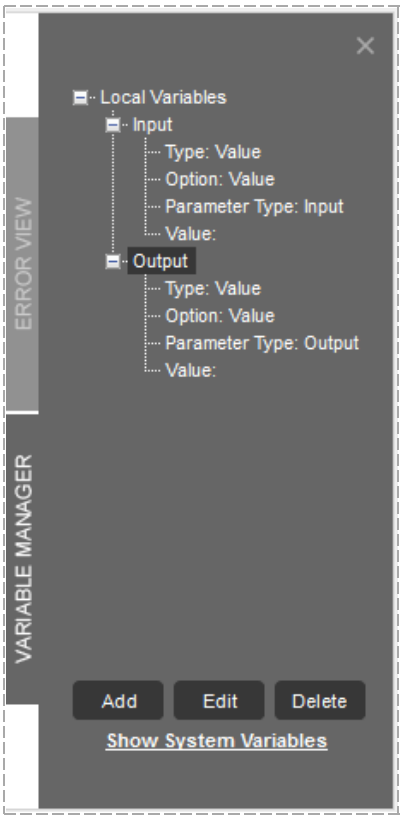
> Tip: Use an Array type variable to input values if you want to assign multiple values to a single parameter. Learn More

6. You can also choose to assign the output value to a variable.

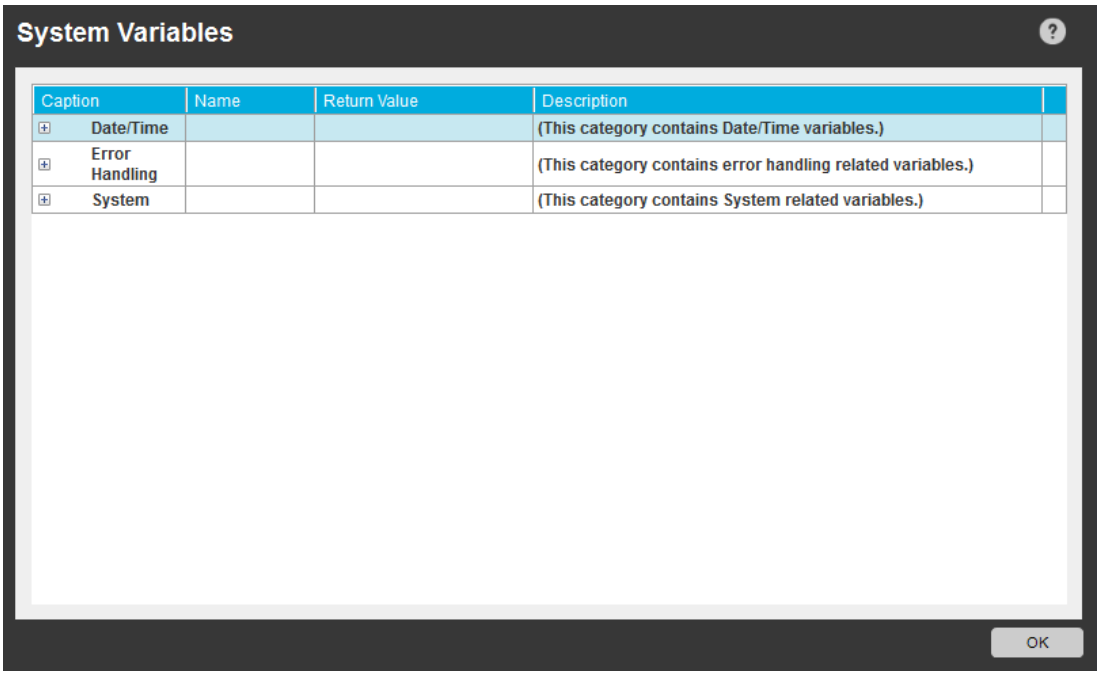
3. Adding and Editing Variables in the Variable Manager

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

- 'Add', 'Edit' or 'Delete' variables from the Variable Manager in the Logic Editor. Learn More

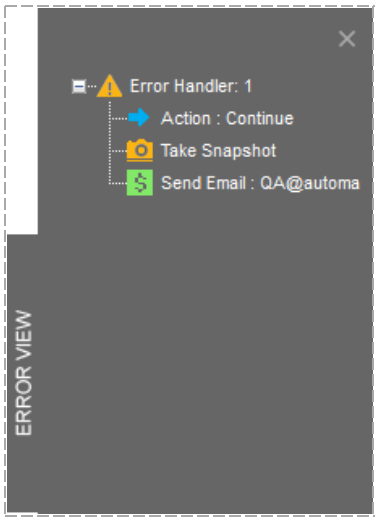


- Also assign System Variables, if required, while configuring the block.
- **System Variables -** MetaBot Designer provides powerful pre-defined system variables that you can use to automate common processes. Learn More

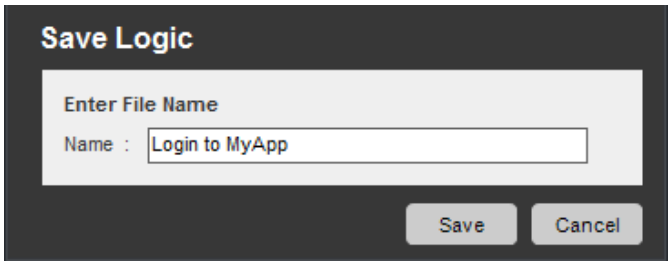


4. Using the Error View

Use the Error View to manage errors that occur in your Logic Block. Use the Error Handling command to manage those.



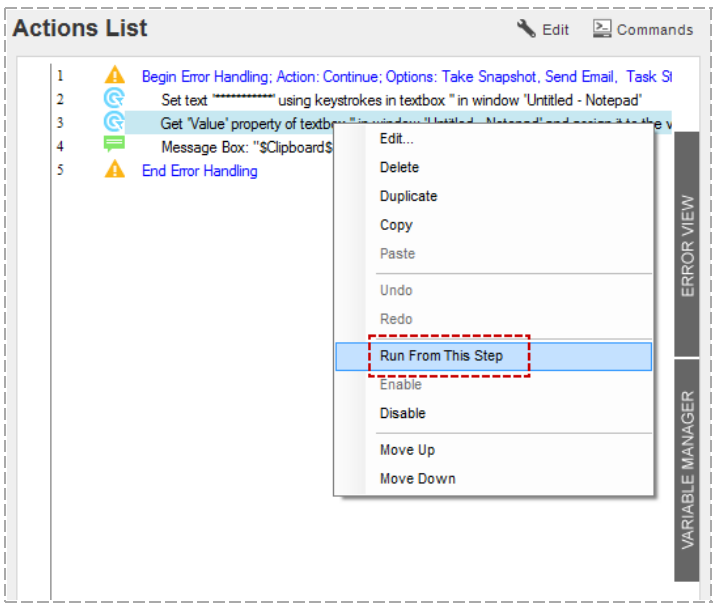
Save the Logic.



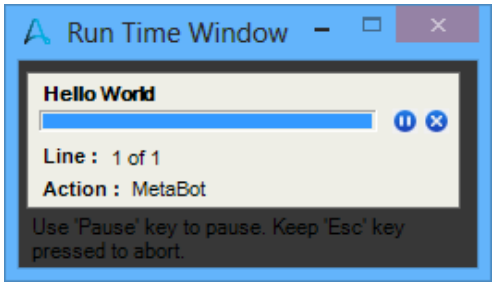
5. Running the Logic

Click 'Run' when in Logic Editor view. You need to verify whether your Logic runs as required.

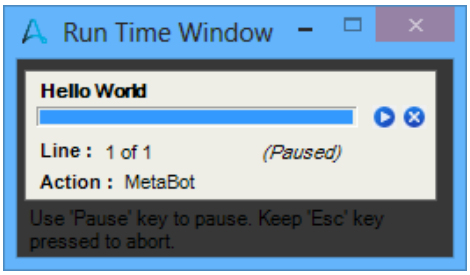
Tip: Use 'Run from this step' option if you want to skip the earlier steps while during a test run.



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

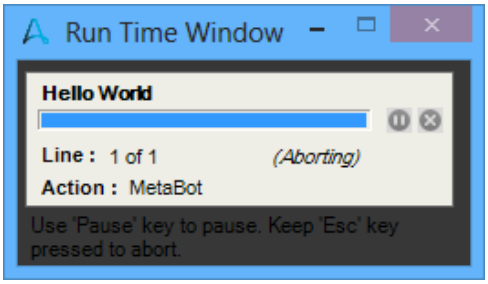


- Click the Pause icon to pause the Logic run:



Tip: 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

10. Selecting Actions in the Logic Editor

You can select various 'Actions' in the Logic Editor 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:

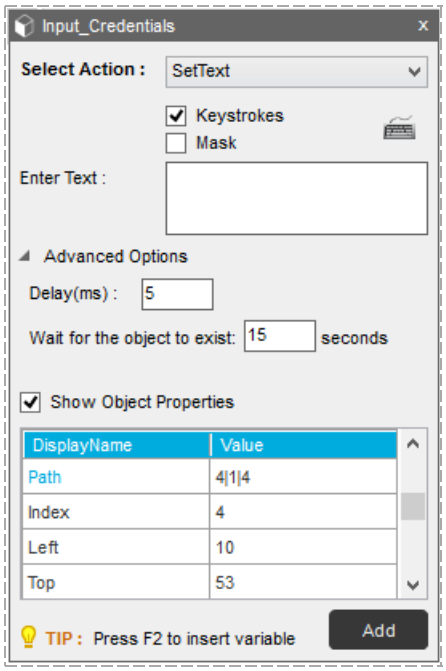
1. Click
2. DoubleClick
3. Right Click
4. Left Click
5. SetText
6. AppendText
7. GetProperty
8. GetVisibility
9. GetTotalItems
10. GetSelectedIndex
11. GetSelectedText
12. SelectItembyText
13. SelectItembyIndex
14. GetChildrenName
15. GetChildrenValue

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

1. 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.

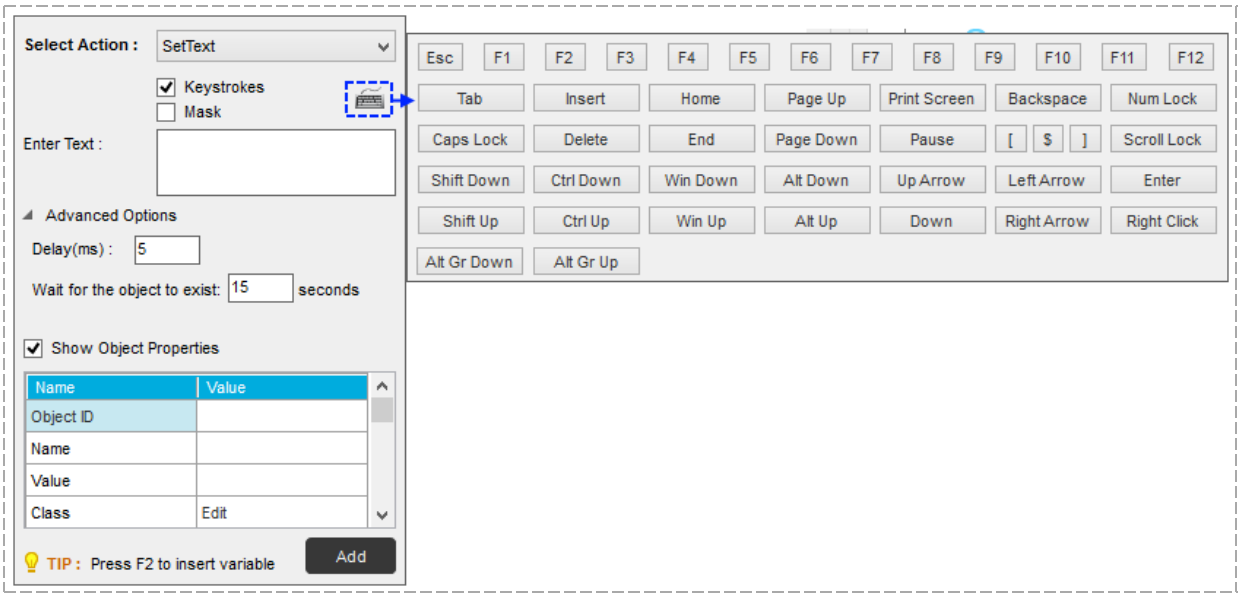
While building your Logic, remember that the properties and relevant actions are controlled by the 'Play Type' configured for the selected Screen.



- 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.

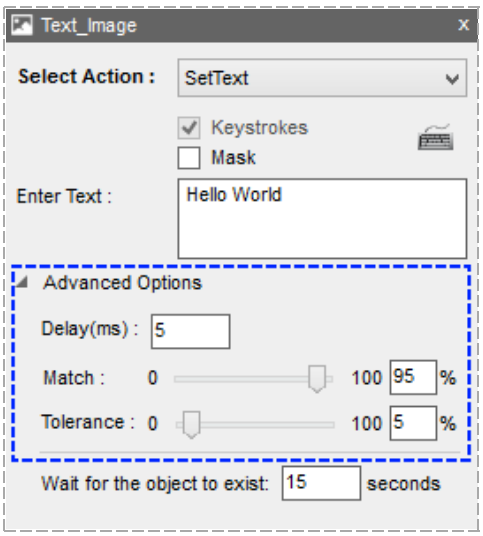
- Use virtual keyboard to enter certain special keyboard keystrokes:



- 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.

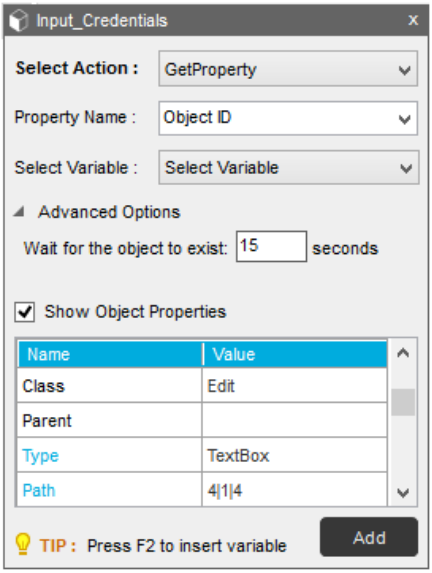


A Match value defines the extent of overall mismatch allowed between Image1 and Image2; Tolerance defines the extent of mismatch allowed between any two pixels under comparison.

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.

2. GetProperty

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



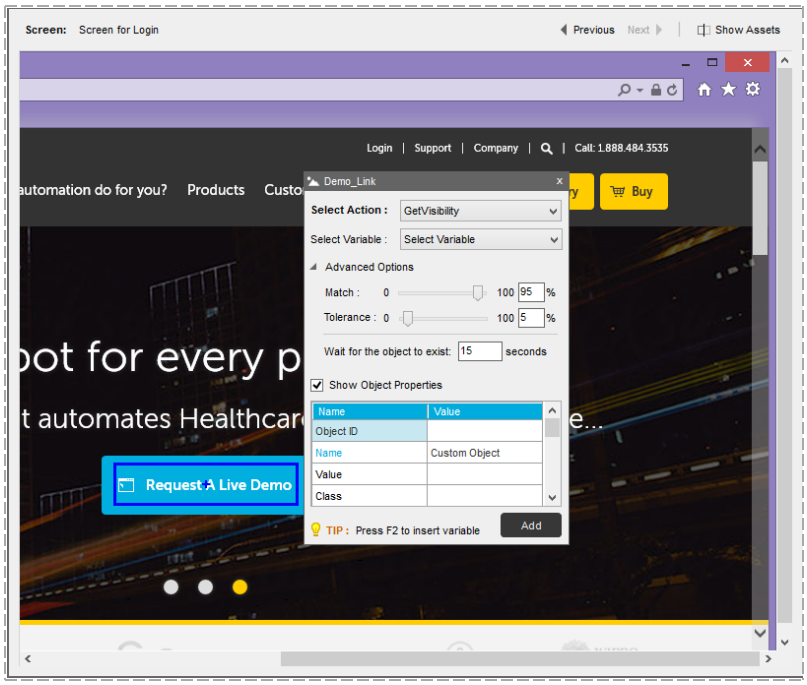
When you select action as 'Get Property', you will be able to select properties names such as Object ID, Name, Value, Class, Type, Index, Description, State, IsVisible, IsProtected etc based on the object control selected.

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'.



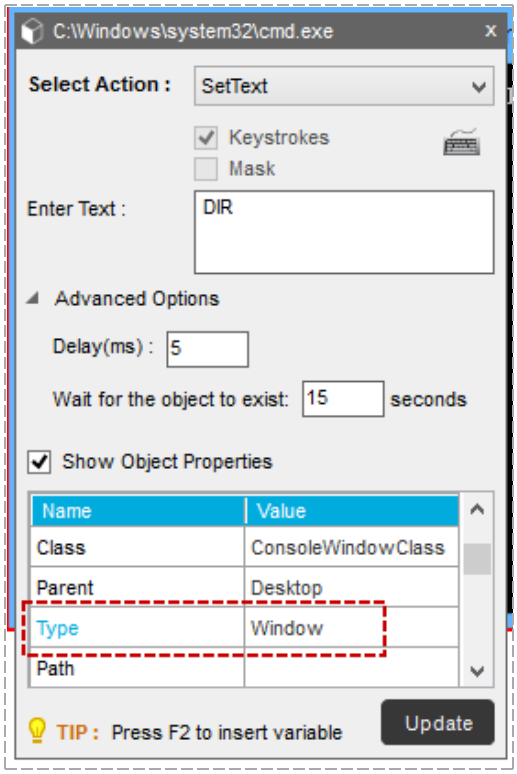
Tip: GetVisibility can be combined with conditional commands such as If.

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

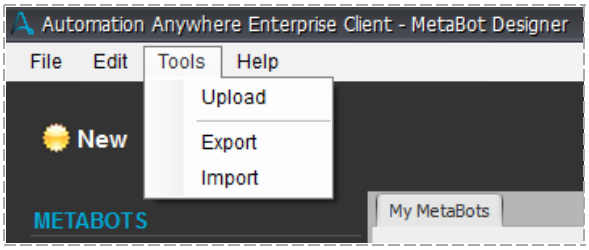
SetText for Window Control - Use the action type 'SetText' for Window Controls. Select the entire window and specify the action type.



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

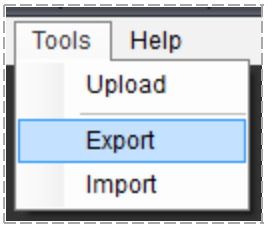
11. Exporting and Importing MetaBots

MetaBot Clients can export and import MetaBots to be able to use those in different Control Room setups. It means that you can use them in certain restricted environments such as automation setups that are disconnected/independent of each other.



Exporting a MetaBot

To Export a MetaBot to another Control Room setup go to Tools options and select 'Export'.



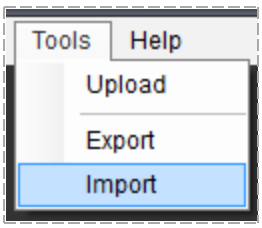
You can rename the MetaBot and save it to a location of your choice.

Note:

- The MetaBots are saved as 'MetaBot Files' with an extension '.mbot'.
- You can also send all exported MetaBots to a zipped folder.

Importing a MetaBot

To Import a MetaBot to another automation setup, go to Tools and select 'Import'.



Select a MetaBot from the file location.

Note: If a MetaBot of the same name exists in MetaBot Designer, it will be overwritten on confirmation.

Section 2: MetaBot Designer - Commands

12. Clipboard Command

Automation Anywhere provides a system variable -*$Clipboard$*, that you can use to retrieve text that has been copied to the Clipboard.

Use this to perform certain activities with Windows clipboard in the Logic.

Sub-Commands

The Clipboard command provides the following sub-commands:

- Clear Clipboard
- Assign to Clipboard
- Assign from Clipboard

Using the Sub-Commands

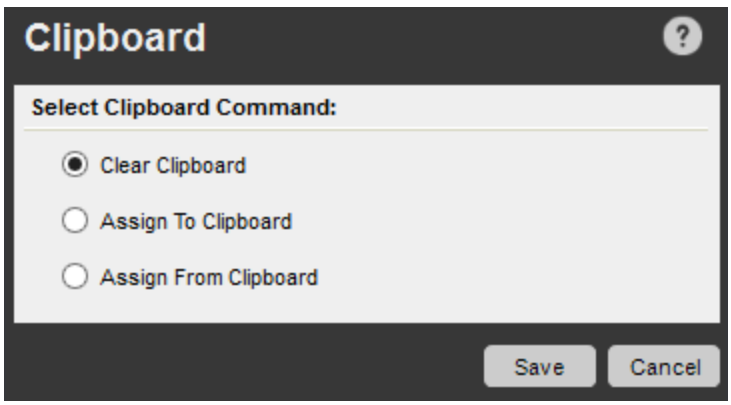
To insert a Clipboard command in your task,

1. Open the Logic Block in the Logic Editor.
2. Click on the Commands button.

3. Specify the required parameters in the fields of the Clipboard command window.
4. Save the command to your Logic Block.

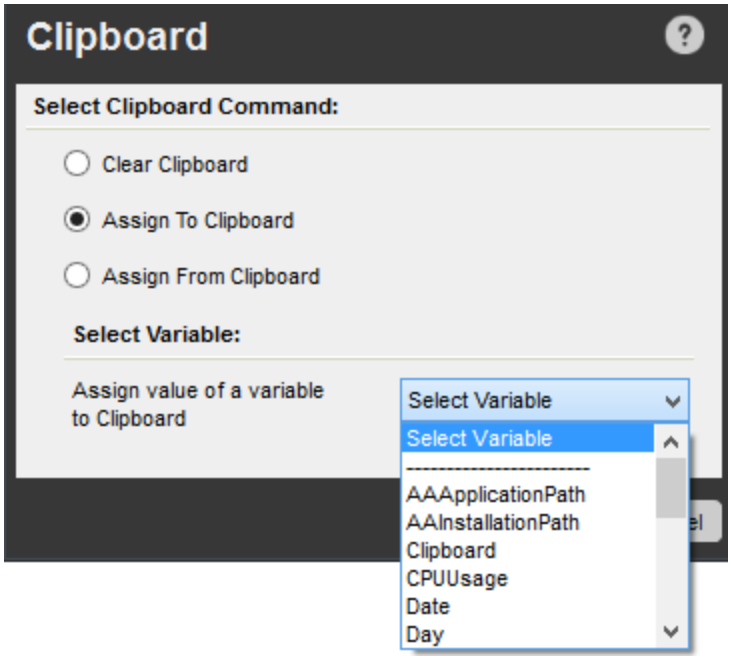
Clear Clipboard

Use this command to clear the contents of the Clipboard.



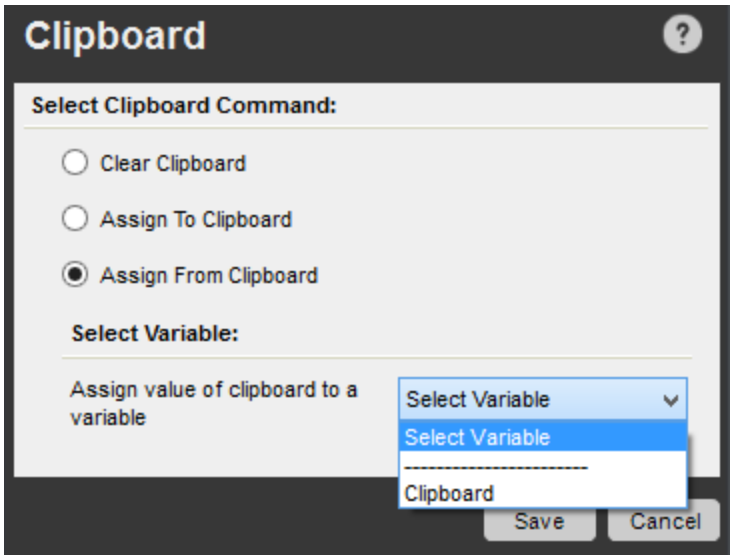
Assign To Clipboard

Use this command to assign the value of any variable that has been created using the Variable Manager to the Clipboard. You can then access this value by using the *$Clipboard$* system variable.



Assign From Clipboard

Use this command to assign the value contained in the Clipboard to any value type variable that has been created using the Variable Manager.



13. Message Box Command

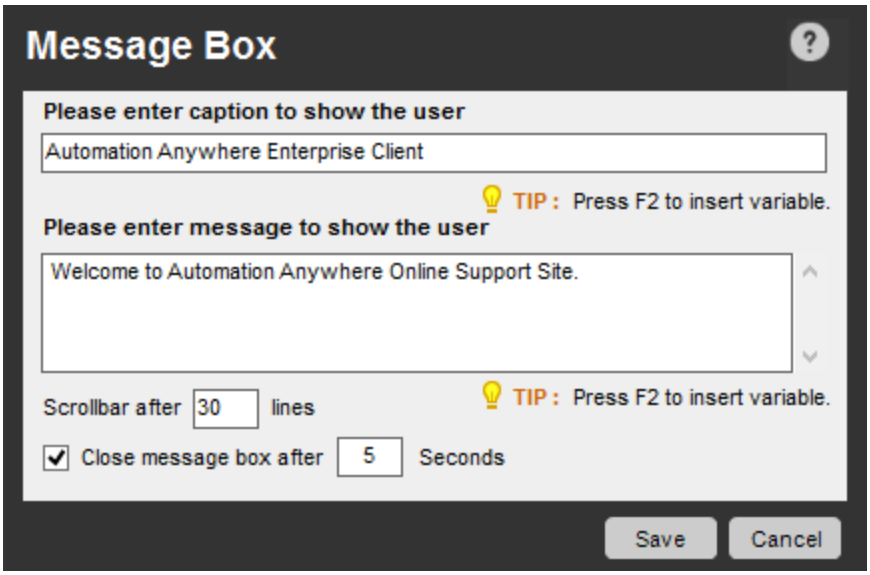
The Message Box command enables you to insert a message box in a Logic Block when you want to display a message to the user when the task runs. You can specify a custom caption for the message box.

Inserting a message

To insert a message in your Logic Block,

1. Open the Logic Block in the Logic Editor.
2. Click on the Commands button.

3. Specify the following components in a message box:
   - **Caption:** Enter a caption name for your message box.
   - **Message:** Enter your message in the dialog box.
   - **Scroll bar:** Insert a scroll bar in your message. The minimum message length for a scroll bar to be displayed is 10 lines.
   - **Close Message Box:** Enter the seconds value after which the message box should close.



Common Use Case: Message boxes are useful for issuing a message to the user whenever the task concludes running. For example, when a web form task completes a run, you might issue a message that states: "Web Form Filled and Complete."

14. Comment Command

Use this command to insert comments in your Logic Block to provide additional information. Comments are ignored when the task runs.

Comments are useful in annotating the task steps. Some people use comments to extensively document details about their tasks. Some people use fewer comments just as reminders.

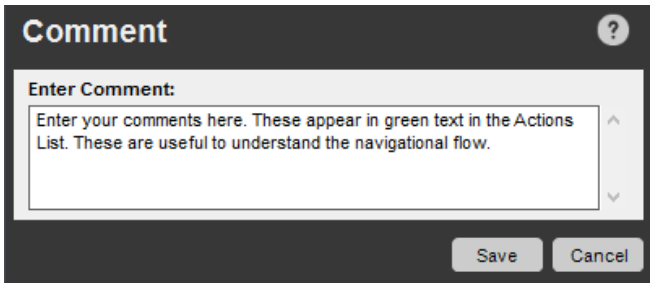
Inserting a Comment

To insert a comment,

1. Open the Logic Block in the Logic Editor.
2. Click on the Commands button.

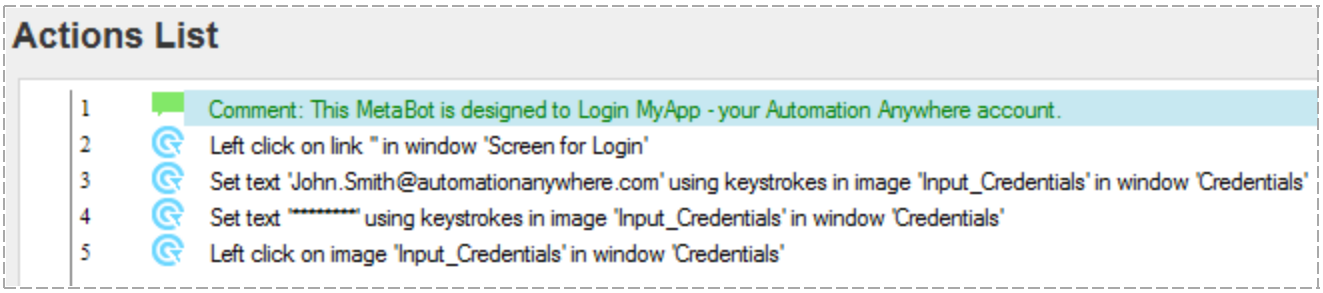
3. Type your comments in the Comment text box provided.
4. Click Save.
5. A comment is inserted in the Logic Block. You can copy this comment to add multiple comments, or move the comment to another position simply by dragging it.

The comment is displayed in green in the Actions List, and is always saved as a single line. Multiple-line comments are displayed as a single line when the comment is saved.



Note: Though a command, a comment line is not executed.

Sample Comments



15. If/Else Command

Use this command to add conditional logic and actions to your automated task.

Sub-Commands

The IF/ELSE command provides the following sub-commands:

- Window Exists / Window Does Not Exist
- Variable
- Else If
- Else
- End If

Overview

One of the most powerful features of Automation Anywhere is the IF/ELSE command. You can use the sub-commands to perform actions when certain conditions exist.

When using most of the conditional sub-commands, you can specify how long to wait for the condition to be true before taking another action.

Using the Sub-Commands

To insert an IF/ELSE command in your task, follow these steps:

1. Double-click or drag one of the **If/Else** commands to the Actions List pane. The IF/ELSE window is displayed.
2. Specify the required parameters in the fields.
3. Save the command to your Logic Block.

Window Exists/Window Does Not Exist Sub-Commands

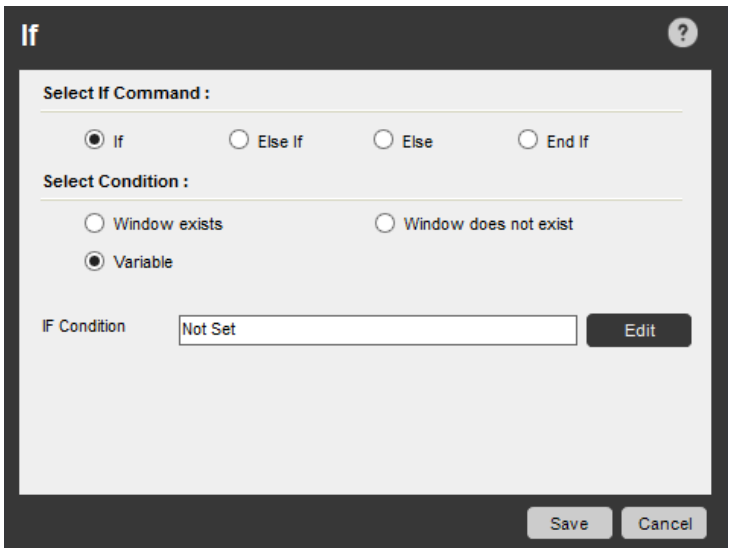
Use these commands to do the following:

- Check or verify that a specific application is open.
- Check whether an error has occurred.
- Check whether a file download has completed.

> Note: A window might open in a second or could take longer. To tackle this situation, use the "Window does not exist" option of If command.

Variable Sub-Command

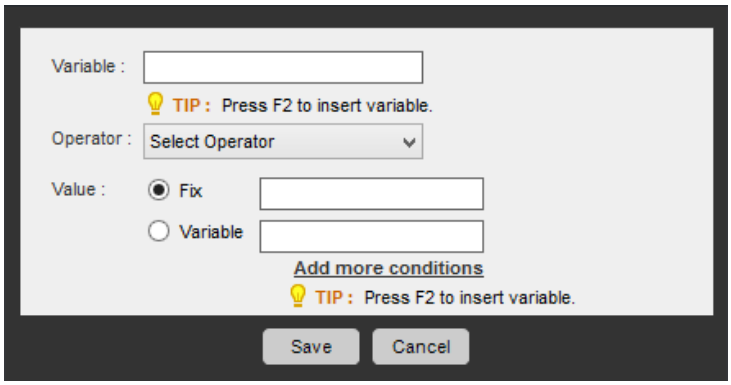
This command, though it looks simple, is perhaps the most powerful conditional command. Use this command to perform hundreds of checks.



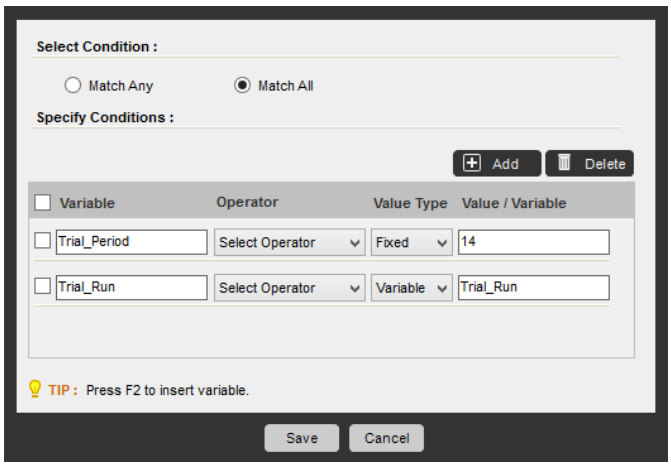
For example, you can copy data from an application or website and verify that the data you copied matches a specific text string. You can also verify a numeric string, or check to see if it is greater than or less than a variable or a fixed value.

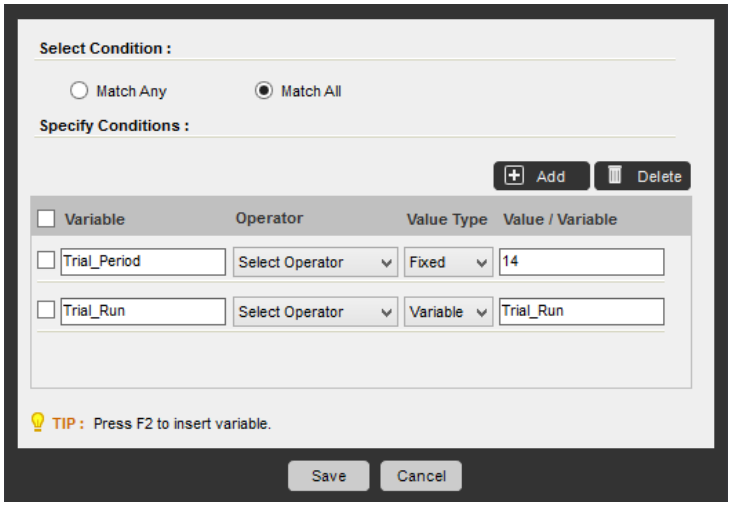
Using 'AND' or 'OR' conditions in IF Variable command

When you need to include the 'AND' or 'OR' conditions in conjunction with the 'Variable' sub-command, use the 'Add more conditions' option. It offers you the luxury of including all your AND or OR conditions within a single parameter, instead of specifying separate parameters for each condition



Enable the 'Match Any' option for 'OR' conditions or 'Match All' for 'AND' conditions (refer image below)





Note: You cannot include both; 'Match Any' + 'Match All' within a parameter. You are required to specify them within a task, separately.

16. Error Handling Command

Use this command to handle errors while running a Logic, to aid in debugging.

Overview

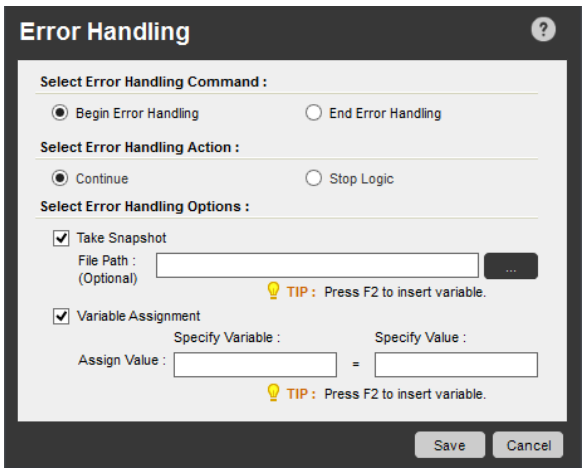
By using the Error Handling commands, you can isolate errors that occur when running your Logic.

When an error occurs, you can continue running the Logic or stop.

Sub-Commands

The Error Handling command provides the following sub-commands:

- Begin Error Handling
- End Error Handling



Error Handling Options

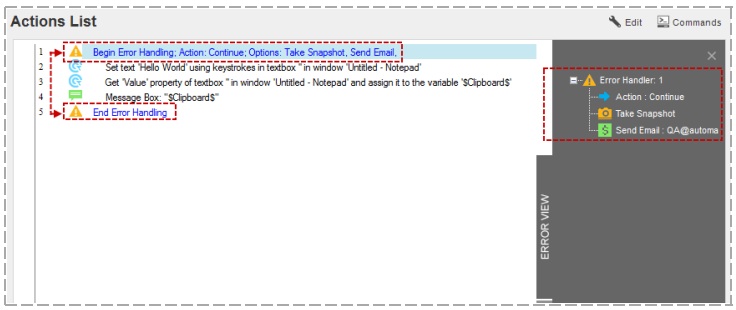
MetBot Designer provides the following options for error handling when an error occurs:

1. **Take Snapshot:** In case of any error, take a snapshot of the screen.
2. **Variable Assignment:** Specify a value to be assigned to a variable and Set Task Status Pass or Fail depending on the Error handling action.

   This is more useful when one wants to Continue running a Logic, and thus in that case the user wants to set a specific value to a particular variable.

Use any or all of these options in combination.

When any of the Begin Error Handling option is selected, End Error Handling is automatically inserted in the Logic Editor. However, one can also specifically select End Error Handling action when the related actions are completed.



You can verify the values of error handling in the 'Error View':

17. String Operation Command

Use this command to manipulate a text string or extract part of a string and store it in a variable.

For example, use it when you want to extract a portion of a text block from an email, a website, or an application window.

You can also manipulate strings after capturing them. You might search the string for particular phrases, compare two strings, convert a string to upper or lowercase, or obtain the length of a string.

Sub-Commands

The String Operation command provides the following sub-commands:

- Before/After
- Compare
- Find
- Length
- Lower Case
- Replace
- Reverse
- Sub-String
- Trim
- Upper Case

Using the Sub-Commands

To insert a String Operation command,

1. Open the Logic Block in the Logic Editor.
2. Click on the Commands button.

3. In the String Operation Command, select the relevant sub-command.
4. Specify the required parameters in the fields.
3. Save the command to your Logic Block.

Before/After Command

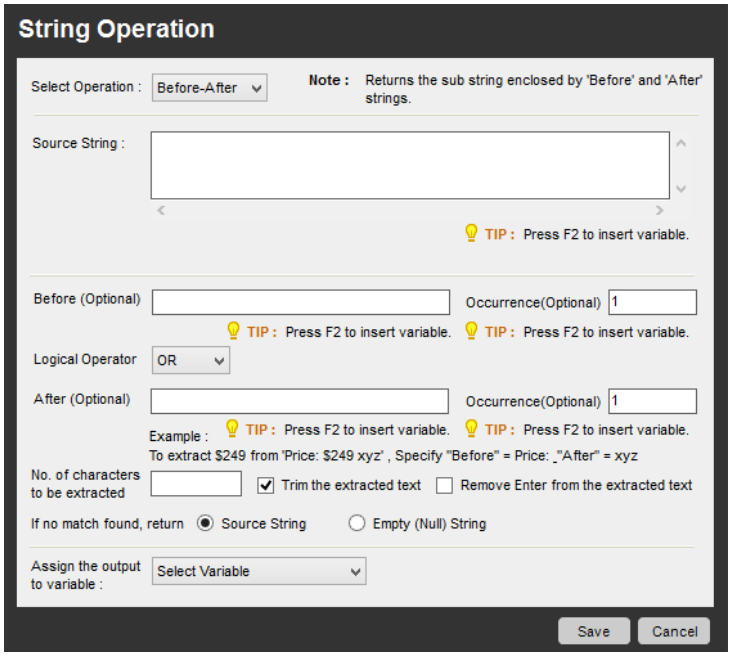
Use this command to specify a range of text to extract, using **Before** and **After** keywords.

You can insert 'Enter' and 'Tab' variables by using the F2 function key. These variables act as separators for your string between the **Before** and **After** keywords.

To refine your 'Before-After' string command, you can use the 'OR' or 'AND' Logical Operators.

- Use the 'AND' operator to ensure 'Before' and the 'After' conditions are met.
    - For instance, when you input a string in 'Before' and if not found, the search will be aborted. The program will not search the string input in 'After'.
- Use the 'OR' operator to ensure either 'Before' or 'After' condition is met.
    - For instance, when you input a string in 'Before' and if not found, the program will continue to search for the string input in 'After'.

Note: The default Logical Operator is 'OR'



Example: In the text string, *'Name of Applicant James Smith Applicant Location,'* let's copy only *'James Smith'*. We specify **Before** = *'Applicant'* with an Occurrence of 1. We also specify **After** = *'Applicant'* with an Occurrence of 1. This is the first occurrence after the **Before** keyword, although it is the second occurrence from the beginning of the string.

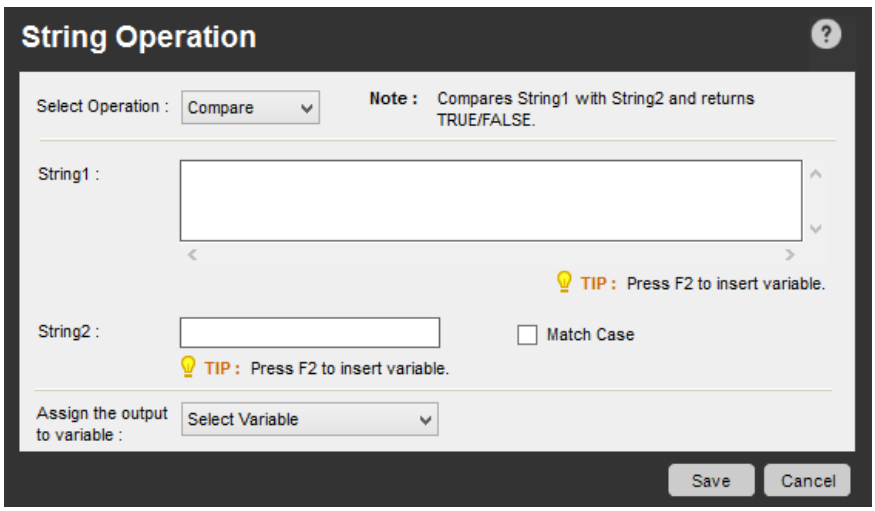
You can also specify the number of characters to be extracted, starting from the first character after the **Before** keyword. The extracted string can be trimmed to remove leading or trailing spaces, and you can save the string to a variable.

Tip:

- When using the **Before/After** command, enter a space as a separator between strings using the SPACE BAR key on the keyboard.
- In some cases, content extracted from websites are separated by an Enter symbol (a small square), which you can see in the CSV file.

Compare Command

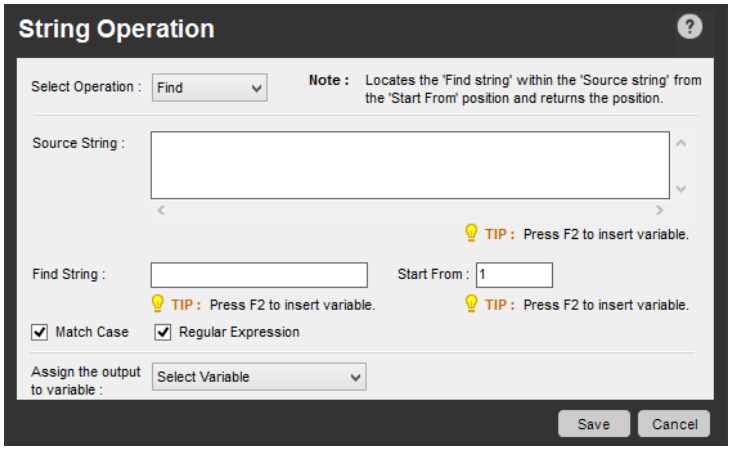
Use this command to compare strings by specifying the strings. You can refine your search by selecting the 'Match Case' option. A value of True or False is returned. You can assign the values that are returned to a variable.



Find Command

Use this command to locate a sub-string that exists within the source string by specifying the Start parameter.

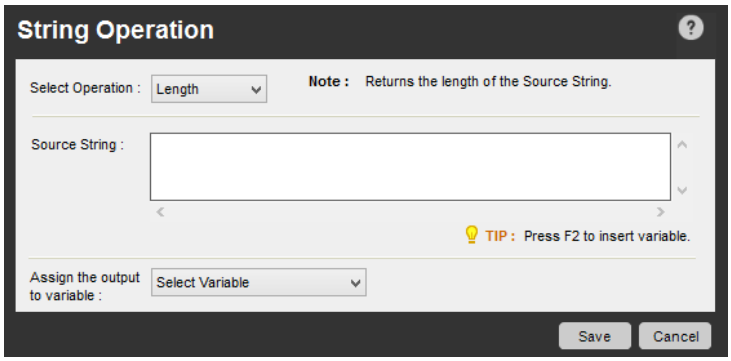
If the specified string is a regular expression, select the check box that is provided. Insert the regular expression in the **Find String** field. Also, refine your search parameters by specifying the line number in 'Start From', selecting 'Match Case' and/or Regular Expression.



For example, to find all email addresses in the source, specify the following as a regular expression: `\b[A-Z0-9._%-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\b`

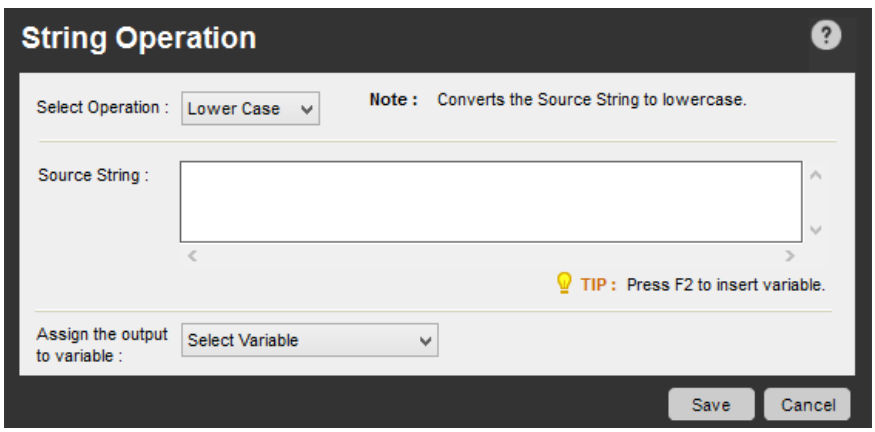
Length Command

Use this command to obtain the length of a string by specifying the source string.



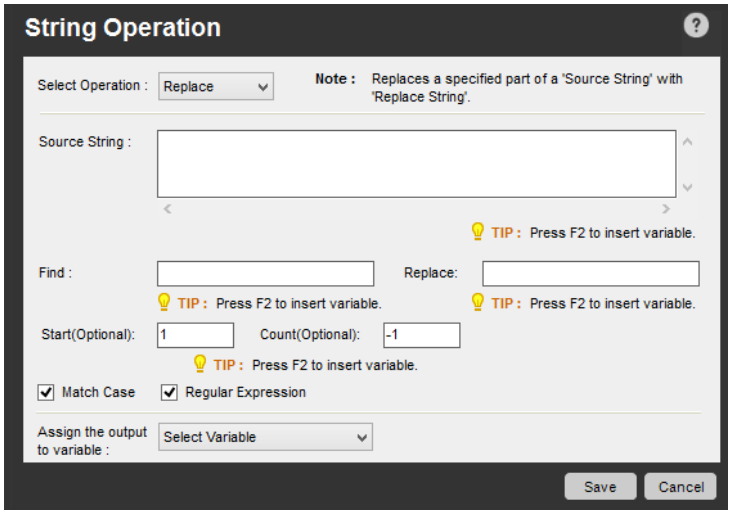
Lower Case Command

Use this command to convert a source string to lower case .



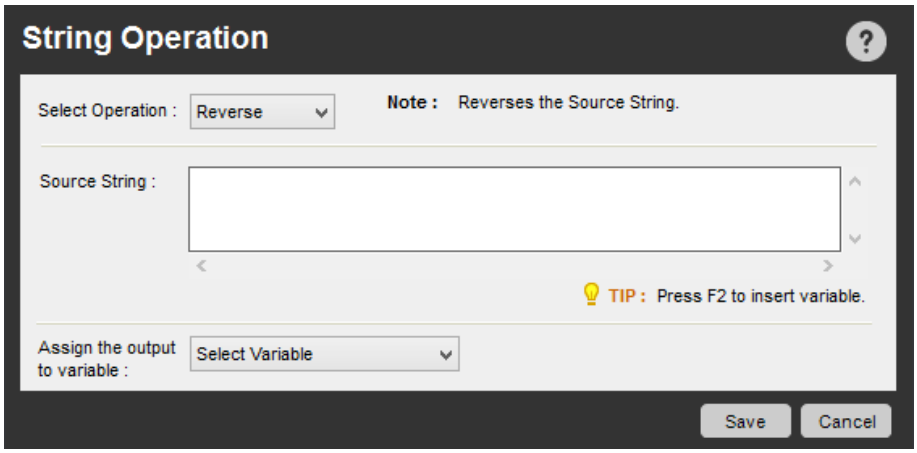
Replace Command

Use this command to replace a portion of a source string with a specified replacement string. Optionally specify the character position from which to begin in 'Start' and the number of times it is to be replaced in 'Count'. If the Find String field contains a regular expression, select the Regular Expression check box.



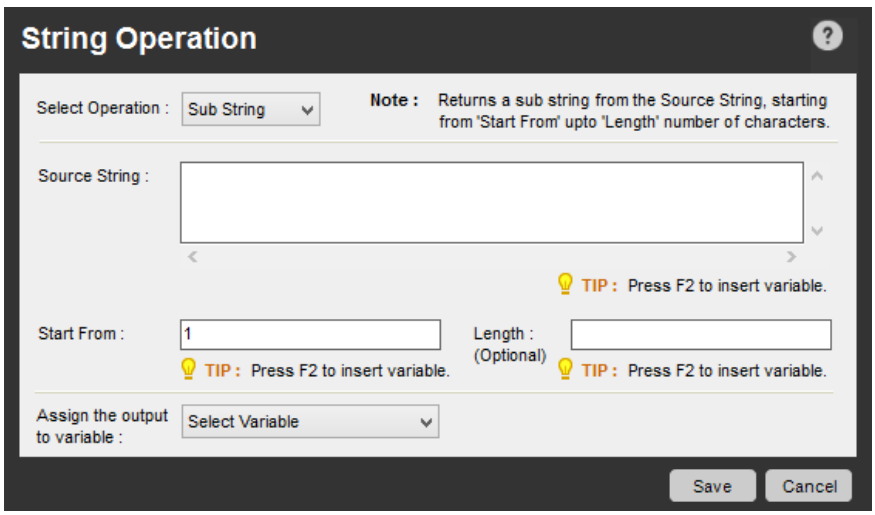
Reverse Command

Use this command to reverse a specified source string.



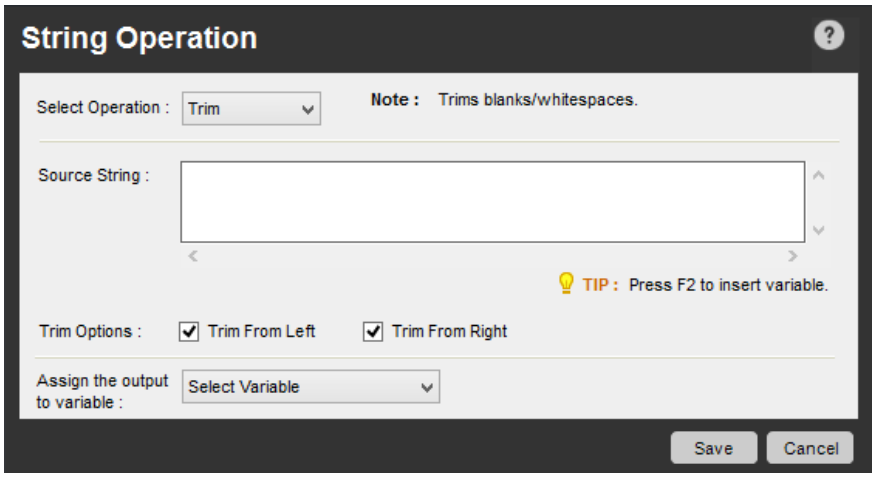
Sub String Command

Use this command to retrieve a sub-string. Specify the source string and parameters such as 'Start From' and optionally 'Length'.



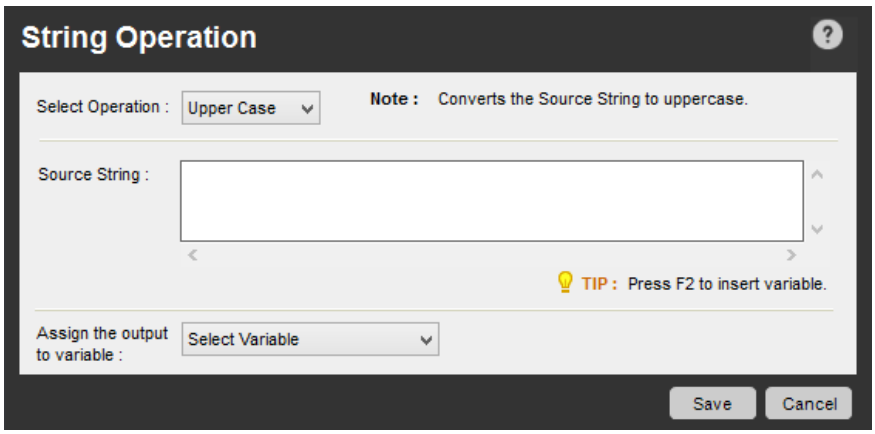
Trim Command

Use this command to trim blanks and spaces from a specified source string.



Upper Case Command

Use this command to convert a source string to upper case.



18. Variable Operation Command

Use the Variable Operation command to assign user specified variables (user variables) in a Logic Block.

The type and source of variables used dictate the assignment. These are applicable during task execution thus allowing the user to reuse the user variables.

Using the Variable Operation Command

To use the Variable Operation command,

1. Open the Logic Block in the Logic Editor.
2. Click on the Commands button.

3. Select either User Variables or System Variables.

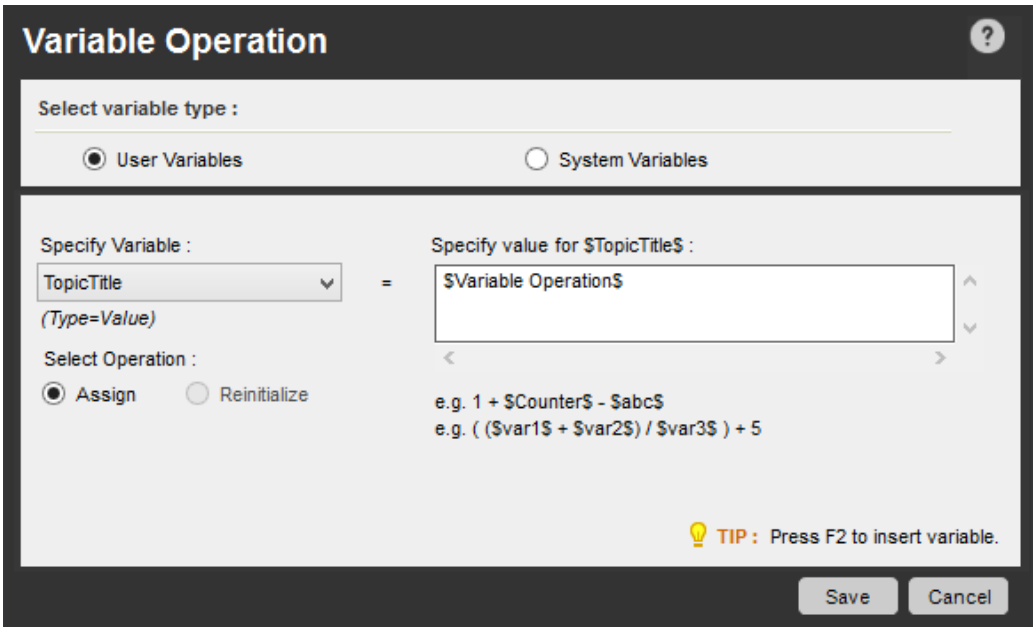
- **Assigning User Variables:**

  a. Select a user variable to specify a value to an existing user variable that you have defined.

  b. Assign a value. You can assign a value to the variable. The right side of the 'equal to' operator can be a regular operation; for example, you can specify: `$var1$ = ($var2$ + $var5$ - 10) / 5`.

> Note: The operators (,), /, *, + and - are supported. The left and right parentheses take precedence, and the operation within them is evaluated from left to right.

Example: In the expression `($Var1$ + 5) * ($Var2$ -10)`, the first calculation `($Var1$ + 5)` will be evaluated first, then `($Var2$ -10)`, and the result of both will be multiplied and assigned to the variable specified on the left side of the equals sign.

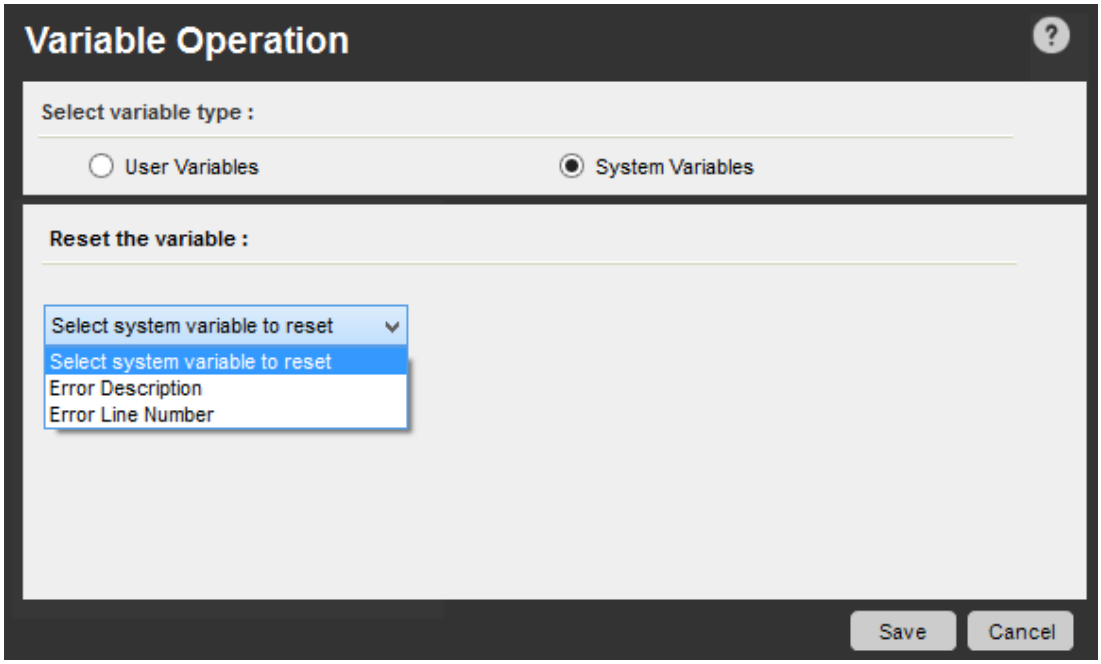


- **Assigning System Variables:**

  This option enables you to reset one of the following system variables:

  a. Error Description

  b. Error Line Number



Section 3: MetaBot Designer - Variables

19. Adding, Editing and Deleting Variables

Variables are storage locations for known or unknown information.

When building 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 is designed with various types of variables which can be defined for each Logic.

These are known as local variables. It has pre-defined system variables available for all tasks.

Some commands, like String Operations, provide a navigation for you to select and insert variables in your task.

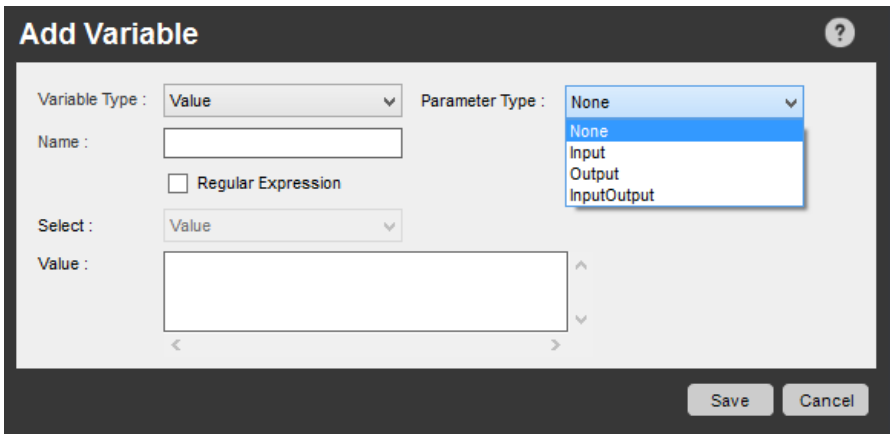
Adding Variables to a Task

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

Note: You can create 'Value' and 'Password' type variables in Logic.

Variable Types

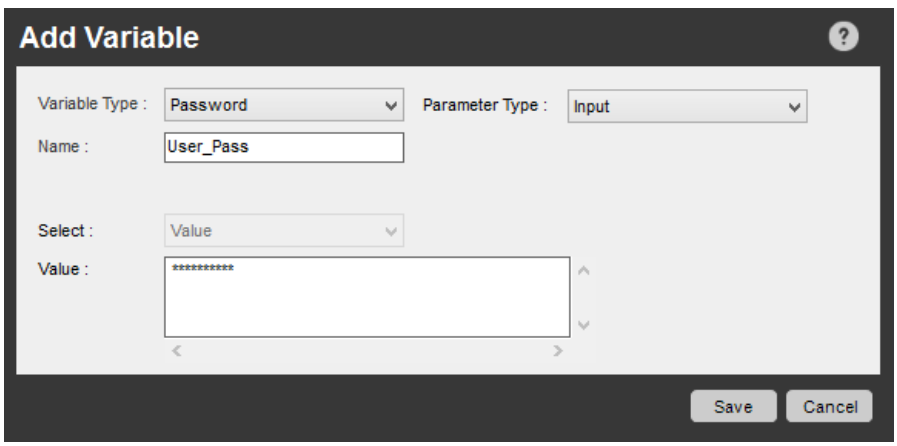
1. **Value -** You can use a value type variable when you need to hold a single data point and use it in multiple places. This "placeholder" value can represent either text or numeric data.



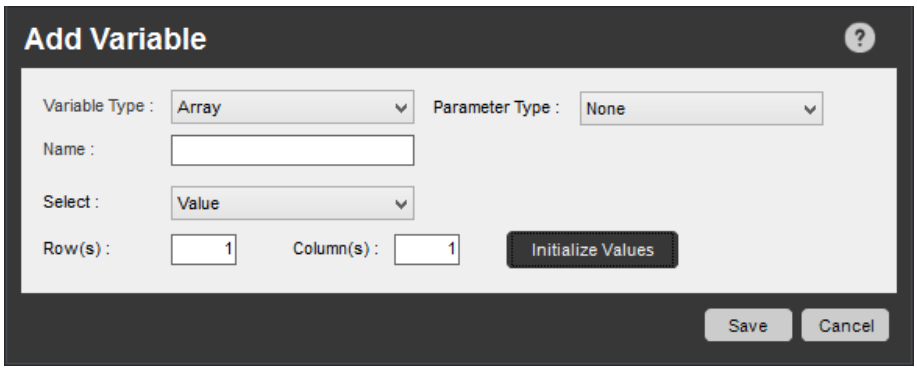
After you create the variable, you can use it by inserting the variable in several of the Logic Editor commands. When the value of the variable is modified, this value is reflected in any subsequent commands.

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

2. **Password -** You can use a password type variable when you specifically need to hold the data as a "Password". This value is encrypted by default.



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.



Assign Array Type variables to your Screens and DIls via commands and/ or values.

Note: MetaBot Designer supports Array type variables as 'Value' and 'Read from Text File'.

Parameter Types

You can select any of the following to assign as a parameter while designing your logic:

None - This is the local variable which will not be exposed for use in tasks once the logic is uploaded.

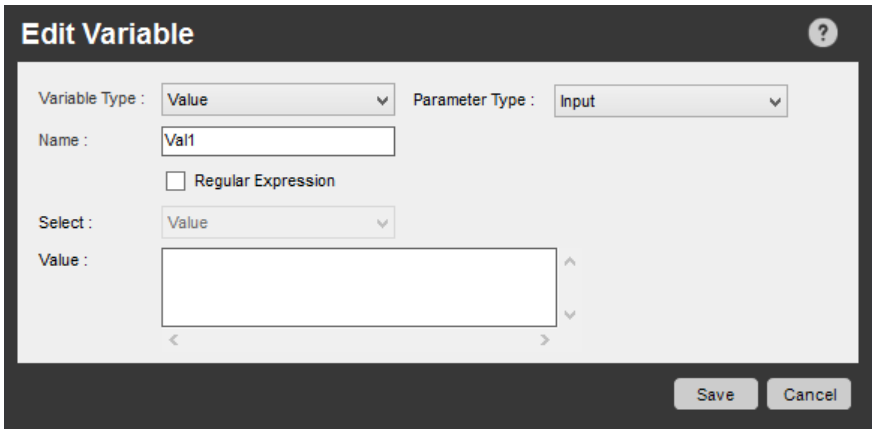
Input - This variable allows inputting values in the Logic.

Output - This variable allows outputting values in the Logic.

InputOutput - This variable allows both - inputting and outputting of values in the Logic.

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, or the method of determining value.
4. Click Save.

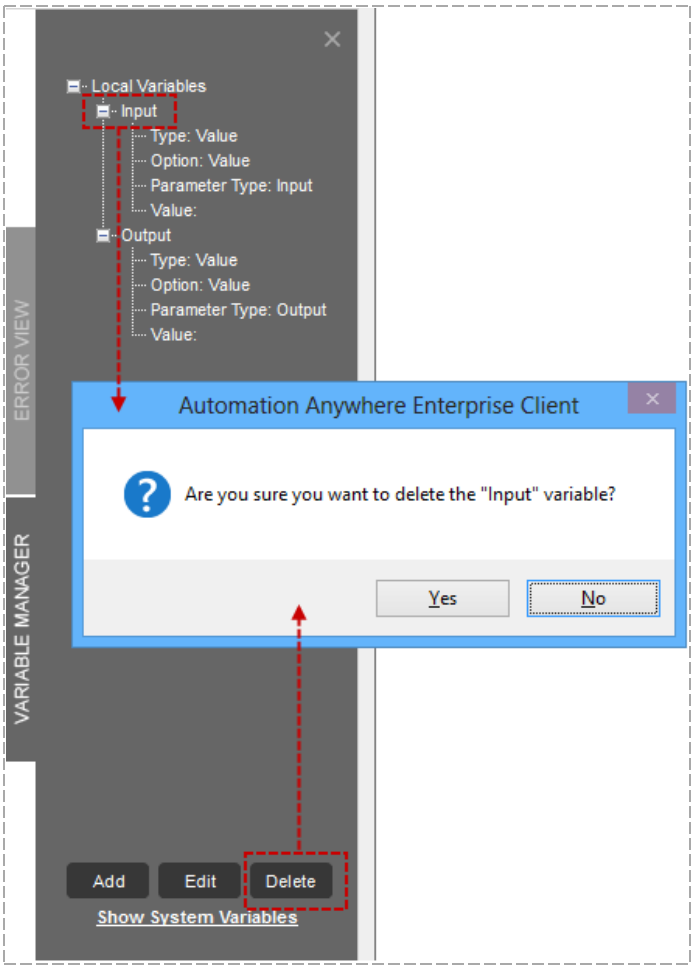
Deleting a Variable

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,

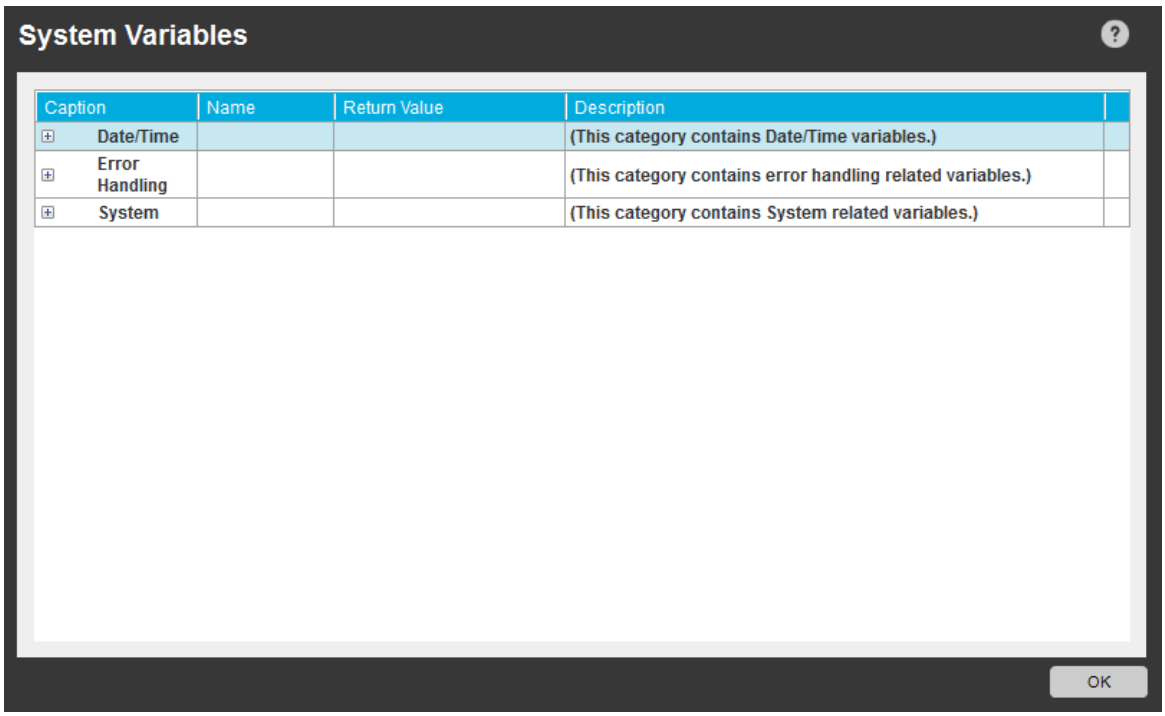
1. Select the variable you want to delete.
2. Click the Delete button or right-click on the variable and select Delete.
3. When the confirmation message is displayed, click yes.



Note: You can delete variables one at a time.

20. System Variables

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



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.

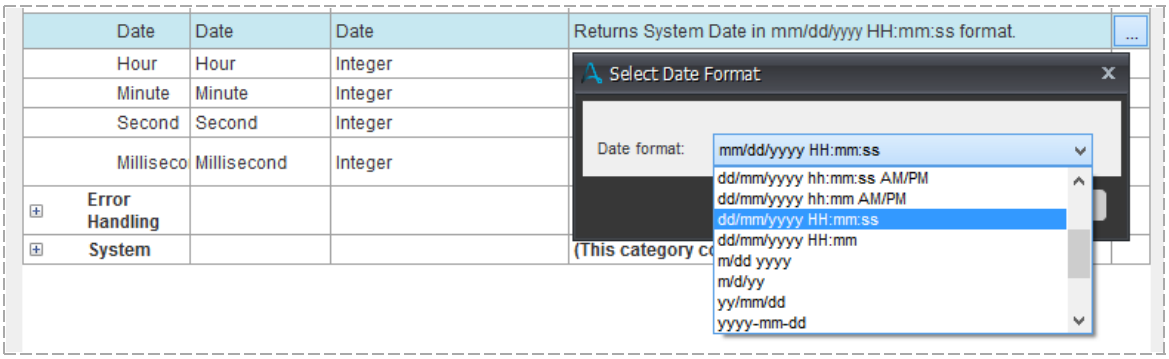
Date/Time System Variables

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



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.

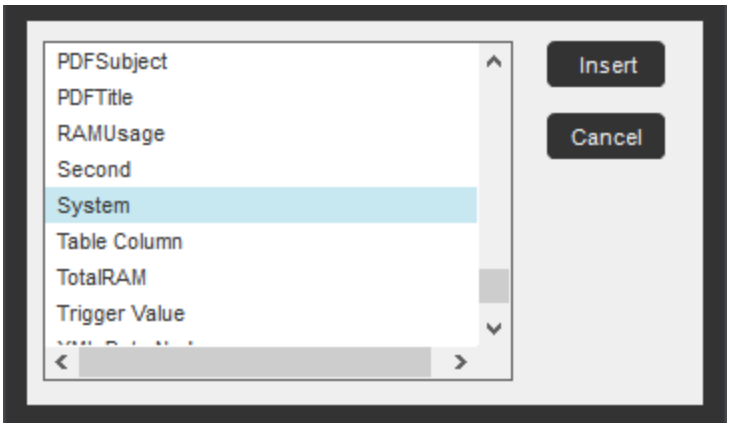
Common Use Case: These variables are useful when the performance of a system needs to be tracked during an activity; for instance load testing.

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

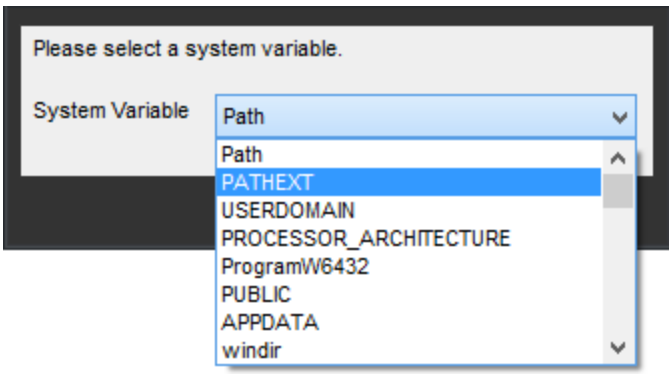
*When you select the System variable, a menu is displayed from which you can select the specific system variable (see steps below).

Steps to select System from Variables:

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



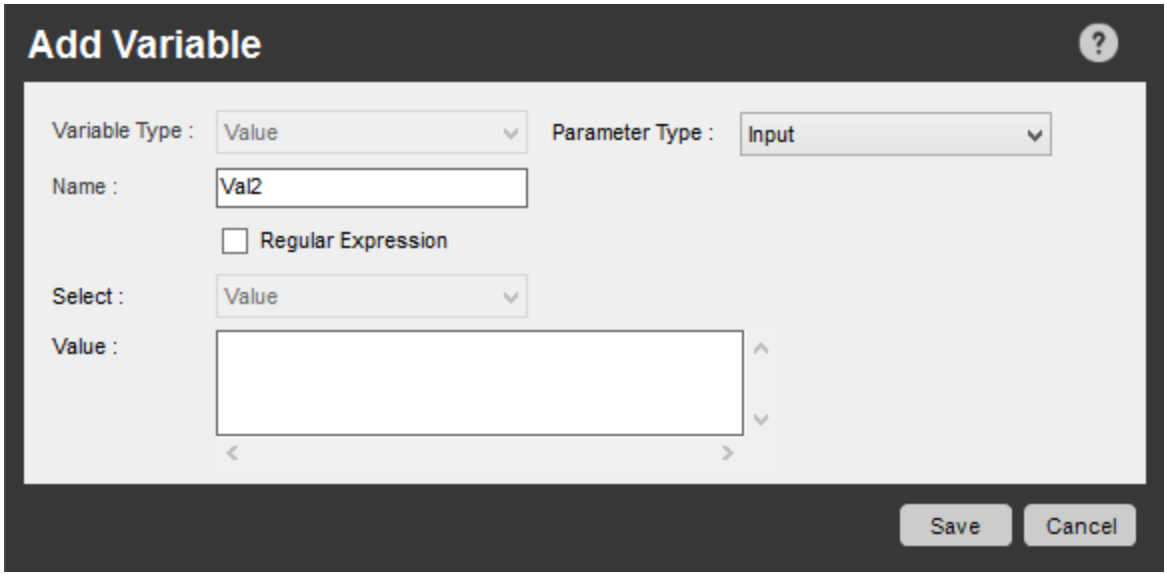
2. Select System and click Insert; a pop up window for System Variable Option appears.



3. Click OK and insert the System Variable.

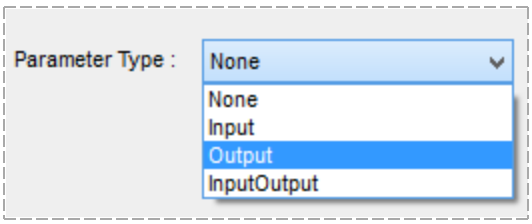
21. Variables - Parameter Types

MetaBot Designer uses Value Type variables by default. You can also define the parameters for the variables that you add. Parametrize variables for local use, to be only read from, to be set to or read as well set while designing your Logic Blocks.



Parameter Types

You can select any of the following to assign as a parameter while designing your logic:



You can select any of the following to assign as a parameter while designing your logic:

1. **None -** This is the local variable which will not be exposed for use in tasks once the logic is uploaded.
2. **Input -** This variable allows inputting values in the Logic.
3. **Output -** This variable allows outputting values in the Logic.
4. **InputOutput -** This variable allows both - inputting and outputting of values in the Logic.

You can use these when you want flexibility when assigning variables. Also assign 'Fixed' and 'Variable' values to this parameter in tasks.

22. Using Array Type Variables

An array variable is a two-dimensional variable that holds multiple values in a table of rows and columns. Arrays are very powerful for creating staging areas for data that need to be retrieved by your process as it runs.

Common uses of array variables include:

- Extracting data from web pages
- Extracting many rows of data from an Excel spreadsheet or a database
- Read or write data from/to a legacy system, an ERP system, or another application
- Filling out order forms with different fields from Excel to Database

The values can represent either text or numeric data.

After you create the variable, you can use it by inserting the variable in several of the MetaBot commands.

When the value of the variable is modified, this value is reflected in any subsequent commands that are run by the task.

Creating an Array Type Variable

You can create an array type variable in one of two ways:

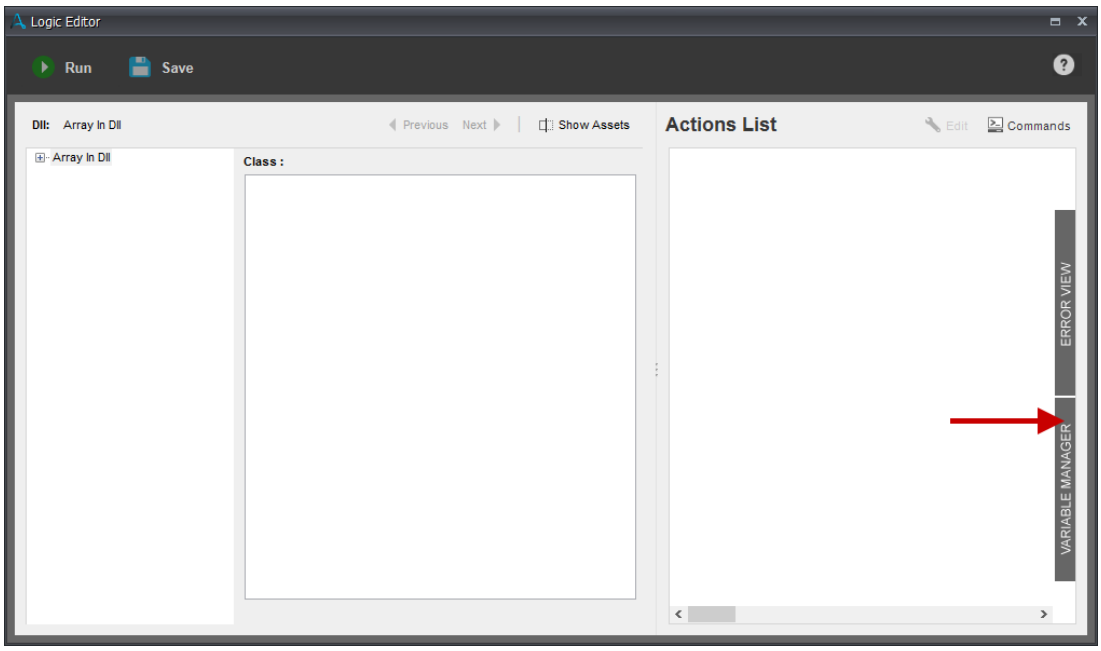
- Using direct assignment
- Using a text file

Direct Assignment

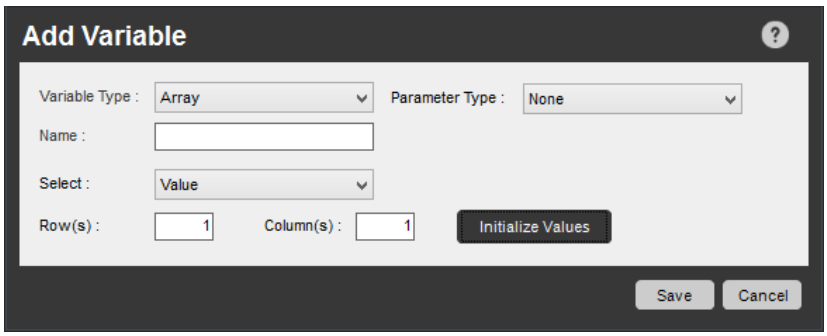
When the values of an array variable need to be defined directly, you can opt to use 'Value' Array type variable. This is termed as 'Direct Assignment'.

To create an array type variable using direct assignment, follow these steps:

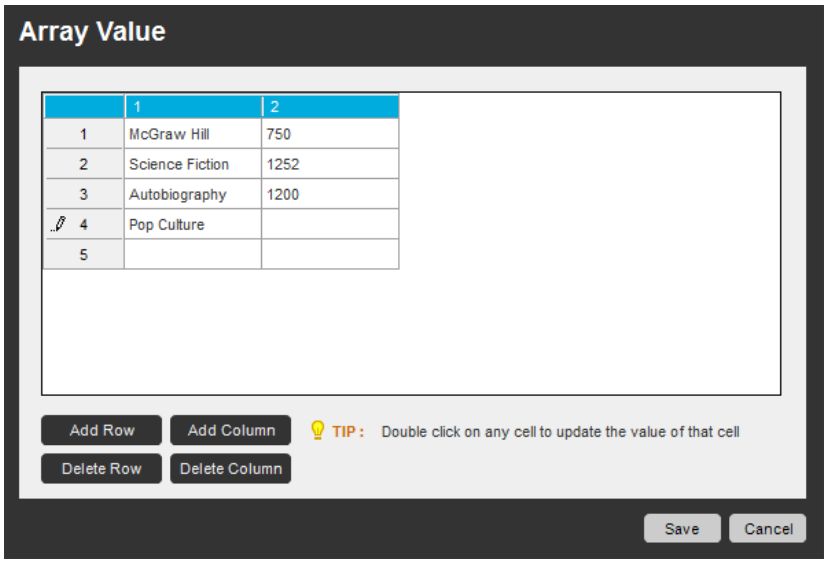
1. In the Logic Editor, click on the Variable Manager tab on the right side.



2. Click on the Add button. The 'Add Variable' window is displayed.
3. Select type 'Array'.
4. Enter a name for the variable. The name must begin with an alphabet and should not contain spaces.
5. Select how you want the array to be created. Default selection is 'Value' which indicates that you will define the values of this variable directly in it. .
6. Specify the number of rows and columns in the field provided. Default values are 1 X 1.



7. Click the Initialize Values button. The Array Value Details window is displayed based on the rows and columns provided in step 6.

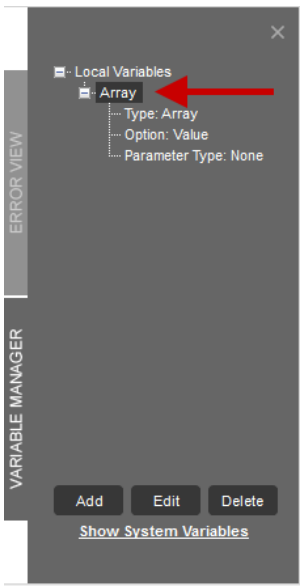


8. Enter the values for each cell.

> Tip: You can modify the array dimensions using the Add Row, Add Column, Delete Row, and Delete Column buttons.

9. Click the Save button to save the values.
10. Click the Save button in the Add Variable window to save the Array variable.

After the variable is saved, it is displayed in the Local Variables section of the Variable Manager.



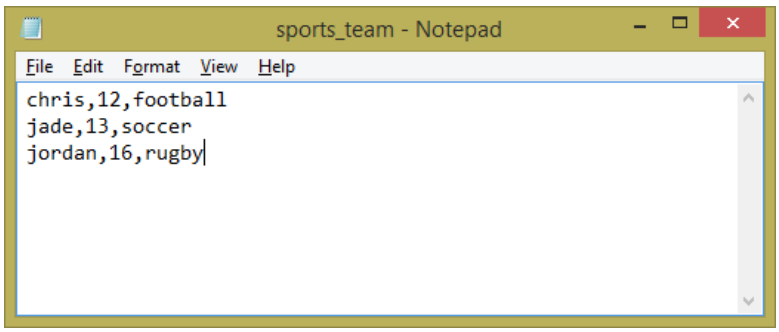
If you don't want to specify the value of the variable at the time you create the variable, MetaBot Designer allows you to set the values of variables using a text file. This is described in the next section.

Reading from a Text File

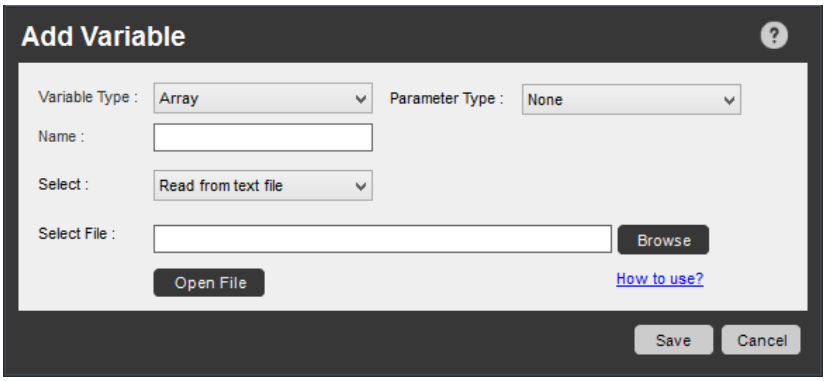
Create an array type variable using text file assignment when you want to read values from a specified text file into the array variable while the task is running. The text file is used to define the array variable.

To create an array type variable using text file assignment, follow these steps:

1. Determine the text file that you will use for assigning values.
    - For Rows enter the data in a new line.
    - For Columns use comma separated values.
    - Example: sports_team.txt



2. In the Task Editor, click on the Variable Manager tab on the right side.
3. Click on the Add button. The 'Add Variable' window is displayed.



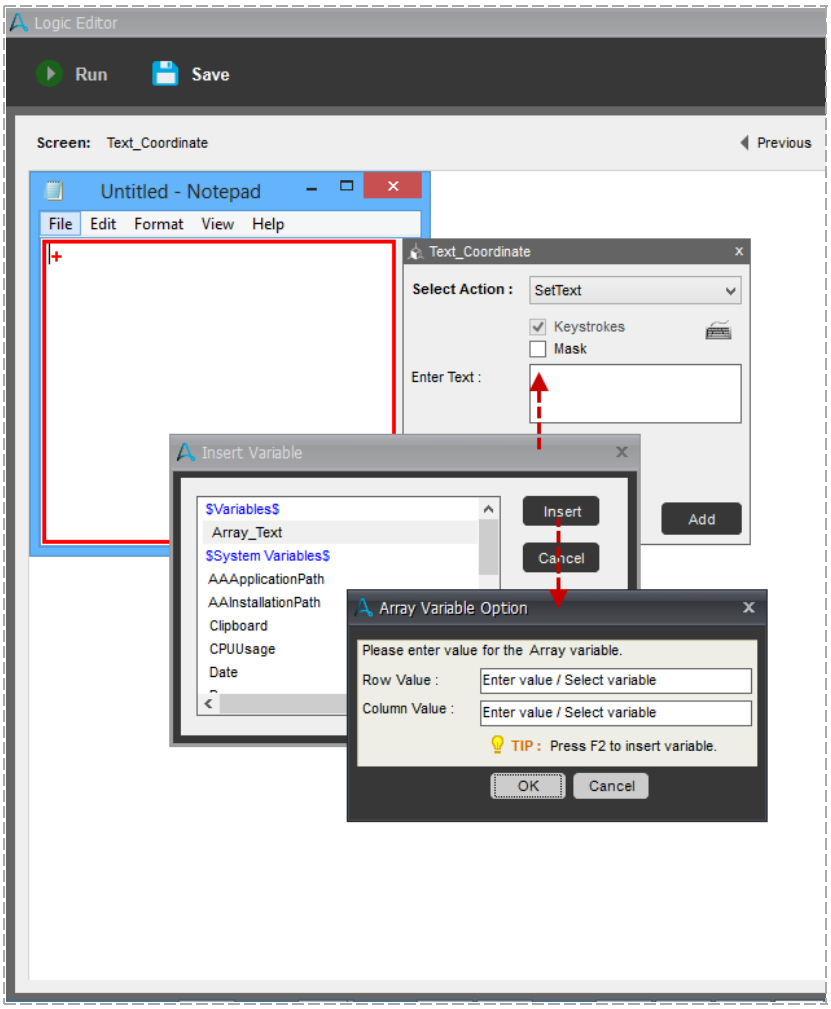
4. Select type 'Array'.
5. Enter a name for the variable. The name must begin with an alphabet character and should not contain spaces.
6. Select 'Read from text file' from the list.
7. In the Select File field, browse to the file or type the file path for the required text file. Use the Open File button to view the selected text file or to modify it.
8. Click Save.

Inserting Array Variables in Screens and DLLs

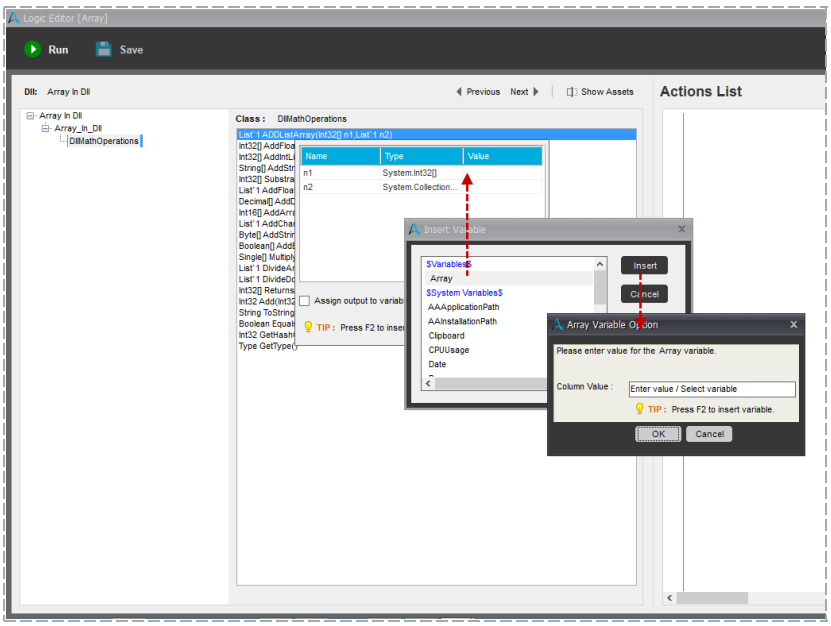
You can insert Array Variables while creating Logic in the Logic Editor. It can be used to assign values in Screens and/or Dlls.

Tip: You can insert rows and columns in MetaBot Commands for a Screen. However, when used with a DLL, you can insert only a single column to the Array type variable during assignment.

1. **Adding an Array variable to a Screen -** You can add an Array variable in Screens via MetaBot commands; namely Message Box, String Operation, and Variable Operation. You can also assign a variable in the Screen directly in the property window as shown:



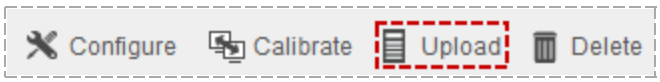
2. **Adding an Array variable to a Dll -** You can insert an Array variable while inputting values by pressing the function key 'F2' as shown:



Section 4: MetaBot in Enterprise Control Room

23. Uploading MetaBots to Control Room

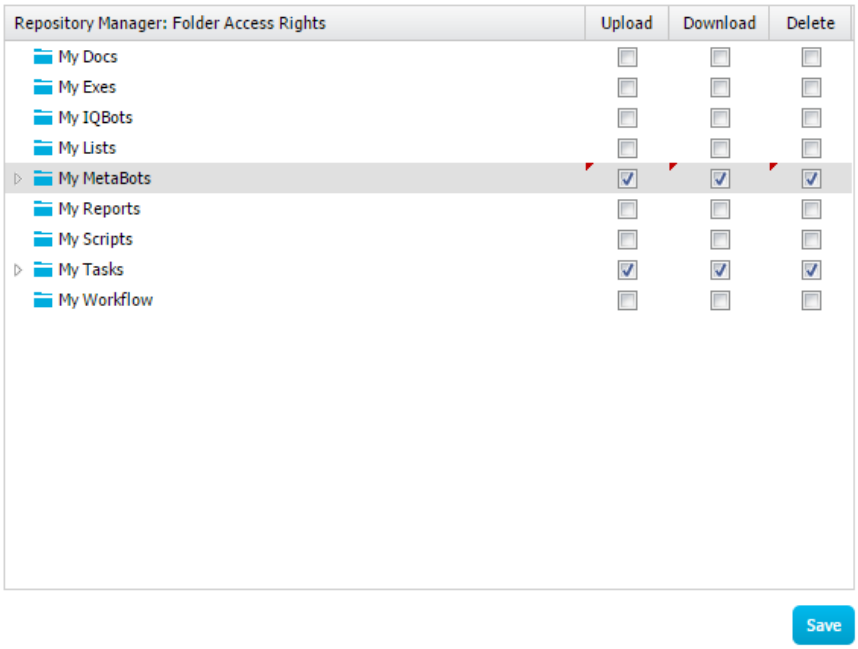
MetaBots, once created, can be uploaded from MetaBot Designer to Control Room from where any number of Automation Anywhere Enterprise Development Clients can download and use them.



Note: The Upload and Download features require logging in to the Control Room.

Providing Folder Access Rights in Control Room

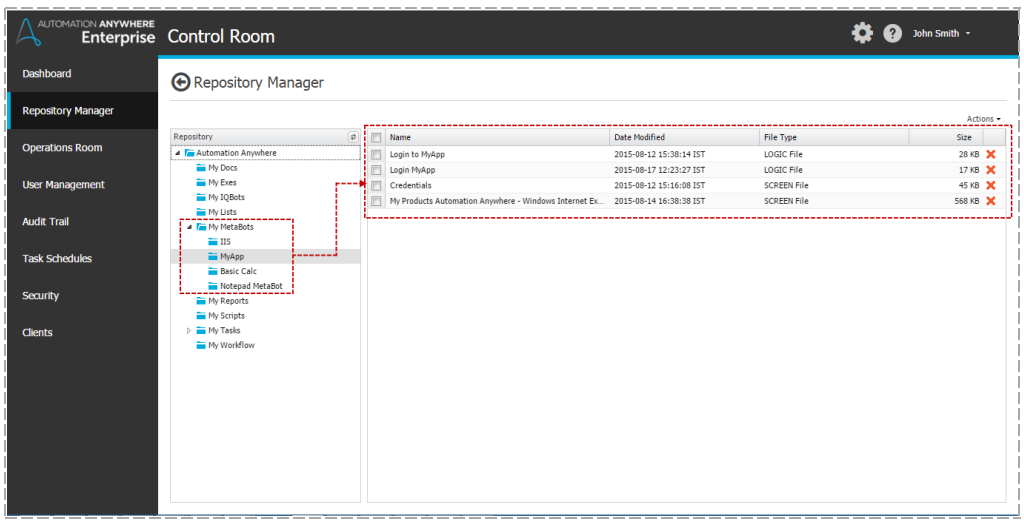
The Control Room administrator can provide appropriate rights to the Enterprise Client user to Upload, Download and/or Delete the MetaBots from the Roles and Permissions tab in Security console of the Control Room. Learn More



Note: All the users will be able to upload to/download from the My MetaBots directory on the Control Room.

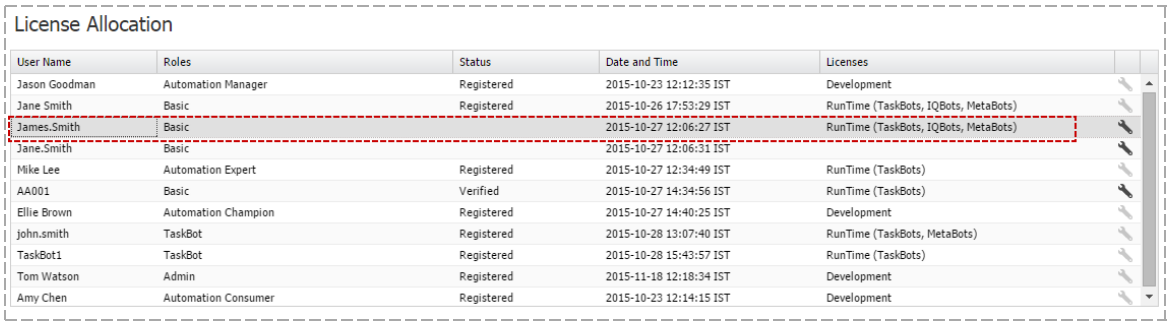
Accessing My MetaBots Folders

The Control Room administrator can access the MetaBots uploaded in the Repository Manager. Learn More



Managing MetaBot Clients

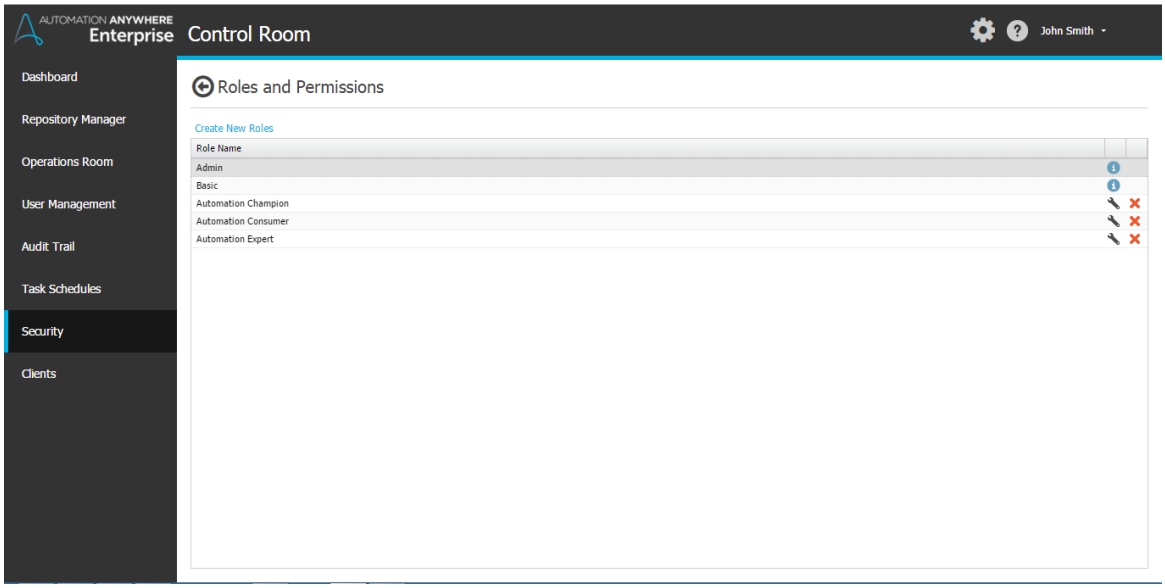
The Control Room administrator can manage the number of RunTime Clients for MetaBot Designer based upon the number of licenses allocated (for Run Time Clients) from the License Management tab. Learn More


License Allocation

User Name	Roles	Status	Date and Time	Licenses
Jason Goodman	Automation Manager	Registered	2015-10-23 12:12:35 IST	Development
Jane Smith	Basic	Registered	2015-10-26 17:53:29 IST	RunTime (TaskBots, IQBots, MetaBots)
James.Smith	Basic		2015-10-27 12:06:27 IST	RunTime (TaskBots, IQBots, MetaBots)
Jane.Smith	Basic		2015-10-27 12:06:31 IST	
Mike Lee	Automation Expert	Registered	2015-10-27 12:34:49 IST	RunTime (TaskBots)
AA001	Basic	Verified	2015-10-27 14:34:56 IST	RunTime (TaskBots)
Ellie Brown	Automation Champion	Registered	2015-10-27 14:40:25 IST	Development
john.smith	TaskBot	Registered	2015-10-28 13:07:40 IST	RunTime (TaskBots, MetaBots)
TaskBot1	TaskBot	Registered	2015-10-28 15:43:57 IST	RunTime (TaskBots)
Tom Watson	Admin	Registered	2015-11-18 12:18:34 IST	Development
Amy Chen	Automation Consumer	Registered	2015-10-23 12:14:15 IST	Development

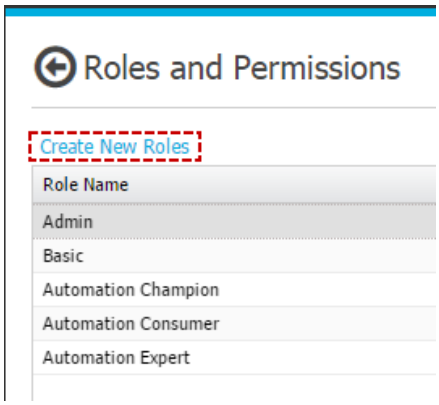
24. Creating Roles and Assigning Permissions for MetaBots

You can define user roles and assign the necessary permissions for using MetaBot Designer in the web based Control Room from the Security console.

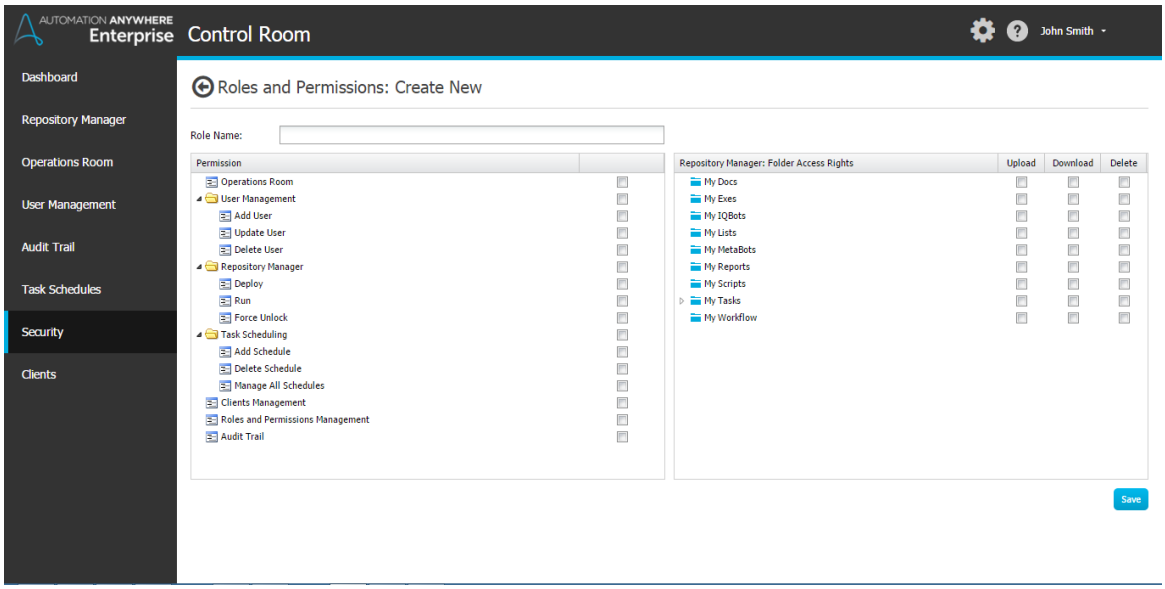


Creating New Roles and Assigning Permissions

To define new user roles, click on 'Create New Roles'.



In the 'Create New' roles page, input the role name:

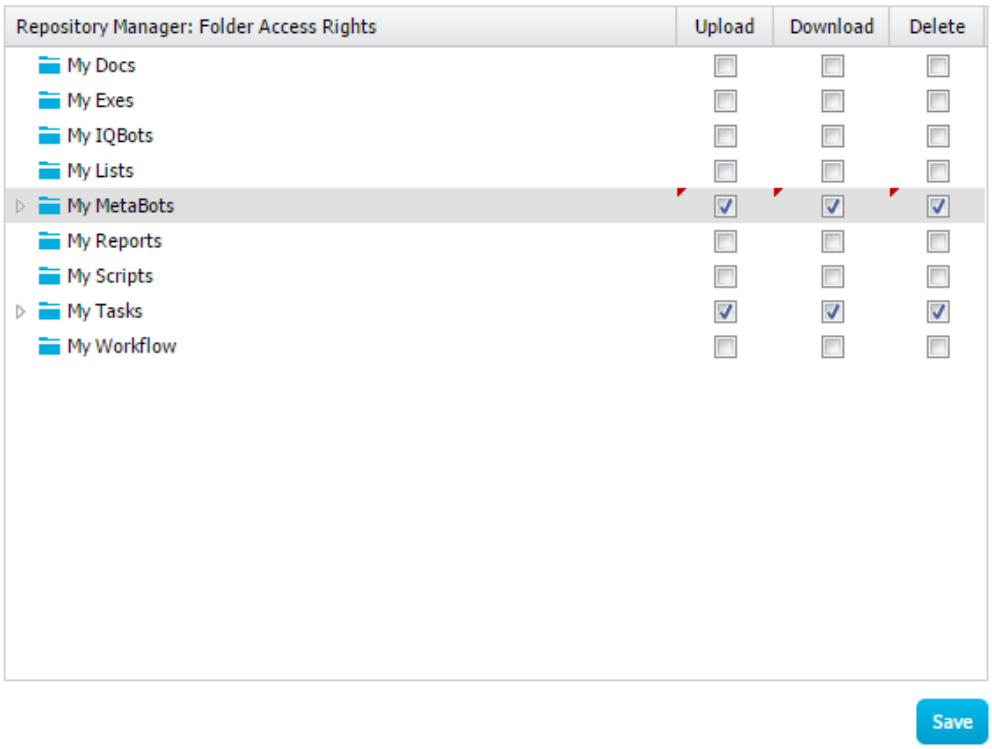


You can assign permissions based on the role you wish the user to play.

- For instance you can permit the new User to manage MetaBots only; in which case you can select the My Metabots folder.

You can then assign the User access rights that could include any or all - Upload, Download and Delete permissions for different folders.

- For instance, you can assign a User 'Upload', 'Download' and 'Delete' permissions to the 'My MetaBots' folder only.



Managing the User

You can verify whether the User has been created in the list in 'Roles and Permissions'.

To return click on .

Note: To edit the User roles and permissions, click on . To delete the User click on .

Section 5: MetaBot in Enterprise Client

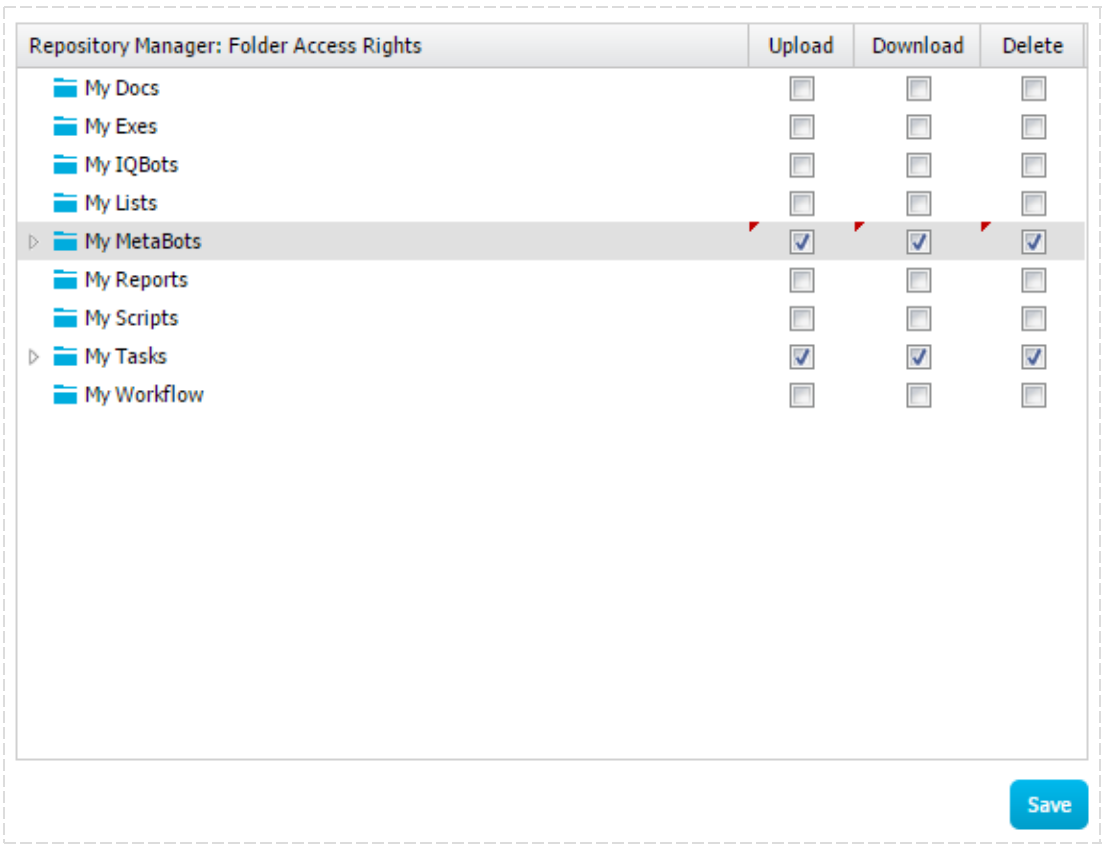
25. Using MetaBot Designer with Automation Anywhere Enterprise

To use MetaBots in Automation Anywhere Enterprise, you are required to create a MetaBot user on the Control Room, upload the MetaBots to the Control Room and download them in Enterprise Client. At the same time, if you have appropriate rights, create your own MetaBots and 'upload' those to the Control Room so that fellow MetaBot users can download automation integration.

Assigning/Revoking access permissions

To assign permissions to upload, download and/or delete MetaBots to the MetaBot user, follow the steps given below:

1. As a user with administrator rights you can assign or revoke appropriate privileges to upload, download and delete MetaBots to an Enterprise Client user.



   Refer Creating Roles and Assigning Permissions for MetaBots for details.
2. Login to MetaBot Designer using your user credentials.

Note:

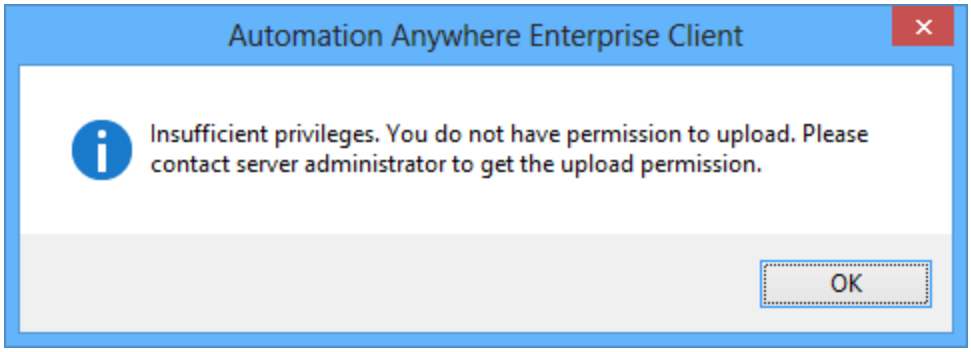
- The creation of MetaBot user is subjected to license settings of Control Room. The MetaBot licenses are consumed on a first logged in, first allocated basis.
- Same user credentials are required if accessing Enterprise client and MetaBot from same machine. If the user logs in to MetaBot Designer using a particular credential, the same credentials are used to auto login to Enterprise client upon launch. The opposite of this case is also true.
- If the MetaBot user on Control Room is created as a run-time client, the MetaBot will still get launched in a full-fledged mode.

Restricting a MetaBot user from uploading

To restrict a MetaBot user from uploading:

1. Simply un-check the upload permissions from the desired MetaBot MetaBot checkbox in Repository Manager > Folder Access Rights.
2. Click on Save.

- If the user now tries to upload, s/he would come across the following message:



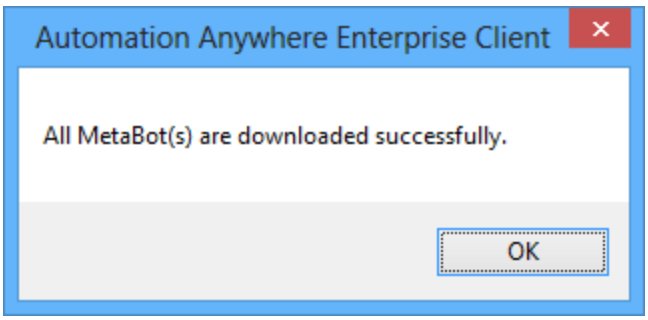
Tip: If you don't want the user to upload any MetaBot at all, just un-check the upload permissions for My MetaBots parent folder.

MetaBot in Enterprise Client

To add a new MetaBot/Sync an existing MetaBot -

1. Click on MetaBots link under Manage on the Client Main screen. This will invoke MetaBot Manager.
2. The MetaBot Manager shows a list of all the MetaBots that are available to be synced.
3. Select the desired MetaBot(s) and click on download.

- A success message will be displayed upon completion of download -



Refer Downloading MetaBots in Enterprise for details.

To add a new MetaBot/Sync an existing MetaBot -

- To view the synced MetaBots
    1. Invoke the editor, either to new task creation or to edit a new task.
    2. The MetaBot list appears on the top of left panel
- To use MetaBots
    1. Select and drag and drop a desired MetaBot on the editor to invoke the configuration window, in a similar manner to Command, and proceed with automation.

Refer MetaBot - Logic, MetaBot - Screens and MetaBot - DLLs for details.

Note:

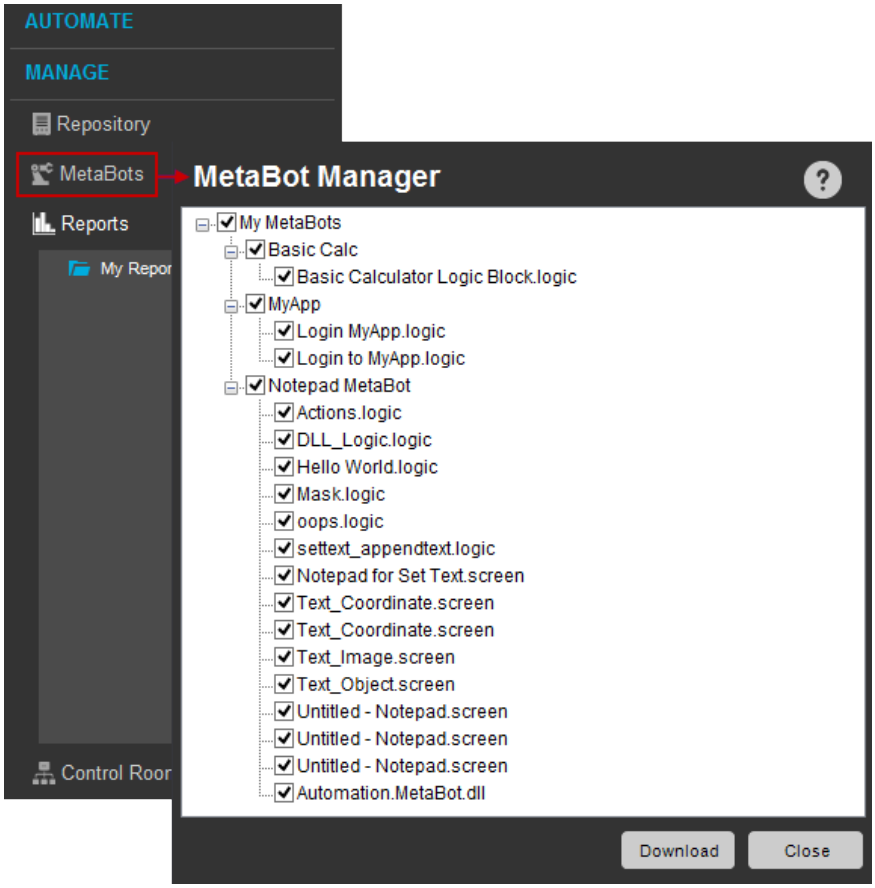
- MetaBots link under Manage on main screen/MetaBot list in Editor will be available only if an appropriate client license that enables the MetaBot feature is installed.
- For some reason, if the synced MetaBot doesn't appear in the MetaBot list in the Editor, click on the refresh icon on the MetaBot list.

26. Downloading MetaBots in Enterprise Client

MetaBot Manager in Enterprise Client is a launch pad for integrating the configured MetaBots into your automated processes. You can connect to the Control Room for synchronizing these if you have been given the consumption/download rights.

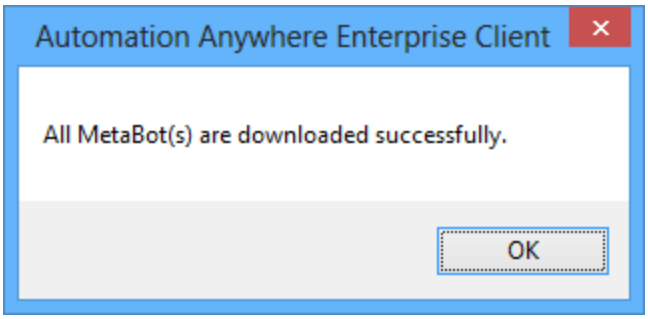
How to Sync

In the Manage panel of your client, launch the MetaBot Manager which will provide you visibility of the MetaBots that are uploaded to the server and ready for use.



Select the one(s) that you wish to integrate as commands and click 'download'.

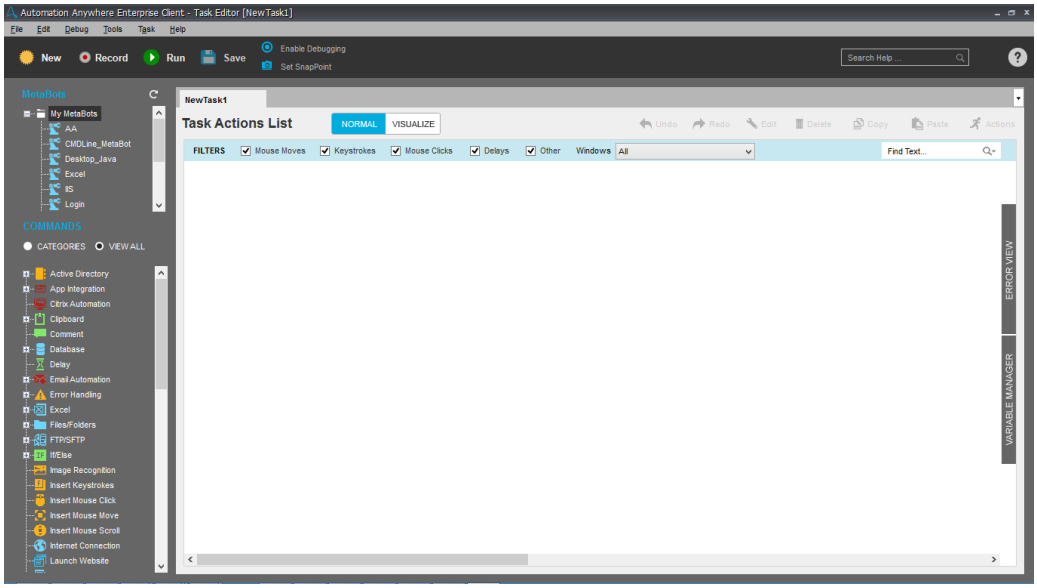
- A success message will be displayed upon completion of download.



You are now set to use the downloaded MetaBots in the commands for automated processes.

To add a new MetaBot/Download an existing MetaBot

- To view the downloaded MetaBots
    1. Invoke the editor, either to new task creation or to edit a new task.
    2. The MetaBot list appears on the top of left panel



- To use MetaBots
    1. Select and drag and drop a desired MetaBot on the editor to invoke the configuration window and proceed with automation.

Note:

- Metabot panel under Manage on main screen/MetaBot list in Editor will be available only if an appropriate client license that enables MetaBot Designer is installed.
- For some reason, if the synced MetaBot doesn't appear in the MetaBot list in the Editor, click on the refresh icon on the MetaBot list.

To understand how to create MetaBots using MetaBot Deigner, refer Adding and Recording a New MetaBot.

27. Using Metabots in Tasks

Use the Visual Captures i.e. Screens, Application APIs i.e. DLLs and Navigational Flows i.e. Logic Blocks that you created in MetaBot Designer in your automation tasks.

Using GUI based MetaBots

When you want to include GUI components into your tasks, use the Screen Metabots.

Learn More

Using API based MetaBots

When you want to include low level operations of an application without wanting to use the GUI, you can opt for DLL MetaBots.

Learn More

Using Navigational Flow MetaBots

When you want leverage the advantage of both - GUI components and API level properties, use the Logic MetaBots. These are basically, navigational flows created using the Screens and DLLs.

Learn More

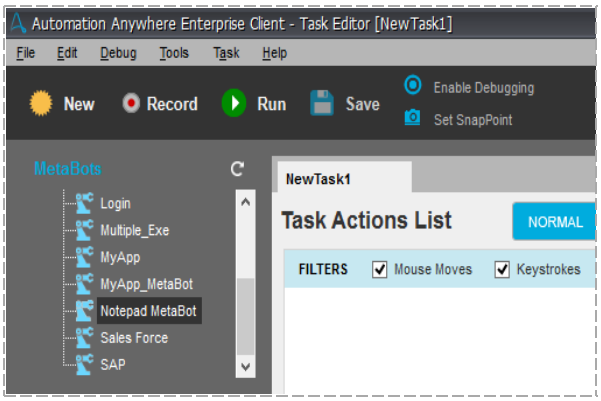
28. Using MetaBot Screens in Task

To work with UI objects, you can integrate a MetaBot Screen to your automation tasks in the Automation Anywhere Enterprise Editor

Adding MetaBot Screen to a Task

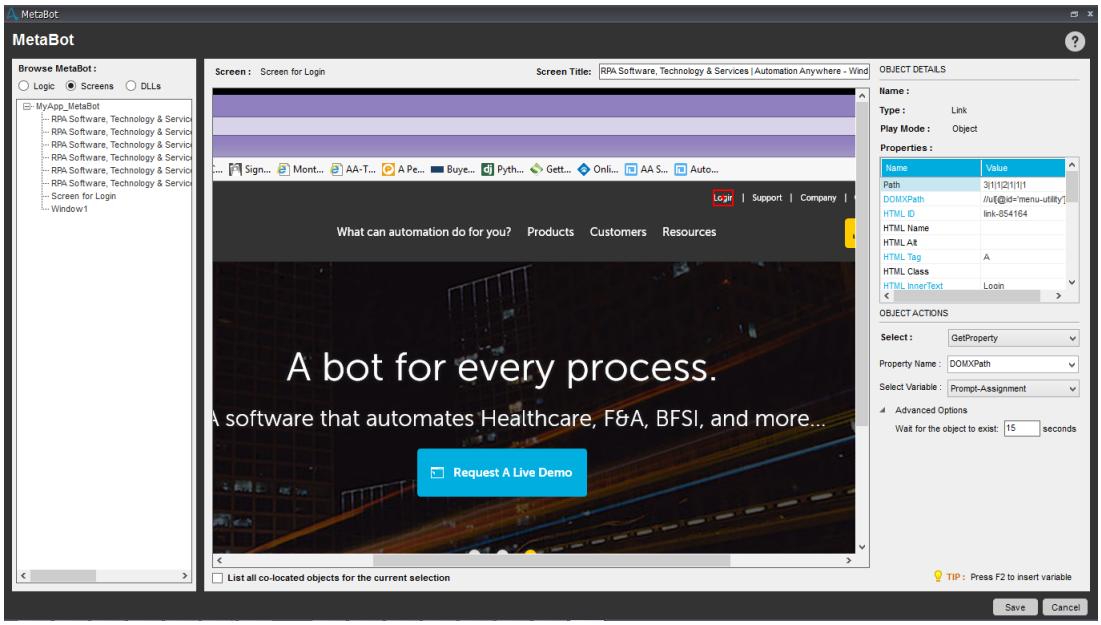
You can use the MetaBots in your tasks when working with UI based objects. The pre-configured MetaBot will ensure that your tasks that require a common application are set to work using a standard object based command.

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



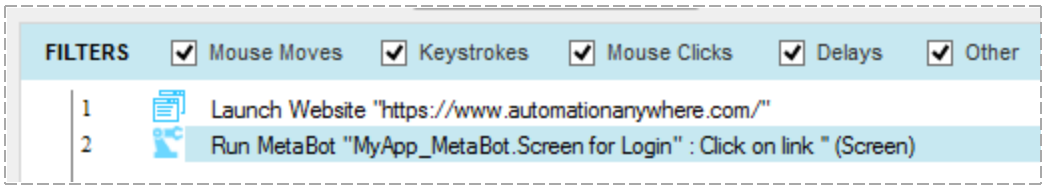
2. Double click/drag and drop a MetaBot from the list of MetaBots.

3. The MetaBot window is invoked.



4. Under 'Browse MetaBot', select 'Screen'. This will list one or more screens on the left pane.
5. Select a screen from this list. This will display the UI on the right pane.
6. Select the desired screen control (object) on this screen. This will list the selected control's properties and related actions alongside the screen UI.
7. Select an action and complete the input associated with that action.
8. Once done, click save to include the MetaBot screen object in the task.

The Task Editor depicts the event data for Screen:

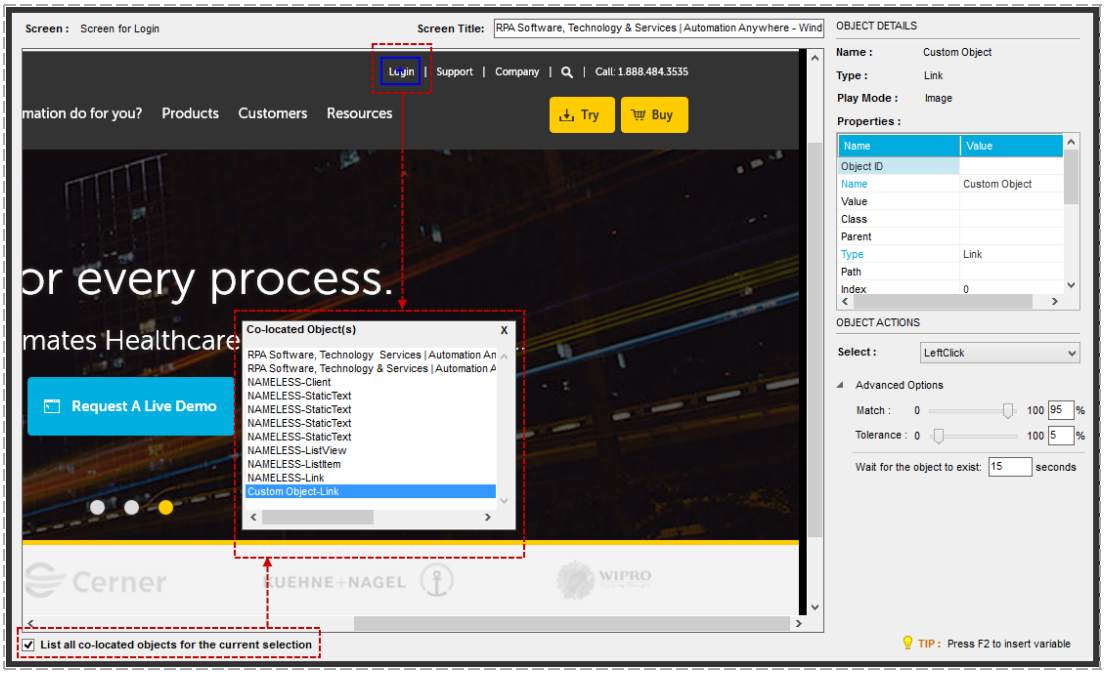


MetaBot Screen Command

- As seen in MetaBot Command, a Screen comprises various controls, properties, values and actions.
- Once you select a control on the screen in the MetaBot command, its Name, Type and Properties (Name/Value pair) get listed.

> Note: You can change the Value field of Window Title in Control Properties.

- Based on the Control Type, the 'Select Action' list is populated. Choose an action from a list, enter its associated inputs or assign a variable.
- Select the 'List all co-located objects for the current selection', to view a list of objects that are neighboring to the selected object. This helps in selecting the precise object control based on its parent object. Use this option when you are unable to select a particular control on the screen.

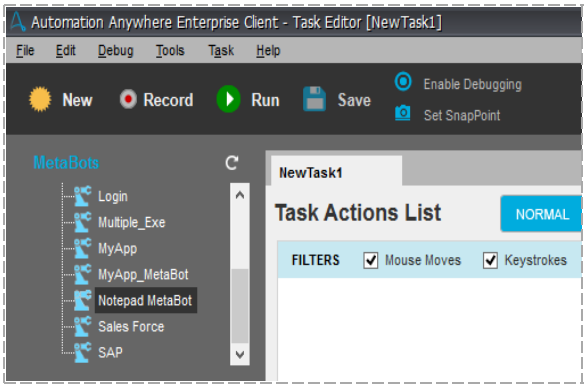


29. Using MetaBot DLLs in Task

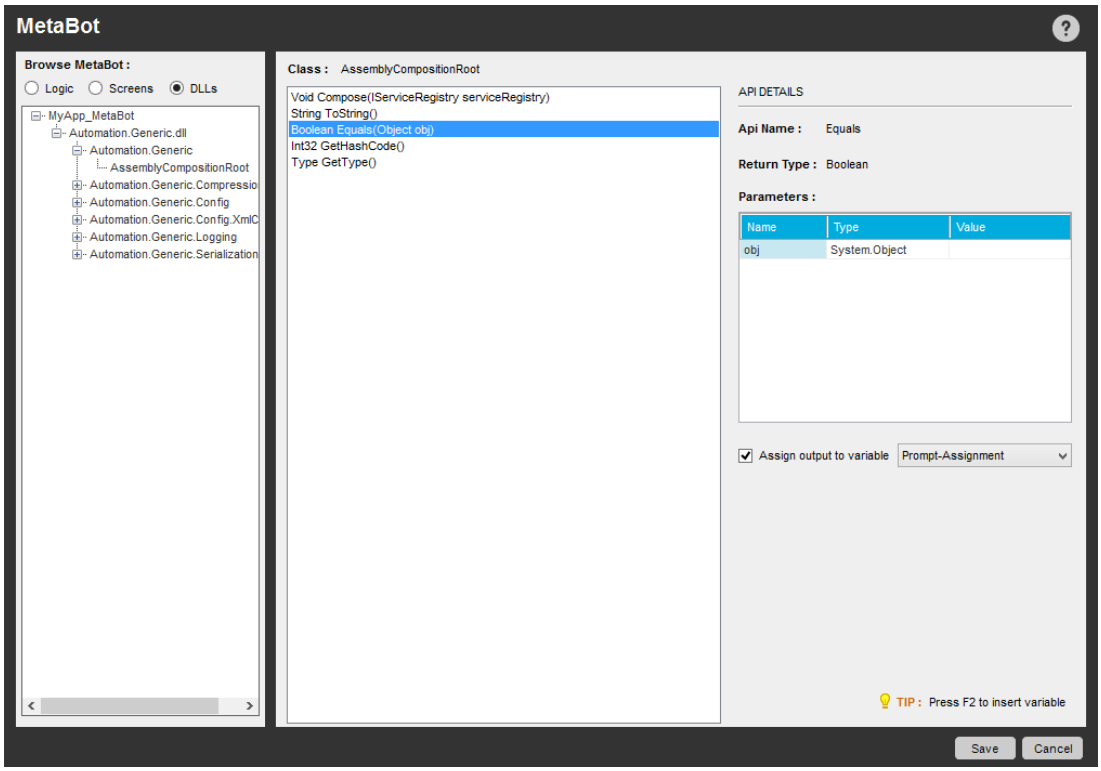
You can use a DLL in your automation tasks in the Automation Anywhere Enterprise Editor just like you would use a screen.

Adding MetaBot DLL to a Task

1. To begin, click on the MetaBots section given at the top of the command panel at the left in the Task Editor.

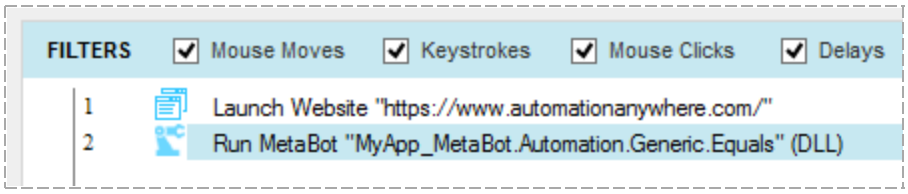


2. Double click/drag and drop a MetaBot from the list of MetaBots.
3. The MetaBot window is invoked



4. Under Browse MetaBot, select 'DLL'. This will list one or more DLLs in the left pane.
5. Expanding your desired DLL will reveal its namespaces and related classes.
6. Select a class from the list. The supported APIs for the selected class are displayed on the right pane.
7. From the list of APIs select the one for which you need to input a value. Alternately, you can also assign the output value to a variable.
8. Once done, click save to include the MetaBot Dll in your task.

The Task Editor depicts the event data for DLL:



More on DLLs in MetaBots

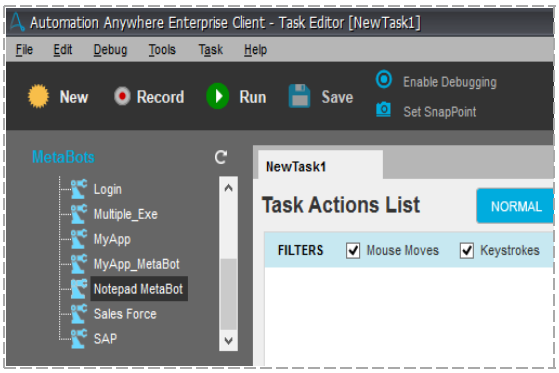
- Only those API's that are of Public type shall be visible in the command.
- Parameters that are of the Value type are supported. These include - System.Char, System.String, System.Int16, System.Int32, System.Int64, Float (System.Single), System.Double and System.Boolean.
- You can output the selected Value type parameters provided the DLL types are other than void.

30. Using MetaBot Logic in Task

You can integrate the pre-configured Logic blocks created using the MetaBot Designer in your automation tasks in the Automation Anywhere Enterprise Editor.

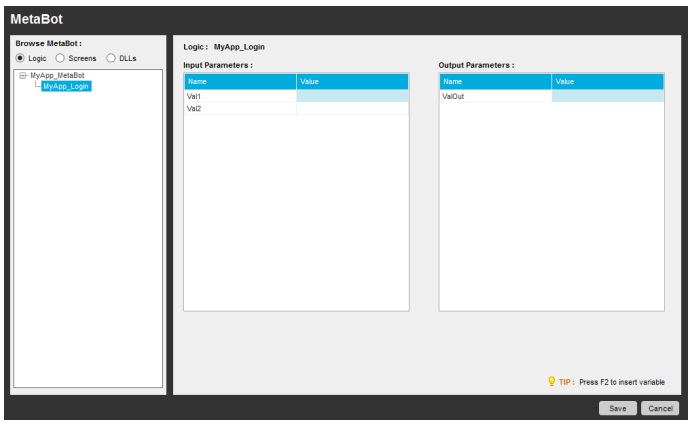
Adding MetaBot Logic to a Task

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



2. Double click/drag and drop a MetaBot from the list of MetaBots.
3. The MetaBot window is invoked.
4. Under Browse MetaBot, select 'Logic'. This will list one or more Logic blocks on the left pane.
5. Select the desired logic from the list. If this logic needs certain input/output parameters, they get listed on the right pane. Otherwise, the parameter tables appear empty.
6. 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.



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

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

The Task Editor depicts the event data for Logic: